1988

ANALYTICS User Manual

Preston K. Covey
Carnegie Mellon University, dtrollcovey@gmail.com

Follow this and additional works at: http://repository.cmu.edu/philosophy
Part of the Philosophy Commons
ANALYTICS User Manual

A Guide to the
Computer Tutorial Programs in Logic and Argument Analysis

Third Edition

Preston K. Covey, Jr.
Carnegie-Mellon University

Copyright (C) 1988 Preston K. Covey, Jr.
I. Using the TOPS-20 Systems

1. The TOPS-20 Systems: Which One Are You On?

Student users of the TOPS systems at Carnegie-Mellon University are distributed (for AY 1983-84) as follows:

- TOPS-B: H&SS Undergraduate students and external users
- TOPS-D: MCS Undergraduate students
- TOPS-F: CIT Undergraduate students

Since your account for any course, including 80-100, will be valid only on the TOPS system assigned for your particular college (irrespective of the college in which the course is offered), be sure to obtain a terminal that is linked to your assigned system.

Some terminals are 'hard-wired' to one and only one of the TOPS systems (and should be clearly so labeled). Other terminals (the majority available to you on campus) are plugged into a MICOM port contention device that allows you to link to any of the variety of systems on which you have an account. When using the MICOM linking mechanism, just be sure to ask for the TOPS system assigned your college (from the list given above). (See Using the MICOM Port Contention Device in Section 4 on how to gain access to your assigned system.)

2. Getting Started: User ID, Password, Account Number

To gain access to the TOPS-B or any other C-MU computer system, you first need three things: a User ID (and associated Directory for storing computer files), a Password, and an Account Number. These items, and how you find out about them, are explained below.

2.1 Your User ID

Your User ID is a four-character identification, usually consisting of your last and first initials and two other characters (usually a number and your middle initial, or two numbers). For example, my (Preston K. Covey's) user ID is PC0K.

Notice that the third character in my user ID is a zero 0 -- not the letter O. Many user ID's
contain a zero. Be very careful when typing any information into the system not to confuse the zero with the letter 0 (which are unfortunately very close to each other on most keyboards and which obviously look very similar) -- the computer cannot read your mind and will give you an error message if you type the letter 0 when it is expecting a zero. This is one of the most common of errors when typing information to the system, so watch carefully what you type.

On video/screen terminals (commonly called CRT's, cathode-ray-tube terminals), the zero is printed out with a diagonal line through it; on 'hard-copy' (paper-printing) terminals, the zero 0 is a more narrowly shaped oval than the letter O to help you distinguish them readily.

NOTE: You can find out your User ID by either of two means, as follows.

1 - If you are duly registered for 80-100 or some other course requiring work on the computer, your name and assigned User ID will be listed on the course section rosters posted on the wall in the 5100 corridor of Wean Hall (the first corridor to the left of the elevators on the fifth level of Wean Hall, the level with the cantelevered entrance off the cut between Wean Hall and Porter/Baker Hall). Section lists for 80-100 will also be posted on the Philosophy bulletin board in the corridor opposite 231 Baker Hall.

2 - Find a terminal (see Section 3 below), get the system's attention (see Section 4 below), and, when the system has prompted you with its 'at' sign, @, type:

   TR Yourlastname <CR>

TR is short for Translate, which is the command by which you get the system to give you the user ID assigned to a given name (or vice versa). When you ask the system to 'translate' your last name, it will print out a list of everyone with that name who has a valid user ID on the system. If you have been assigned a user ID, it will appear by your name in this list.

<CR>: You do not actually type the characters ' CR '. In computer manuals like this ' CR ' stands for 'carriage return' and means that you should hit 'Return', i.e., press the return bar. This is always necessary in order to actually send what you have typed on the terminal to the computer. The system will do nothing with any command or in-put you type until you press the carriage return bar.

Notice the space between the TR command and your last name in the example above: Be sure to place spaces between commands and items of information as specified in any instructions and illustrations! Failure to do so will cause the computer to be unable to read your input properly and will result in an error message.

NOTE: If you do not find a user ID assignment for your name either on the course section
lists in the 5100 WH corridor or on the system (using the Translate command), then:

1 - Give your full name and social security # (which is used for assigning you an initial password; see 2.2 below) either to your instructor or to the department secretary in 240 Baker Hall and request that an order be placed for a user ID on the 80-100 course account N595.

2 - Within a day of placing this request, begin checking the system for your assigned user ID using the Translate command (per instructions above).

Every user is allotted a certain amount of storage space on the system called a Directory in which to keep files, computer mail, etc. Your user ID placed within angle brackets < > serves as the name of your directory on any system. For example, the name of my (Preston K. Covey's) directory is <PC0K>. The computer programs that you will be using in 80-100 are stored in PC0K. Your user ID also serves as your address on any system, such that other users can send you computer MAIL routed to your directory by your user ID.

You will be assigned one user ID for all your work on any computer system throughout your career at Carnegie-Mellon. However, you will be assigned different account numbers for different courses (see Section 2.3), and you can change your password whenever you want (see Section 2.2 below).

2.2 Your Password

Your password is an alphanumeric (letter/number) code, presumably known only to yourself and the Computation Center, that is used to gain access to the system and your own directory. Your password is like the key to your house or room; it is meant to be exclusive and secret in order to prevent other people from using your allotted computer time or entering your directory.

Every user is assigned an initial password when he is first given an account on a system. Your initial password consists of the first eight digits of your social security number.

Because of this policy for assigning initial passwords, anyone can learn what your initial password is. So, to keep your password secure, one of the first things you should do is change your password. Any string of letters/numbers (no spaces) will do: choose something that only you know and that you can easily remember (up to eight characters, or more if you wish, although shorter passwords are easier to type without error). It's a good idea to change your password periodically or whenever you have any reason to believe that someone else has learned it. Do not write your password down any place where someone might see or identify it.

Changing Your Password: To change your password, when you have the system's attention (it
has prompted you with the 'at' sign, @), type the Set Directory Password command and hit 'Return.' The system will then prompt you for your old password. You type it in.

Notice that whenever you type your password in response to the system's request for it, it will not be printed out or 'echoed' on the screen. This helps secure its secrecy.

The system will then prompt you to type in a new password. Do so, and hit 'Return.' The system will prompt you to type the new password again, to be sure there's no error. Do so, and hit 'Return.' If you have made no error in typing, the system will again give you its @ sign, indicating that you have executed the change in password and that it is again at your disposal to take some other command.

This interaction will look like the following (top, next page), where what you type is represented in boldface (except for the passwords, which are 'silent,' and the reminders that you must press the 'Return' bar, CR , which does not appear on the screen):
@ Set Directory Password<CR>
Old password: <CR>
New password: <CR>
Retype new password: <CR>
@

If you make a mistake in typing or get an error message, begin again from the top with the Set Directory Password command.

2.3 Your Account Number

Your account number identifies the account to which your work on the computer is to be charged. Your account number for your work in 80-100 (for AY 1983-84) is N595. This account number should be used only for work in 80-100 and for no other purposes: the funds you are allotted for work in 80-100, as for other courses, are limited and should not be squandered on other computer activities. You identify the account to which you want your work charged during a given session on the computer when you login (see Section 5 below).

You may request an account for other computing purposes: consult User Services in the Computation Center. (User Consultants are on duty in WeH 5103 between 9:00 am and 4:30 pm, Monday through Friday.)

You can check to see whether you are validated for a given account or to find out what accounts you can use by using the I ACC command (the short command form for Information Accounts). When you have the system's prompt, @, type:

I ACC UserID <CR>

The system will then print out (1) the valid accounts for your user ID on the system between square brackets, e.g.: [N595], and (2) all the accounts for which your user ID is valid on any system, with the current total charges on those accounts.

You can use the I ACC command with your user ID specified as illustrated above to get information on your account(s) before you actually log in. After you have logged in under your user ID, the system knows who you are and you do not have to specify your user ID in order to get information on your account(s). After you have already logged in, you need type only:

I ACC <CR>

NOTE: When giving illustrations in this manual of what you should type in to the system, I will refer to your user ID, password, or account as: UserID, <Password>, or Account#. In these instances what you actually type are your own user ID, password, or account number, respectively.
3. Terminal Orientation: Locations and Types

3.1 Terminal Locations and Reservations

Computer terminals connected to the various TOPS systems at Carnegie-Mellon are available at several locations on campus, the most salient of which for student use are:

- **5200 Wean Hall:** The Public Terminal Room & 'The Cave' (5201 WeH)
- **Morewood Gardens** Dormitory
- **Margaret Morrison Plaza** (the 'store fronts' by the sororities)

Sign-up lists and reservation procedures are posted on the wall inside the Public Terminal Room, 5200 Wean Hall. Ditto for the other public terminal rooms. New sign-up lists are posted each week. You should make a point of reserving at least two or three hours on a terminal each week for 80-100 (preferably in the evening—see advice below). Read the reservation policy carefully. If you have questions or hassles regarding the reservation of terminals, see the User Consultant in 5205 WeH between 9:00 am and 4:30 pm, Monday through Friday.

Advice regarding terminal access and reservations: About one-third of the terminals in 5200 WeH will be available to H&SS students and connected to TOPS-B by means of a MICOM port contention device (see Section 4) with access to TOPS-B only. Margaret Morrison Plaza and Morewood Gardens both have several terminals available to H&SS students that connect to the various TOPS systems by means of a 'free for all' MICOM unit.

Terminals whose traffic is handled by MICOM units must contend for open ports to the computer systems. During peek traffic hours (9:00 am to 5:00 pm), priority is given to faculty and graduate users on the systems by the MICOM. This means that students have a low expectation of finding an open port to a TOPS system between 9:00 am and 5:00 pm.

Thus, students are well advised to plan to do their computer work and to reserve terminal time in the evening or early morning -- before 9:00 am, or after 5:00 pm. The terminal rooms themselves are open 24 hours, so computer work is possible at any hour (except for periodic 'down time' for system maintenance).
3.2 Terminal Types: Check Out Your Terminal

There are both CRT (cathode-ray-tube) screen terminals and 'hard-copy' (paper printing) terminals available to you. The CRT's outnumber the hard-copy terminals and as a rule the screen terminals are nicer to work on and easier to read for most purposes. So, as a rule, you will want to get a screen terminal. Hard-copy terminals are convenient, however, if you should want a running printed copy of your session on the computer. (There are other convenient ways of getting a printed copy of your work on the computer, using the Photo and Print commands. So the desire for printed copy needn't dictate your choice of terminal. See the TOPS Systems Manual, published by the Communications Design Center and available through the bookstore, for information on the Photo record and Print command as well as other useful system options.) Hard-copy terminals will always be adequate if not optimal for your work in 80-100, if no screen terminals are available.

The keyboards of the various terminal types are all basically similar to typewriter keyboards, with slight variations in the locations of various keys. However, terminal keyboards include special keys that perform crucial functions. You should be aware of the variations in terminal keyboards. Salient features of the different terminal types that you should know about are noted below (Section 3.2.2).

Be sure to check out the keyboard on your terminal to see that the crucial keys are set properly. (See Section 3.2.2 below.)

Be sure to notice what type/brand of terminal you are on: certain programs will ask you to specify the type of terminal you are using (because they are programmed differently to address different terminal types). (See Section 3.2.1 following.)

3.2.1 Check Out Your Terminal Type

Sometimes the system will need you to specify the type/brand of terminal you are using. Sometimes a program will ask you to tell it what terminal type you are using also (because it is programmed differently to address different terminal types).

The ANALYTICS programs are designed to address five types of terminals. These are listed below, along with what you should tell any 80-100 program when it asks you to identify your terminal type if that is different from the terminal's given name.

The name or designation of any terminal is printed somewhere on the top / front of the terminal chassis.

1. The Perkin-Elmer 1100
2. **The Perkin-Elmer 550** (also called the Bantam)

3. **The VT52 Decscope**

4. **The Zenith**: This terminal is addressed as if it were a VT52. So, if you are using a Zenith and an *ANALYTICS* program asks you to identify the type of terminal you are on, tell it you are on a VT52 Decscope.

5. **The LA36 Decwriter II**: Whenever you are on a Decwriter, tell the *ANALYTICS* program that you are on a 'Hard-Copy' terminal.

### 3.2.2 Check Out Your Terminal's Keyboard

There are certain keys with crucial functions that you should know about. Some of these keys have slightly different locations on different terminal types. The first thing you should do upon sitting down to work on a terminal is check out the location of these crucial keys and make sure that they are set properly for the work you want to do. (Read the advice for the Perkin Elmer 1100: whatever terminal you are on, much of this advice applies.)

1. **The Perkin-Elmer 1100**: The On/Off switch is located at the bottom right of the backside of the chassis. If there is no printing on the screen, turn the terminal ON. (It will take any screen terminal a few seconds to warm up: it's ready to go when the cursor or 'Port Group?' message appears on the screen.)

   The **LINE** key should be DOWN. This key is located among the eight special keys grouped at the far left of the keyboard. Only when it is locked in the DOWN position is the terminal 'On Line' and able to communicate with the computer system.

   The **Scroll Enable** key should be DOWN.

   If you want everything you type to appear in UPPER CASE, the **U/C** key should be DOWN. If you want a choice between UPPER and lower case (by using the SHIFT key as on a typewriter), the **U/C** key should be UP.

   The other five keys among the special eight keys grouped on the far left of the keyboard should all be UP.

   The **DEL** key located on the upper right portion of the keyboard is used to simultaneously backspace and **delete** characters, one by one. This is one way to erase mistakes. On the Perkin-Elmer 1100, in order to delete characters, you must press the **SHIFT** and **DEL** keys simultaneously.

   Do NOT try to use the **BACKSPACE** key to delete/erase mistakes. This will not work (even though it appears on the screen to erase) and will result in confusing error messages!!!!!!

2. **The Perkin-Elmer 550**: The On/Off switch is on the right front of the chassis, in the recess under the frame around the screen.

   The **LINE** key is among the four special keys at the top of the keyboard and should be DOWN.
The DEL key is in the upper right corner of the keyboard. Use this key to delete/erase mistakes. Do NOT use the BACKSPACE key for this purpose!

3. The VT52 Decscope: The On/Off switch is on the right side wall of the console. There is no LINE key. The DELete key is in the upper right portion of the keyboard. Use it to delete/erase mistakes. Do NOT use the BACKSPACE. The cursor is a flashing bit of underlining _ rather than a rectangular light (as on the Perkin Elmers).

4. The Zenith: The On/Off switch is around the back of the console, bottom right. Locate the LINE key and be sure it is DOWN.

5. The LA36 DecwriterII: The On/Off switch is in the upper right corner of the keyboard area.
   
   The LINE / LOC. key is among the eight keys grouped on the left of the keyboard. Contrary to the cases above, it should be UP, as should the other keys.

4. Getting the System's Attention: MICOM & Control-C

Any terminal you use in the public terminal rooms will be either 'hard wired' directly to a TOPS-20 system or else connected to a MICOM port contention device that will allow you to select among available systems. Your first task in gaining access to a system will be to get the system's attention. If your terminal is connected to a MICOM unit, the procedure for connecting to a TOPS system and getting the system's attention requires an extra step. The procedure for each case (MICOM and direct hard-wired terminals) is as follows.

4.1 Getting the MICOM's Attention

MICOM-connected terminals will be so labeled. If no printing or cursor (rectangular light or flashing line) is evident on the screen, turn the terminal ON. (See 3.2.2 above for location of On/Off switch on various terminals).

If you are at a Decscope terminal that has no LINE key and if the terminal is already ON, turn it OFF and then ON again.

If you are at a terminal that has a LINE key and if the terminal is already ON, punch the LINE key in and out a couple times. Make sure you end up with the LINE key DOWN if you're on a Perkin Elmer or Zenith terminal. Make sure that you end up with the line key UP if you're on a Decwriter printer terminal.

NOTE: Turning a terminal OFF and then ON, or taking it OFF and then ON LINE again, will get the MICOM's attention.
The MICOM will then ask you what Port Group you want, as follows:

**Port Group?**

If you are an H&SS student, your accounts and directory are on TOPS-B and you should respond by typing:

```
CMCCTB <CR>
```

(Remember the CR is not something you type out: it means that you are to press the 'Return' bar in order to send what you have typed in to the system.) If you are an MCS student, your accounts and directory are on TOPS-D and you should respond by typing: **CMCCTD CR**. If you are in CIT, you type: **CMCCTF CR**. The letters 'CMCCTB,' 'CMCCTD,' 'CMCCTF' are the computer network designations for the TOPS-B, TOPS-D, and TOPS-F systems, respectively: 'CMCC_' stands for 'Carnegie-Mellon Computation Center.'

When you have identified the TOPS system to which you want to be connected, and if there is a free port and everything is A-OK, the MICOM will give you the go-ahead by saying 'GO'. A successful transaction with MICOM, once you have gotten its attention with the ON / OFF or OFF-LINE / ON-LINE maneuver, will look like this (where what you type is represented in boldface):

```
Port Group? CMCCTB <CR>
GO
```

You are now ready to get the attention of the TOPS system. (See Section 4.2 below.)

If there is a problem getting a port, or the system you want is down, or you have made a typing mistake, MICOM will respond with a variety of messages, like:

- Unassigned Disconnected
- Unavailable Disconnected
- Busy Wait 001

To simplify your alternatives greatly, at this point you may either forget it and come back later, or else you can have another go at getting a port. You should try again a couple times to be sure you didn’t make a mistake yourself.

To get the MICOM's attention and the 'Port Group?' prompt again, simply turn the terminal OFF and then ON again. Or, if the terminal has a LINE key, take it OFF and then ON line again. Anytime you want to start again with the MICOM's 'Port Group?' option, just repeat one of these ON/OFF maneuvers.

When you get the MICOM's go-ahead, GO, proceed as below to get the system's attention (Section 4.2).
4.2 Getting the Systems Attention: CTRL-C

If you are at a terminal that is connected to the MICOM, you must first identify the
system/port group you want and get the MICOM's go-ahead (see Section 4.1 above).

If you are at a terminal that is connected directly to the TOPS system you want, you need first
make sure the terminal is turned ON.

You are then ready to get the system's attention. Locate the **CTRL** (Control) key at the left
side of the keyboard.

**Simultaneously** press the **CTRL** key and the key for the letter **C** -- be sure they are both
pressed simultaneously/together, not one after the other!

This maneuver will be referred to as a **CTRL-C** (Control-C). In this manual, **CTRL-C** or
<CTRL-C> will mean that you are to press the CTRL and C keys simultaneously.

When you perform a **CTRL-C**, the system will respond by identifying itself and then print its
prompting character, the 'at' sign @, as follows:


@

The system's prompt, the 'at' sign @, signals that the system is 'at' your disposal waiting for a
command.

You have gotten the system's attention. At this point, if you want to work on or with the
computer, you must gain access to the system by requesting that it 'log' you in and by identifying
yourself (by user ID and password) and the account number to which your work will be charged.
(See Section 5 below on the Login procedure.)

At this point, before or without logging in, there are certain sorts of information that you can
and may want to get from the system. For example, you may want to find out whether you
have been assigned a valid user ID on the system, or whether your user ID is validated for a
given account.

To find out whether you have a valid user ID recognized by the system, use the **Translate**
command (as described in Section 2.1 above).

To find out whether your user ID is valid for a given account number, use the **I Acc**
command (as described in Section 2.3 above).
5. Gaining Access to the System: The LOGIN Procedure

You are ready to **login** (gain access to the system) when you have gotten the system's attention by a **CTRL-C**, the system has identified itself and prompted you with the '©' sign as follows:


©

At this point you type the **Login** command, a space, your **UserID**, a space, the account# (N595 for 80-100), and hit 'Return'. At the '©' sign you type:

**Login UserID Account# <CR>**

If you have made any mistakes (forgot to leave spaces between entries, typed 0 instead of 0, etc.), the system will give you an error message and you need to begin your login again. If you have given the login command and information correctly, the system will then prompt you to type in your password as follows:

**Password:**

You then type in your password (the first eight digits of your social security number if it is the first time you've logged in, or whatever you've changed your password to). Be careful typing your password: remember, it will not be printed out or 'echoed' when you type it in, so you won't be able to see what you're typing! Remember to hit 'Return' when you've typed in your password.

If you correctly perform your login, the system will then print out information about your job and terminal identification numbers, the date and time, and it will again prompt you with its '©' sign to indicate that it is now 'at' your disposal to receive whatever commands.

If you fail to execute a correct login, the system will print some appropriate error message and you will have to start the login procedure again from the top. If you take too long in logging in, the system may log you out. If this happens or you want to be sure you have the system's attention, perform another **CTRL-C** and start again.

A successful login for Preston Covey (PC0K) on account N595 (the 80-100 account) would look like this (my password of course remains 'silent'/secret; except for my password, what I type in is represented in boldface):


©Login PC0K N595 <CR>

**Password:** <CR>

Job 8 on TTY3 11-Sep-83 13:44:31

©

Having executed a successful login like the above, you are now ready to enter some command to
the system. For purposes of doing work for 80-100 you will want to run the ANALYTICS program or some program in the ANALYTICS package. (See Part II of this manual, Introduction to ANALYTICS, for this purpose.)

There are other things you may want or need to do in connection with your course work for 80-100, among them: read your computer mail (if you have received any mail since your last logout, this will be announced when you login), send computer mail, delete your mail file, create or print some file for the course. (See Section 7 below.)

There is at least one thing you should do the first time you login besides run a course program: create a Login.Cmd (login command) file. (See Section 7 below.)

There is one thing you must do after you’re finished with every session on the computer: Logout. (See Section 6 below.)

6. Exiting the System: The Logout Procedure

When you are finished with your work on the system and you are ready to leave, you must terminate your job and exit the system by logging out. Until you logout your account will continue to be charged for the time connected to the system. And if you leave a terminal connected to the system on your account (without logging out), you leave your directory and account open to sabotage.

When you are ready to logout and leave your terminal, you need to have or get the system’s attention, signified by the '@' sign. If you’re in the middle of an ANALYTICS program, type STOP when given the option by the program. Type STOP again to exit ANALYTICS proper. This will return you to the system’s monitor and '@' sign. A CTRL-C pressed once or (if necessary) a couple of times will also always get the system’s attention. (But if you use CTRL-C to exit ANALYTICS, be sure you have created a Login.Cmd file containing the Delete Philo.Tmp command. See Section 7. Otherwise, be sure to exit ANALYTICS legally by use of the STOP command.)

When you have the system’s '@' sign, you may logout by simply typing the command

```
Logout <CR>
```

and hitting the 'Return' bar (as indicated by the CR reminder above). The system should then log you out, printing information about your job and what it cost. A successful logout might look like the following (where what you type is represented in boldface):

As you begin to accumulate files in your directory, one danger arises: you may exceed your permanent disk storage limit (10 pages for 80-100). If you exceed your limit, the system will not allow you to logout. Your only recourse at the moment would then be to delete some file(s) in order to bring your directory within its set limits. (See Section 7 below.) Then repeat the logout procedure above.

7. Creating A Login.Cmd File & Other System Options

There are many general-purpose things you will need and want to learn to do on the TOPS-20 system, like how to read and send MAIL, PRINT, DELETE and CREATE files, etc. The system functions and commands that allow you to do these and other useful things are explained in a two-volume manual published by the Communications Design Center and available in the bookstore, Introduction to the CMU TOPS-20. Get this manual, read it, and bring it with you to the terminal room.

One thing in particular that you should do for your convenience in using the ANALYTICS programs in 80-100 is to create a Login.Cmd file that contains the following command:

Delete Philo.Tmp

This command will delete the file called Philo.Tmp from your directory automatically when you login. (Login.Cmd files contain commands which are executed by the system when you log in.)

The Philo.Tmp file is a temporary file created by the ANALYTICS programs whenever you tell one of them what type of terminal you are using: they store this piece of information in the Philo.Tmp file so that when you run different programs they don’t have to keep asking you for your terminal type.

The Philo.Tmp file is automatically deleted when you exit ANALYTICS 'legally' using the STOP command, in case you should be on a different type terminal when you next run the programs.

Should you ever want to exit ANALYTICS 'illegally' by using CTRL-C, this file would not be
deleted. You would then be in danger of entering \textit{ANALYTICS} using a different terminal type while the program assumed that you were on the type last reported and recorded in the Philo.Tmp file.

To be sure that \textit{ANALYTICS} always knows or asks what type terminal you're using, it's convenient to have any past Philo.Tmp file deleted automatically upon logging in (since you may be logging in on a different terminal type from the one you used in your last session with \textit{ANALYTICS}).

So, check out your TOPS-20 manual and create a Login.Cmd file to delete Philo.Tmp each time you log in.

\textbf{Bon Voyage}
II. Introduction to ANALYTICS

1. How to Run ANALYTICS: Program Contents

The ANALYTICS program itself is just a table-of-contents or menu program that allows you easy selection among and access to the ten programs in formal logic and argument analysis designed for the Philosophic Methods and Social Values course, 80-100. 'ANALYTICS' is the name of a package of computer programs, each of which can be run separately and directly, without going through ANALYTICS.

Each of the ten programs contained in ANALYTICS can be run by using the following command schema:

```
RUN<PC0K>Programname <CR>
```

The easiest way to remember about the placement of spaces in typing this command sequence is simply to remember to insert no spaces. A run command like the above includes my directory name, PC0K, because the system must be told where/in what directory the program is to be found. In place of 'programname' you would of course type the actual name of the program you wanted to run. (Remember: ' CR ' is the reminder to hit the 'Return' bar upon entering a command or information that you want the system or a program to act upon.)

Until you know your way around the programs and the computer better, you're best advised to always run the ANALYTICS program itself and then choose among the programs contained within it according to the assignments in your 80-100 syllabus. To run the ANALYTICS program and gain access to its several contents, you simply type the following (when prompted by the '@' sign -- what you type is in boldface):

```
@RUN<PC0K>Analytics <CR>
```

The results of a successful execution of this run command is depicted on the next page.
@Run<PC0K>Analytics <CR>

The following programs are available:

1 - SENT : Sentential Logic Lessons
2 - SYMBOL : Symbolization Exercises -- Sentential Connectives
3 - SYMBO2 : Symbolization Exercises -- English Sentences
4 - TRUTH : Truth-value Analysis Exercises
5 - VALIDITY : Rules of Inference and Valid Argument Forms
6 - EQUIV : Logical Equivalence and Rules of Replacement
7 - QUANT : Quantificational Logic Lessons [OFF LINE]
8 - BERTIE : Proof Checker for Symbolic Logic
9 - RECON : Argument Reconstruction Program
10 - ARGUE : Free-form Argument Construction Program

Which would you care to do?

<1-10 or STOP> ?

At this point, where ANALYTICS is prompting me for a response/choice with its question mark ?, I could either elect to stop and exit ANALYTICS by typing the command 'STOP', or I could select one of the ten listed programs to run by typing the appropriate number 1, 2, 3, etc...or 10. Let's pick it up from ANALYTICS' question and see what would happen if I elected to stop (What I type is represented in boldface):

Which would you care to do?

<1-10 or STOP> ? STOP

RIGHTO! GOODBYE!

@  

Here ANALYTICS says good-bye and returns me to the system's '@' sign. (It also deletes any record of what terminal type I might have said I was using. This is why, when first using these programs, you should exit any program via ANALYTICS using the STOP command.)

Often, your options or choices within a program will be presented in abbreviated form between angle brackets < >, like the options to choose programs 1, 2, 3, etc...or 10 or STOP in the example above: <1-10 or STOP>. Your options of this sort will either be explained or clear to commonsense in the context of the programs.
2. Instructional Programs: SENT, VALIDITY, EQUIV, QUANT

These programs each consist of instructional material, like that in a textbook, organized into 'frames' of discrete information upon which you are questioned at regular intervals. Unlike a textbook, the programs require you to respond to the material and they respond to you in turn: they require you to be actively responsive and they are interactive. Thus they are presumably more effective media for learning than textbook reading.

At the end of each lesson (which take 10 to 15 minutes to complete) you are given choices regarding what to do next. Follow your syllabus. You are given the option to exit these programs only at the end of each lesson. The instructions needed to negotiate the lessons are contained therein.

In the sample 'tour' provided into the programs below (Section 5), the table of contents or lessons for the SENT and VALIDITY programs are provided. (We will not be using the QUANT or EQUIV programs.)

This brief 'tour' also illustrates how the programs are all looped together in ANALYTICS.

3. The Drill-and-Practice Programs: SYMBOL, SYMBO2, TRUTH

These programs are subdivided into lesson types of graduated orders of difficulty. You choose the lesson/level assigned in your syllabus and do as many problems as you need to feel mastery of the exercise (five or more in each lesson/level).

These programs generate their own problems (the problems are not stored or 'hard-coded' in the programs, but created by the programs); and the programs know the correct answer in each case and will correct you accordingly. Read the instructions and note the commands available for these programs (Sections 5.2 through 5.4 below -- the instructions are also available within the programs themselves by use of the INS command).

NOTE: Because these programs generate exercises and their answers ad infinitum, they will keep giving you exercises until you tell them to stop by using the STOP command (in response to their prompt 'What next?'), or until you tell them to start another lesson by using the START command (in response to 'What next?').

NOTE: You can exit or stop these programs or go back to the 'start' to choose a new lesson/level only at the points where the programs ask you 'What next?' Once you have begun a single exercise, you must and should finish it. Be aware that help or hints are available, and that the program will guide you through the exercise or give you the answer if you fail after a couple tries.
4. Argument Analysis / Proof-Checking Programs: BERTIE, RECON, ARGUE

These problem-solving guidance programs have separate chapters of this manual devoted to them and will not be used until later in the course. You will be given the relevant parts of the manual at that time.

5. A Brief Tour With Sample Interactions:

SENT, SYMBOL, SYMBO2, TRUTH, VALIDITY

The following sections contain a guided tour of the programs you will be using in your initial assignments for 80-100. The tour will follow the order in which you are introduced to the programs in your syllabus. It will include a look at the lesson content of each program and, for the drill-and-practice programs, a listing of the instructions and commands.

We will take this tour by running and looping through ANALYTICS, so you can see how all the programs are connected through this menu program. The tour will be continual, from entry into to exiting from ANALYTICS, but I will highlight the segments within each program with numbered subsections as follows:

5.1 The SENT Program
5.2 The SYMBOL Program
5.3 The SYMBO2 Program
5.4 The TRUTH Program
5.5 The VALIDITY Program

We will begin by running ANALYTICS. What you would type in the following interactions will be printed in boldface. What the system or programs print will be in normal typeface. In running ANALYTICS I will first select the SENT program, selection 1 in the listing below.
5.1 The **SENT** Program -- via ANALYTICS

@Run<PC0K> ANALYTICS

The following programs are available:

1 - SENT : Sentential Logic Lessons
2 - SYMBOL : Symbolization Exercises -- Sentential Connectives
3 - SYMB02 : Symbolization Exercises -- English Sentences
4 - TRUTH : Truth-value Analysis Exercises
5 - VALIDITY : Rules of Inference and Valid Argument Forms
6 - EQUIV : Logical Equivalence and Rules of Replacement
7 - QUANT : Quantification Logic Lessons
8 - BERTIE : Logic Proof Checker
9 - RECON : Argument Reconstruction Program
10 - ARGUE : Free-form Argument Construction Program

Which would you care to do?

<1-10 OR STOP> ? 1

What type of terminal are you on?

1 - Hard copy
2 - Perkin-Elmer 1100
3 - Perkin-Elmer 550
4 - VT52 Decscope

? 1

Do you want an introduction <Y or N> ? N

The following [SENT] tutorials and drills are available:

0 - Introduction
1 - Negation
2 - Review of negation
3 - Conjunction
4 - Shorthand truth-value analysis
5 - Review of disjunction
6 - Disjunction 1: The ambiguity of disjunction
7 - Disjunction 2: The rule of disjunction
8 - Conditional 1: What conditionals mean
9 - Conditional 2: The rule for 'IF'
10 - Contradiction

Which would you enjoy doing?

Enter the number <0 - 10 or STOP> ? STOP

The following [ANALYTICS] programs are available:

1 - SENT : Sentential Logic Lessons
2 - SYMBOL : Symbolization Exercises -- Sentential Connectives
3 - SYMB02 : Symbolization Exercises -- English Sentences
4 - TRUTH : Truth-value Analysis Exercises
5 - VALIDITY : Rules of Inference and Valid Argument Forms
6 - EQUIV : Logical Equivalence and Rules of Replacement
Which would you care to do?

<1-10 OR STOP> 2

5.2 The SYMBOL Program

Note: To do the sample exercise shown below you must have done SENT lessons 1, 2 and 4!

SYMBOL: SYMBOLIZING SENTENTIAL CONNECTIVES

Do you want instructions <Y or N>  Y

These drills are designed to give you practice translating English connectives into symbolic form.
You will be given sentence forms with sentential connectives in ordinary English and asked to
1) Identify the logical forms

- NEGATION - N
- CONJUNCTION - CJ
- DISJUNCTION - D
- CONDITIONAL - CD
- BICONDITIONAL - B

2) Type the symbolic forms.

The symbols for the connectives are:

V - DISJUNCTION
& - CONJUNCTION
=> - CONDITIONAL
<=> - BICONDITIONAL
- - NEGATION

Hit RETURN to continue.

The following commands are available:

'RETURN' - go to next problem when in response to 'What next?'
END or STOP - exit program in response to 'What next?'
INS - get program instructions in response to 'What next?'
START - choose another lesson in response to 'What next?'
HELP - when in response to 'What next?' will give this list of commands.

Hit RETURN to continue.

The following [SYMBOL] lessons are available:

1 - Conjunction
Enter lesson number or stop <1-8 or STOP> ? 1

OK, let's give it a go.

It is not the case that (not (it is not the case that P, however I) and G).
What form is this sentence? CONJUNCTION

No, try again.
? NEGATION

Right! Good!

SYMBOLIZATION:
? -(--P & I) & G

Not quite. Try again.
? -(--P & I) & G

Wrong again. It's

-(-(-P & I) & G)

What next? START

The following [SYMBOL] lessons are available:

1 - Conjunction
2 - Disjunction
3 - Conjunction plus Disjunction
4 - Conditional
5 - Conditional plus Conjunction/Disjunction
6 - Biconditional
7 - Mixed Bag: Random Problems
8 - Free Form: Construct Your Own Problems

Enter lesson number or stop <1-8 or STOP> ? STOP

The following [ANALYTICS] programs are available:

1 - SENT: Sentential Logic Lessons
2 - SYMBOL: Symbolization Exercises -- Sentential Connectives
3 - SYMBO2: Symbolization Exercises -- English Sentences
4 - TRUTH: Truth-value Analysis Exercises
5 - VALIDITY: Rules of Inference and Valid Argument Forms
6 - EQUIV: Logical Equivalence and Rules of Replacement
7 - QUANT: Quantificational Logic Lessons
8 - BERTIE: Logic Proof Checker
9 - RECON: Argument Reconstruction Program
10 - ARGUE: Free-form Argument Construction Program
Which would you care to do?
<1-10 OR STOP> ? 3

5.3 The SYMBO2 Program

SYMBO2: SYMBOLIZING ENGLISH SENTENCES

Do you want instructions <Y or N> ? Y

These drills are designed to give you practice translating English sentences into symbolic form.

You will be given molecular sentences with sentential variables assigned to each component atomic sentence. You will then be asked to symbolize the sentence using the assigned sentential variables and our conventional symbols for the sentential connectives.

The symbols for the connectives are:

- V - DISJUNCTION
- & - CONJUNCTION
- => - CONDITIONAL
- <=> - BICONDITIONAL
- - NEGATION

Hit RETURN to continue.

The following commands are available:

'RETURN' - go to next problem when in response to 'What next?'
END or STOP - exit program in response to 'What next?'
INS - get program instructions in response to 'What next?'
START - choose another lesson in response to 'What next?'
HELP - When in response to 'What next?' will give this list of commands.

Hit RETURN to continue.

The following [SYMBO2] lessons are available:

1 - Conjunction
2 - Disjunction
3 - Conjunction plus Disjunction
4 - Conditional
5 - Conditional plus Conjunction/Disjunction
6 - Biconditional
7 - Mixed Bag: Random Problems
8 - Free Form: Construct Your Own Problems

Enter lesson number or stop <1-8 or STOP> ? 1

OK, let's give it a go.
G - the life of a cancerous cell is sacred
H - living things automatically possess a right to life
I - persons have a special right to life
J - people have rights

It's not the case that although the life of a cancerous cell is not sacred, living things do not automatically possess a right to life; and people don't have rights.

SYMBOLIZATION:
? -G & H & -J

Not quite. Try again.
? -(G & H) & J

Wrong again. It's

-(G & -H) & -J

What next? <CR>

G - human fetuses are persons
H - people have rights
I - all forms of life are deserving of rights
J - life per se is sacred

Life per se is sacred and all forms of life are deserving of rights.

SYMBOLIZATION:
? START

Not quite. Try again.
? STOP

Wrong again. It's

J & I

What next? START

The following [SYMBO2] lessons are available:

1 - Conjunction
2 - Disjunction
3 - Conjunction plus Disjunction
4 - Conditional
5 - Conditional plus Conjunction/Disjunction
6 - Biconditional
7 - Mixed Bag: Random Problems
8 - Free Form: Construct Your Own Problems

Enter lesson number or stop <1-8 or STOP> ? STOP

The following [ANALYTICS] programs are available:

1 - SENT : Sentential Logic Lessons
Which would you care to do?

<1-10 OR STOP> ? 4

5.4 The TRUTH Program

NOTE: The sample exercise below requires having done SENT lessons 1, 2 and 4!

TRUTH: TRUTH-FUNCTIONAL ANALYSIS OF CONNECTIVES

Do you want instructions <Y or N> ? Y

These drills are designed to give you practice translating English connectives into symbolic form and in determining the truth-value of sentences according to the truth-functional rules governing the sentential connectives. You will be given a list of sentential variables and their truth-values. You will then be given sentence forms with sentential connectives in ordinary English. Based on the assigned truth-values and the logical form of the sentence, you will be asked to determine the truth-value of the sentence.

If you fail to give the correct answer, you will be asked to work through the 'short-hand' truth table for the sentence. If you give the correct answer, the truth table exercise is optional.

If you need a hint or aid when the program asks you a question, like '<T or F>?' or 'What next?' type HELP.

Hit RETURN to continue.

The following commands are available:

'RETURN' - go to next problem when in response to 'What next?'
END or STOP - exit program in response to 'What next?'
INS - get program instructions in response to 'What next?'
START - choose another lesson in response to 'What next?'
TT - get truth table of the connectives in response to 'What next?'
HELP - when in response to 'What next?' will give this list of commands.
- when in response to the question '<T or F>?' will get either the SYMBOLIC TRANSLATION or TRUTH-TABLE RULES.

Hit RETURN to continue.
The following [TRUTH] lessons are available:

1 - Conjunction
2 - Disjunction
3 - Conjunction plus Disjunction
4 - Conditional
5 - Conditional plus Conjunction/Disjunction
6 - Biconditional
7 - Mixed Bag: Random Problems
8 - Free Form: Construct Your Own Problems

Enter lesson number or stop <1-8 or STOP> ? 1

OK, let's give it a go.

Variables and values:

G  F
H  F
I  T
J  T

J but (G, nevertheless J).

T or F ? HELP

Symbolic form or truth tables <S or T> ? T

P ! Q ! -P ! P&Q ! PVQ ! P=>Q ! P<=Q

T T F T T T T
T F F T F F F
F T F T T T T
F F T F T T T

Hit RETURN to continue.

T or F ? HELP

Symbolic form or truth tables <S or T> ? S

J & (G & J)

Hit RETURN to continue.

T or F ? F

Correct! Or lucky!

Do you want to work through truth table <Y or N> ? Y

J & (G & J)
T  F  T
 ? F

? F
Do you want to see the truth-table <Y or N> ? Y

J & (G & J)
T   F   T
F
F

Table is complete and correct.
What next ? START

The following [TRUTH] lessons are available:

1 - Conjunction
2 - Disjunction
3 - Conjunction plus Disjunction
4 - Conditional
5 - Conditional plus Conjunction/Disjunction
6 - Biconditional
7 - Mixed Bag: Random Problems
8 - Free Form: Construct Your Own Problems

Enter lesson number or stop <1-8 or STOP> ? STOP

The following [ANALYTICS] programs are available:

1 - SENT : Sentential Logic Lessons
2 - SYMBOL : Symbolization Exercises -- Sentential Connectives
3 - SYMBO2 : Symbolization Exercises -- English Sentences
4 - TRUTH : Truth-value Analysis Exercises
5 - VALIDITY : Rules of Inference and Valid Argument Forms
6 - EQUIV : Logical Equivalence and Rules of Replacement
7 - QUANT : Quantificational Logic Lessons
8 - BERTIE : Logic Proof Checker
9 - RECON : Argument Reconstruction Program
10 - ARGUE : Free-form Argument Construction Program

Which would you care to do?

<1-10 OR STOP> ? 5

5.5 The VALIDITY Program

VALIDITY: Rules of Inference and Valid Argument Forms

Do you want an introduction to this lesson set?
<y or N> ? N

The following lessons explaining the validity of inference rules are available:

0 - Introduction
1 - The CONJUNCTION rule
2 - The SIMPLIFICATION rule
3 - The ADDITION rule
4 - The DISJUNCTIVE SYLLOGISM rule --
   Validity Test Explained
5 - The MODUS PONENS and MODUS TOLLENS rules -- CONDITIONALS Explained
6 - The CONSTRUCTIVE DILEMMA rule
7 - The HYPOTHETICAL SYLLOGISM rule

Type the number of the rule you would like to prove valid and have explained.

<0 - 7 or STOP> ? STOP

The following [ANALYTICS] programs are available:

1 - SENT : Sentential Logic Lessons
2 - SYMBOL : Symbolization Exercises -- Sentential Connectives
3 - SYMBO2 : Symbolization Exercises -- English Sentences
4 - TRUTH : Truth-value Analysis Exercises
5 - VALIDITY : Rules of Inference and Valid Argument Forms
6 - EQUIV : Logical Equivalence and Rules of Replacement
7 - QUANT : Quantificational Logic Lessons
8 - BERTIE : Logic Proof Checker
9 - RECON : Argument Reconstruction Program
10 - ARGUE : Free-form Argument Construction Program

Which would you care to do?
<1-10 OR STOP> ? STOP

RIGHTO! GOOD-BYE!
III. The BERTIE Program

1. BERTIE's Functions

BERTIE is designed to give you practice in constructing derivations in symbolic notation. Practice in doing derivations with BERTIE will exercise your sense of logical form and reinforce your judgments about validity within the very precise constraints of symbolic logic. BERTIE will check each line of any derivation you construct for the following crucial logical properties:

1. **Well-Formedness**: BERTIE will check any symbolic formula you enter in a derivation to see that it is a well-formed formula (WFF). It will give you an error message when any formula is not well formed.

2. **Validity**: BERTIE will check any line you enter in a derivation to see that it is a valid move according to our given rules of inference and replacement. (See Appendix II for summary of given rules.) When you enter any line in a derivation, you must justify it by citing the previous line(s) of the derivation and/or the rule that allows you to derive or enter the new line. BERTIE will check your justification to see:

   - Whether you have cited correct or sufficient line numbers to justify the new line.
   - Whether you have cited the correct rule to justify the new line.

If you fail to justify any line in a derivation correctly, or if the line you enter is not allowed or valid by the cited rule, BERTIE will give you an error message. BERTIE thus prompts you to detect and correct your errors immediately. BERTIE allows you three different options for practicing derivations:

1. **'Request a Problem'**: You can work on stored problems from BERTIE's problem sets. These problems are supplied with stored hints and solutions that you can see by using the HINT, HELP, and UNCLE commands. (For a description of BERTIE's problem sets, see Appendix I. For a description of BERTIE's various commands, see Section 3 below. See Section 2.1 below for illustration of how to work with BERTIE's stored problems.) When you have successfully derived the conclusion assigned in a given problem, BERTIE will congratulate you (as illustrated in Section 2.1 below).

2. **'Enter a Conclusion to Derive'**: You can enter your own conclusion (in symbolic notation) to derive, and thus create your own derivation problems to work on. When you have successfully derived your stated conclusion in valid fashion, BERTIE will congratulate you (as illustrated in Section 2.2 below).

3. **'Or Type 'Begin'**: You can simply begin a derivation without specifying any target conclusion that you want to derive. BERTIE will allow you to proceed as you will, checking each step for validity. In this 'free-form' mode BERTIE doesn't know what in particular you want to derive, because you have not specified any particular conclusion; so it will not congratulate you at any point in your derivation, but will simply prompt you for successive steps in your derivation until you request a stored
problem, ask to start another line of derivation or stop. Section 2.3 below illustrates this 'free-form' option in BERTIE.

These three options are best illustrated with some sample interactions. Let's take a look at what it's like to run and work with BERTIE.
2. Running the BERTIE Program: Options & Key Commands

You can run the BERTIE program either by running the ANALYTICS menu program and choosing option 8 BERTIE, or by running BERTIE directly, as follows. When you have the system's attention (the @ sign), what you type is represented in boldface below. The program's response is in normal typeface (CR indicates the required carriage return):

@Run<PCOK>BERTIE <CR>

IF YOU NEED INSTRUCTIONS, TYPE 'INS'; OTHERWISE
ENTER A CONCLUSION TO DERIVE, REQUEST A PROBLEM, OR TYPE 'BEGIN'
? INS <CR>

BERTIE IS A COMPUTER PROGRAM THAT CHECKS DERIVATIONS IN SENTENTIAL AND QUANTIFICATIONAL LOGIC.

BERTIE ACCEPTS THE FOLLOWING COMMANDS:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SET N</td>
<td>OPENS PROBLEM SET N</td>
</tr>
<tr>
<td>PROBLEM N</td>
<td>ASKS BERTIE TO GIVE YOU PROBLEM NUMBER N</td>
</tr>
<tr>
<td>HINT</td>
<td>REQUESTS A HINT ABOUT SOLVING THE PROBLEM</td>
</tr>
<tr>
<td>HELP N</td>
<td>PRINTS N (MORE) LINES OF BERTIE'S STORED SOLUTION</td>
</tr>
<tr>
<td>UNCLE</td>
<td>GIVE UP? BERTIE PRINTS A SOLUTION</td>
</tr>
<tr>
<td>NEW</td>
<td>ALLOWS YOU TO BEGIN ANOTHER PROBLEM OR DERIVATION</td>
</tr>
<tr>
<td>LIST</td>
<td>LISTS YOUR DERIVATION (TO GET A CLEAN COPY)</td>
</tr>
<tr>
<td>LIST M-N</td>
<td>LISTS LINES M THROUGH N OF YOUR DERIVATION</td>
</tr>
<tr>
<td>DELETE N</td>
<td>ERASES YOUR DERIVATION FROM LINE N TO THE LAST LINE</td>
</tr>
<tr>
<td>REMOVE N</td>
<td>ERASES LINE N ONLY (DERIVATION IS RENUMBERED; USE LIST)</td>
</tr>
<tr>
<td>RULES</td>
<td>PRINTS RULES OF INFERENCE AND REPLACEMENT</td>
</tr>
<tr>
<td>STOP / BYE</td>
<td>STOPS THE PROGRAM (EXITS THROUGH ANALYTICS)</td>
</tr>
<tr>
<td>INS</td>
<td>PRINTS THIS LIST OF COMMANDS</td>
</tr>
<tr>
<td>/</td>
<td>TYPE THE SLASH '/' TO JUSTIFY EACH LINE IN YOUR DERIVATION FOLLOWED BY THE RULE AND LINE NUMBERS APPEALED TO</td>
</tr>
</tbody>
</table>

ENTER A CONCLUSION TO DERIVE, REQUEST A PROBLEM, OR TYPE 'BEGIN'
?

The interaction above illustrates (1) the BERTIE program's opening response, (2) the program's prompt, the question mark ?, in response to which you would type a command (or a line in a derivation), (3) the commands available in BERTIE (printed in response to the INS command), and (4) the three options allowed you for working in BERTIE: 'Enter a conclusion to derive, request a problem, or type 'Begin'. I will illustrate each of these options in the three sections that follow, and highlight the commands you should be familiar with at each stage. A description of the commands will be found in Section 3 below. I will begin with the middle option, requesting a problem, since it is the one you are most apt to elect in your initial work with BERTIE.
2.1 '... Request a Problem ...

When the BERTIE program prompts you to select one of its three options for practicing derivations and you wish to work on some stored problem(s) (which are assigned in the course syllabus and described in Appendix I), you need first to tell BERTIE which of its problem sets to open by using the SET command.

2.1.1 Open a Problem Set: The SET and PROblem Commands

In order to work any problem you must first ask BERTIE to open a problem set. The available problem sets are identified by number (1, 2, 3... 5). (See Appendix I for a description of available problem sets. These are assigned in your syllabus, but may be reviewed and reworked any time you want.)

You request a problem set N in response to BERTIE's prompt, ?, by typing the command SET N, where in place of the N you would type the number of the set you want. To get BERTIE to open problem set 1, you would type, in response to the prompt ?: SET 1. (You may type any commands or other input to BERTIE in either lower or upper case, in any combination.)

BERTIE will tell you how many problems are contained in the requested set and then again give you its three basic options: 'Enter a conclusion to derive, request a problem, or type 'Begin'. After opening a given set N, you may pursue any of these options and whenever you request a problem by number, say, problem M, BERTIE will give you problem M from set N. Whenever you request a problem, BERTIE will assume that you want a problem from the problem set last opened. (You can open a different problem set by using the set command at any time that BERTIE has prompted you with the question mark ?, even in the middle of doing a given problem.)

To request a given problem, once you have opened a problem set, you use the Problem command. Notice (in the illustration that follows) that the Problem command can be abbreviated to Pro. For example, to request the first problem in a set you could type, in response to BERTIE's prompt '?' : either Problem 1, or Prob 1, or Pro 1. In general, BERTIE's commands (and rule names) can be abbreviated to their first three or four letters.

In the illustrative interaction that follows, I will run BERTIE, request a problem (without first opening a problem set, to show you how BERTIE responds), then open a problem set, request a problem, and (in the middle of the problem) request that BERTIE open a different problem set and give me a problem from that set -- all to show you how easily you can flip in and out of different problem sets with the SET command. (Again, what I type -- or what you would type
-- is represented in boldface. BERTIE's responses are in normal typeface. Commentary which is not part of the in-put/out-put will be between square brackets ... . I will often dispense with the <CR> to remind you to hit the Return bar after typing your in-put in the illustrations provided below.)

@Run <PC0K>BERTIE <CR>

IF YOU NEED INSTRUCTIONS, TYPE 'INS'; OTHERWISE

ENTER A CONCLUSION TO DERIVE, REQUEST A PROBLEM, OR TYPE 'BEGIN'
? Pro 4 <CR>

YOU MUST FIRST OPEN A PROBLEM SET WITH THE 'SET' COMMAND

ENTER A CONCLUSION TO derive, REQUEST A PROBLEM, OR TYPE 'BEGIN'
? Set 2

THERE ARE 13 PROBLEMS IN THIS SET

ENTER A CONCLUSION TO DERIVE, REQUEST A PROBLEM, OR TYPE 'BEGIN'
? Pro 4

DERIVE: -G

1  F => -G / PREMISE
2  -L => -F / PREMISE
3  G => F / PREMISE
4  -L / PREMISE
5  ? Set 3 <CR>

[Notice that after setting up the problem by giving me a conclusion to derive and four premises from which to derive it, BERTIE then prompts me for in-put with its question mark '?' at line 5.

At this point I have elected not to do the problem but rather to open Problem Set 3. BERTIE then responds as follows:]

THERE ARE 15 PROBLEMS IN THIS SET

ENTER A CONCLUSION TO DERIVE, REQUEST A PROBLEM, OR TYPE 'BEGIN'
? Pro 2 <CR>

DERIVE: -(L & K)

1  -L V -K / PREMISE
2  ? Set 1 <CR>

THERE ARE 26 PROBLEMS IN THIS SET

ENTER A CONCLUSION TO DERIVE, REQUEST A PROBLEM, OR TYPE 'BEGIN'
? Pro 3 <CR>

DERIVE: Q & P
2.1.2 Working a Problem: Key Commands

The illustration above shows you that you can request a new problem or problem set at any point while working a given problem where BERTIE has prompted you for input with the '?'.

The illustrations below show you the effects of certain key commands in the course of actually working through a problem.

The Slash Command '/': Justifying the Lines of a Derivation

Whenever you enter a symbolic formula on a line of a derivation, you must give a justification for that line by citing the rule that warrants the line and the numbers of any previous line(s) to which you appeal. You indicate to BERTIE that you are going to give a justification for a line of derivation that you have just entered by typing the slash '/' after entering your formula followed by the appropriate rule and line number(s). For example, given the problem below, you could proceed as follows (where what you type is in boldface):

DERIVE: P & R

1 P & Q / PREMISE
2 ?

3 ? P / SIMPL 1
4 ? R / 2 SIMPR
5 ? P&R/3 4CONJ

CONGRATULATIONS! YOUR DERIVATION IS COMPLETE.

LIST YOUR DERIVATION, ASK FOR A PROBLEM, OR TYPE 'NEW'.

When you have succeeded in deriving the assigned conclusion, BERTIE will congratulate you (as above) and give you the options of getting a clean listing of your derivation, requesting another problem, or starting anew. Typing the NEW command at this juncture, or at any point where BERTIE prompts you with its query '?', will simply get you BERTIE's original three basic options: **Enter a conclusion to derive, request a problem, or type 'Begin'.** More on LISTing your derivation later. For now you should take note of the following:
1. Whenever you enter a formula, like P, on a line of derivation, like line 3 above, you must also enter a justification preceded by the slash / on that line.

2. The justification of a line of derivation, following the slash /, consists of a rule citation and, where previous lines are appealed to, line numbers.

3. Some rules, like the PREMISE introduction rule, do not require the citation of previous lines because they allow the introduction of lines that do not depend logically on previous lines. Notice that in introducing the two premises in the problem above, BERTIE cited only the PREMISE rule preceded by the slash /.

4. You may cite the rule either before the line numbers or vice versa (compare lines 3 and 4 above) -- the order does not matter.

5. You need not put any spaces between any of the entries in a line of derivation except between line numbers (see line 5 above) . . .

6. You must put either a space or a comma between line numbers when more than one line number is cited in a justification: For example, on line 5 above, had we typed 34 instead of 3, 4, BERTIE would have thought we were referring to line 34 rather than lines 3 and 4 (and would have given us an error message to the effect that the line number was too large and too few lines were appealed to for the CONJunction rule).

7. It does not matter how many spaces you put between entries in a line of derivation, so long as you put no spaces in the name or abbreviation of the cited rule.

8. You may use abbreviations for the rules of inference and replacement. (The first three or four letters of the rule name or the initials of the rule will do: legal abbreviations are given on your rule summaries and in BERTIE's rule file, which you can have BERTIE print out by typing RULES in response to the query '?'.)

9. You may type your input to BERTIE in any combination of upper and lower case.

10. You must put the slash / between the formula you enter as a line of derivation and its justification -- unless you use the slash command to have BERTIE type the line for you (see the next section below).

The Slash Command '/': Commanding BERTIE to Type a Line of Derivation

It is often possible to get BERTIE to type a desired line of a derivation for you (and thereby save yourself both effort and the risk of making a typographical error) by using the slash / as a command. When prompted by BERTIE's query ? to enter a line in a derivation, you might type the slash / plus the rule you want to apply plus the line numbers you want the rule
applied to. Then hit the carriage return. This will have the effect of telling BERTIE: 'Give me what I can get by the cited rule from the cited lines.' If, and only if, there is a valid consequence that can be directly derived (in one step) from the cited lines by the rule you cite, then BERTIE will print out that result for you with its justification. If there is no valid consequence that can be obtained by the cited rule from the cited lines in one step, BERTIE will of course give you an appropriate error message to that effect. Were we to use the slash command in this way on the problem in the previous illustration, the results would look like this (where what we type is printed in boldface):

**DERIVE: P & R**

1. **P & Q** / PREMISE
2. **S & R** / PREMISE
3. ? / 1 SIMPL
4. P / 1 SIMPL
5. R / 2 SIMPR
6. ? / 3 4 CONJ
7. P & R / 3 4 CONJ

**CONGRATULATIONS! YOUR DERIVATION IS COMPLETE**

**ENTER A CONCLUSION TO DERIVE, REQUEST A PROBLEM, OR TYPE 'BEGIN' ?**

When the formulae that you want to derive are longer and more complex than the above, this feature of BERTIE's slash command is especially helpful. In using this convenience, however, you should note carefully the following:

1. The slash command cannot be used with all rules to get BERTIE to print the line you want. It can only be used when the result of applying the cited rule to the cited lines would be unique or unambiguous. For example, were you to type / PREMISE in response to BERTIE's query ?, BERTIE could not print out a premise for you because it could not possibly know which of the infinite possible premises you might want.

2. In certain cases where the application of a rule to certain lines might allow more than one valid result, the ambiguity can be resolved by specifying an appropriate rule variation. Certain rules (like SIMPlification) that warrant some basic logical move (like deriving one conjunct from a conjunction) but that could have more than one result (like the derivation of either the left or the right conjunct), have different variations to get different results. Look at lines 3 and 4 in the illustration above: the rule specification SIMPL gave us the left conjunct from line 1; the rule specification SIMPR gave us the right conjunct from line 2. **The SIMPlification**
rule thus has two specifications to allow unambiguous application of the slash command. The case is similar with DSL and DSR, CDI and CDII (see your rule summary, in your logic text or Appendix II to the BERTIE manual).

3. In cases where a replacement rule could be applied to more than one connective in a formula, BERTIE will always apply the rule to the major connective. Thus, for example, if you use the slash command to apply the COMmutation rule to the formula $P \lor (Q \land R)$, BERTIE will commute the disjunctive part of the formula (since the '$\lor$' is the major connective) as follows: $(Q \land R) \lor P$.

4. When using the slash command to get BERTIE to type formulae for you and when there is a choice about the order in which the components (say, the conjuncts in a conjunction) occur, you should type the line numbers of the components (conjuncts) in the order in which you want them combined (conjoined). Look at line 5 in the illustration above. I typed / 3 4 CONJ and BERTIE formed the resultant conjunction with the conjunct from line 3 first and the conjunct from line 4 second. Had I typed / 4 3 CONJ BERTIE would have formed the conjunction in the reverse order: $R \land P$ rather than $P \land R$.

Information on the other useful BERTIE commands, with examples of their applications, is contained in the HINTs in the first problems in BERTIE Set 1. Use the HINT command in these initial problems to read this information, even if you do not feel the need for hints. In this way you will get hands-on experience with the commands and functions of BERTIE while doing actual problems. The remainder of this BERTIE manual will be handed out in class when you have need of it. What you have is all you need, with the HINTs in BERTIE Set 1, to work the initial problems assigned. BERTIE's rule set follows, in Appendix II.
Appendix II

Rules of Inference and Replacement

CONTENTS

SENTENTIAL RULES OF INFERENCE

SENTENTIAL RULES OF REPLACEMENT

CONDITIONAL PROOF RULE & INDIRECT PROOF STRATEGY

QUANTIFICATIONAL RULE SCHEMAS

RESTRICTIONS ON THE USE OF UNIVERSAL GENERALIZATION

RESTRICTIONS ON THE USE OF EXISTENTIAL INSTANTIATION
SENTENTIAL LOGIC: RULES OF INFERENCE

Premise (PREM or P)
A premise may be introduced on any line of a derivation, except within Conditional Proof.

Excluded-Middle Introduction (E-MI or EMI): \( P \lor \neg P \)
At any point in a derivation one may introduce a sentence of the above form.

Conjunction (CONJ)

- \( P \)
- \( Q \)

\( P \land Q \)

Simplification (SIMPL)

- \( P \land Q \)

\( P \)

Disjunctive Syllogism (DSL)

- \( P \lor Q \)
- \( \neg Q \)

\( P \)

Modus Ponens (MP)

- \( P = Q \)
- \( P \)

\( Q \)

Constructive Dilemma (CDI)

- \( P \lor Q \)
- \( P = R \)
- \( Q = R \)

\( R \)

Hypothetical Syllogism (HS)

- \( P = Q \)
- \( Q = R \)

\( P = R \)

Addition (ADD)

- \( P \)

- \( P \lor Q \)

Simplification (SIMPR)

- \( P \land Q \)

- \( Q \)

Disjunctive Syllogism (DSR)

- \( P \lor Q \)
- \( \neg P \)

- \( Q \)

Modus Tollens (MT)

- \( P = Q \)
- \( \neg Q \)

- \( \neg P \)

Constructive Dilemma (CDII)

- \( P \lor Q \)
- \( P = R \)
- \( Q = S \)

\( R \lor S \)

Reductio Ad Absurdum (RED)

- \( P = (Q \land \neg Q) \)

- \( \neg P \)
Note: The dual colon `::` indicates that the respective formulae are logically equivalent -- i.e., that either formula may be replaced by the other in any line of a derivation, OR that either may be derived from the other in a derivation.

<table>
<thead>
<tr>
<th>Rule Type</th>
<th>Formula 1</th>
<th>Formula 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double Negation (DN):</td>
<td>P :: --P</td>
<td></td>
</tr>
<tr>
<td>Commutation (COM):</td>
<td>P &amp; Q :: Q &amp; P</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P V Q :: Q V P</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P = Q :: Q = P</td>
<td></td>
</tr>
<tr>
<td>Transposition (TRANS):</td>
<td>P = Q :: -Q = -P</td>
<td></td>
</tr>
<tr>
<td>Equivalence (EQUIV):</td>
<td>P = Q :: (P = Q) &amp; (Q = P)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P = Q :: (P &amp; Q) V (-P &amp; -Q)</td>
<td></td>
</tr>
<tr>
<td>Implication (IMPL):</td>
<td>P = Q :: -P V Q</td>
<td></td>
</tr>
<tr>
<td>De Morgan (DEM):</td>
<td>-P &amp; -Q :: -(P V Q)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>-P V -Q :: -(P &amp; Q)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P &amp; Q :: -(-P V -Q)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P V Q :: -(-P &amp; -Q)</td>
<td></td>
</tr>
<tr>
<td>Exportation (EXP):</td>
<td>(P &amp; Q) = R :: P = (Q = R)</td>
<td></td>
</tr>
<tr>
<td>Tautology (TAUT):</td>
<td>P :: P V P</td>
<td></td>
</tr>
<tr>
<td></td>
<td>P :: P &amp; P</td>
<td></td>
</tr>
</tbody>
</table>
CONDITIONAL PROOF RULE & INDIRECT PROOF STRATEGY

CONDITIONAL PROOF STRATEGY: HYPOTHESIS + CP

With conditional proof your aim is to derive a conditional, say: P \Rightarrow Q

1. Assume the ANTECEDENT of the conditional as a HYPOTHESIS.

[Be sure to enter any premises BEFORE you enter your hypothesis!]

2. Derive the CONSEQUENT of the conditional you wish to derive.

3. DISCHARGE your hypothesis by conditionalization:

   Form the desired conditional (P \Rightarrow Q), citing the CP rule and the line of your hypothesis (P) through the line of the consequent (Q):

   \[ \text{THUS: } 1 \quad P \quad / \quad \text{HYPOTHESIS} \hspace{1cm} 2 \quad Q \quad / \quad \text{[Cite rules/lines by which derived]} \hspace{1cm} 3 \quad P \Rightarrow Q \quad / \quad \text{CP 1-2} \quad \text{[Note use of dash '-']} \]

INDIRECT PROOF STRATEGY: CP + REDUCTIO

Where you wish to derive some sentence, say: P

1. Assume as a HYPOTHESIS the NEGATION of the sentence to be derived:

   \[ 1 \quad -P \quad / \quad \text{HYPOTHESIS} \]

2. Derive a CONTRADICTION:

   \[ 2 \quad Q \, \& \, -Q \quad / \quad \text{[Rule/line cit.]} \]

3. Apply CP:

   \[ 3 \quad -P \Rightarrow (Q \, \& \, -Q) \quad / \quad \text{CP 1-2} \]

4. Apply the REDUCTIO rule: 4 \quad P \quad / \quad \text{REDUCTIO, 3}

   \[ \text{THUS: } 1 \quad -P \quad / \quad \text{HYPOTHESIS} \hspace{1cm} 2 \quad Q \, \& \, -Q \quad / \quad \text{[Cite rules/lines by which derived]} \hspace{1cm} 3 \quad -P \Rightarrow (Q \, \& \, -Q) \quad / \quad \text{CP 1-2} \hspace{1cm} 4 \quad P \quad / \quad \text{REDUCTIO, 3} \]
QUANTIFICATIONAL RULE SCHEMA SUMMARY

**UI** To constants: (x)Px   (x)(Px = Qx)  
---                     -----------------
  Pa                     Pa = Qa
or pseudo-names: (x)Px   (x)(Px = Qx)  
---                     -----------------
  Pt                     Pt = Qt

**UG** ONLY from pseudo-names NOT occurring in lines obtained by El:  
---
  Pt                     Pt = Qt
--------------------
  (x)Px                 (x)(Px = Qx)

**QN**
(x)Px :: -(Ex)-Px    (x)(Px & Qx) :: -(Ex)-(Px & Qx)  
(Ex)-Px :: -(Ex)-Px    (x)(Px & Qx) :: -(Ex)(Px & Qx)  
-(x)Px :: (Ex)-Px    -(x)(Px & Qx) :: (Ex)-(Px & Qx)  
-(x)-Px :: (Ex)Px    -(x)-(Px & Qx) :: (Ex)(Px & Qx)

Steps:  
1. Change quantifier: (x)-Px -- (Ex)-Px
2. Negate quantifier: ~(Ex)-Px
3. Negate after quantifier: ~(Ex)~Px
4. Drop any double negation: ~(Ex)Px

**EG** From constants: Pa   Pa & Qa  
---                     ---------------
  (Ex)Px                 (Ex)(Px & Qx)
or pseudo-names: Pt   Pt & Qt  
---                     ---------------
  (Ex)Px                 (Ex)(Px & Qx)

**El** ONLY to pseudo-names not used in a previous line:  
---
  (Ex)Px                 (Ex)-(Px & Qx)  
-----------
  Pu                   ~(Pu & Qu)
RESTRICTIONS ON UNIVERSAL GENERALIZATION (UG)

1. You may NOT universally generalize from constants

You may universally generalize only from pseudo-names, never from constants. Generalizing from an individual case is obviously INVALID by commonsense, and proven to be so by the following argument schema and interpretation, whereby the premise (1) is obviously true but the conclusion (2) is obviously false:

Let: $P_x = x$ was president of the U.S.; $a = \text{Abe Lincoln}$

1. $P_a$  
   Abe Lincoln was president of the U.S.  

**INVALID:**

2. $(x)P_x$  
   Everything was president of the U.S.

2. You may NOT UG from pseudo-names that occur in a line obtained by El

When a given pseudo-name (say, $t$) ever occurs in a line that is obtained by El (as $t$ does in line 3 below), you may not universally generalize from it — even if the pseudo-name was itself originally obtained by UI (as $t$ was at line 2 below). The following derivation and interpretation shows that this move can lead from truth (Everyone has a parent) to falsity (Someone is everyone's parent). Therefore, it is **INVALID:**

Let: The Domain = people; $P_{xy} = x$ is a parent of $y$

1. $(x)(Ey)P_{yx}$  
   Everyone has a parent
2. $(Ey)P_{yt}$  
   / UI, 1
3. Put  
   / El, 2 [Cannot UG from $t$]

**INVALID >>>**

4. $(x)P_{ux}$  
   Some person $u$ is everyone's parent
5. $(Ey)(x)P_{yx}$  
   / EG, 4 Someone's everyone's p
RESTRICTIONS ON EXISTENTIAL INSTANTIATION (EI)

1. You may NOT existentially instantiate to constants

You may existentially instantiate only to pseudo-names, never to constants: the following argument schema, existentially instantiating to a constant, is shown to be INVALID by the following interpretation, whereby the premise (1) is obviously true but the conclusion (2) is obviously false.

Let: \( P_x = x \) is president of the U.S.; \( d = \) Princess Diana

\[
\begin{align*}
1 & \quad (\exists x) P_x \quad \text{Someone is president of the U.S.} \\
\text{INVALID:} & \quad 2 \quad P_d \quad \text{Princess Diana is president of the U.S.}
\end{align*}
\]

2. You may NOT EI to a pseudo-name already introduced in the derivation

When a given pseudo-name (say, \( t \)) has been previously introduced in a derivation (say, by UI at line 2 in the example below), you must existentially instantiate to a different pseudo-name (say, \( u \)). The following interpretation shows that existentially instantiating to a pseudo-name already introduced can lead from truth (Everyone has a parent) to falsity (Someone is his own parent). The INVALID move is at line 3; line 4 is legal by EG.

Let: The Domain = people; \( P_{xy} = x \) is a parent of \( y \)

\[
\begin{align*}
1 & \quad (x)(\exists y) P_{xy} \quad \text{Everyone has a parent} \\
2 & \quad (\exists y) P_{yt} / \text{UI, 1} \\
\text{INVALID:} & \quad 3 \quad P_{tt} \quad \text{Someone } t \text{ is his own parent} \\
4 & \quad (\exists x) P_{xx} \quad \text{There is someone who's his own parent}
\end{align*}
\]
IV. The RECON Program

1. Reconstructing Arguments in Valid Symbolic Form

RECON is short for 'reconstruction.' The RECON program is designed to exercise you in the basic steps of formal argument reconstruction, i.e. the reconstruction of natural language arguments in valid symbolic form. Formal argument reconstruction involves symbolizing an argument's stated premises and conclusion, supplying any additional premises necessary to render the argument valid, and proving the argument valid by deriving the conclusion from the premises. Before looking at RECON's functions and the steps of formal argument reconstruction, let's recall the point of symbolizing arguments. Why bother symbolizing natural language arguments?

One way to get clear about a given line of reasoning, about the assumptions that must be made for a conclusion to follow logically, is to reconstruct the line of reasoning in the precise form of a deductively valid argument. This enables one to identify key assumptions, to see the precise logical form these must take in order to support a given conclusion, to uncover tacit assumptions (unstated premises needed to make the argument valid), and to single out inessential assumptions (logically superfluous premises, premises not needed for the argument to be valid).

In many of the arguments we encounter, especially in philosophic disputes, there will be tacit premises (assumptions that are not stated but that must be made explicit for the argument to be valid). Before we can adequately evaluate any argument, we must make any tacit premises (unstated but necessary assumptions) explicit. The first step in our reconstruction of any argument will be to analyze and represent the logical form of the argument's stated premises and conclusion: this provides an important clue to the form and content that any tacit premises must have in order to make the argument valid.

Consider the following argument.

As the first step in reconstruction, I've represented the logical form of the argument symbolically to its right:

\[
\begin{align*}
&H = F \\
&F = M \\
&\text{So, your life will take on purpose only if you're apt to be miserable.} \\
&P = M
\end{align*}
\]

Once the logical form of the argument is represented as above we can tell by inspection that the argument is invalid: There's no way to derive the conclusion from the given premises. But, in this case, it should also be clear from inspection of the argument's logical form that a premise
of the following logical form would be sufficient to make the argument valid:

\[ P = H \]

1. \( H = F \) / Premise Stated
2. \( F = M \) / Premise Stated
3. \( P = H \) / Premise Unstated
4. \( H = M \) / HS, 1, 2
5. \( P = M \) / HS, 3, 4

The derivation above shows that a premise of the form

\[ P = H \]

Your life will take on purpose only if you have ambition

is sufficient to make the argument valid.

Once you’ve depicted the logical form of an argument in some clear and explicit (symbolic) way, it’s easier to inspect the argument for validity and to see what forms of additional premise(s) will make it valid. This is why our first task in reconstructing and analyzing an argument will be to analyze and represent (symbolize) its logical form: to check the argument for validity and unstated premises needed for validity.
2. **RECON**'s Three Phases of Argument Reconstruction

The reconstruction of arguments in valid symbolic form using sentential logic involves three basic steps or phases:

1. Assigning sentential letters (P, Q, R, S, etc.) to the atomic sentences of the stated argument.

2. Identifying and symbolizing the stated premise(s) and conclusion of the argument.

3. Providing any tacit or 'missing' premises required to make the argument valid.

We symbolize the stated premises and conclusion of the argument in order to x-ray and assess its precise logical form for validity. Sometimes it is possible to tell by inspection whether a given argument form is valid or not. When an argument's form is invalid, its logical form itself will provide some clue to the form any additional premise(s) must take in order to make the argument valid. Once we think we have a valid argument form, we can prove it valid by deriving its conclusion from the premises. The **RECON** program will always require that you prove that the form of the argument you are reconstructing is valid by deriving its conclusion from its premises. Thus the **RECON** program will lead you through the following three phases of formal argument reconstruction:

1. Assigning sentential letters (we use letters F through S) to the atomic sentences of the stated argument.

2. Identifying and (using the assignments from Phase 1) symbolizing the stated premise(s) and conclusion of the argument.

3. Deriving the conclusion of the argument — providing any tacit or 'missing' premise(s) required to make the argument valid.

When you supply additional premises to make an argument valid, you will need to keep in mind two rules of thumb (since there is in principle an infinite number of additional premises that will make any invalid argument valid):

a. Make no stated premise of the argument logically superfluous. (**RECON** will tell you whether there are any superfluous lines in your derivation once you have derived the conclusion.)

b. When there are alternative additional premises, each of which will make the argument valid, choose the most plausible candidate.

The HINTs for some of **RECON**'s problems will give clues to the relative plausibility of alternative premises. Often, the assessment of plausibility will be a matter of common sense. Often, it is more complicated. We will discuss the definition and standards of plausibility later in the course. See Appendix I of your logic text for some illustration of this constraint on the reconstruction of arguments.

In Phase 3 of reconstruction, your basic task is to derive the conclusion from the given
premises. Your basic problem is to construct a derivation. RECON will guide you through this task as do BERTIE and ARGUE, checking each line you enter in your derivation for well-formedness and validity. When the original argument, with its stated premises, is not valid, you are posed the additional task of constructing some additional premise(s) that will allow you to derive the conclusion. In this respect, the derivation problems in RECON are more complex than the initial problems in BERTIE.

In supplying an argument with additional premises to make it valid, you must consider not only the logical form and validity of the argument, but also its content, the sense and plausibility of the premises you add. For this purpose, in Phase 3 of argument reconstruction, RECON allows you access to an English translation of any line of the argument or your derivation. The ENGLISH command allows you to inspect both the argument’s logical form (an x-ray of its skeletal structure) and its content, side-by-side: ENG will give you an English translation of the whole argument/derivation so far; ENG N will translate line N of your derivation; ENG N-M will translate lines N through M. The use of this command in Phase 3 of argument reconstruction is illustrated in Section 3.3 below.

Before you can assess an argument for validity, or reconstruct it in evidently valid form, you must have a precise representation of its given logical form. Symbolization is one standard way of representing the precise logical form of an argument. The RECON program guides you through the symbolization of arguments using sentential logic, in Phases 1 and 2 of argument reconstruction. RECON’s functions in these initial tasks of argument reconstruction are illustrated in Sections 3.1 and 3.2 below.
3. Running the RECON Program: Key Commands

As with any program in the ANALYTICS package, you can run the RECON program either by running ANALYTICS and choosing option 9 (RECON), or by running RECON directly as follows:

@Run(PCOK)RECON <CR>

**ARGUMENT RECONSTRUCTION PROGRAM**

Which problem file do you want? **Set 1** <CR>

Type 'INS' for instructions

There are currently 6 problems available.

Enter a problem number or STOP:

? **INS** <CR>

RECON is designed to aid you in learning to reconstruct arguments in valid logical form, to construct missing premises and explicitly to derive conclusions from premises. There is a library of arguments with which you may work.

For each argument you will be asked to:

- Assign variables to all the atomic sentences
- Symbolize the conclusion and stated premises
- Construct a valid derivation of the conclusion, providing any missing premises.

The following commands are available:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INS</td>
<td>Prints instructions</td>
</tr>
<tr>
<td>STOP</td>
<td>Ends the program</td>
</tr>
<tr>
<td>NEW</td>
<td>Starts a new problem; ends current one</td>
</tr>
<tr>
<td>LIST</td>
<td>Gives a clean listing of current problem</td>
</tr>
<tr>
<td>HINT</td>
<td>Gives aid in solving the current problem</td>
</tr>
<tr>
<td>HELP</td>
<td>Prints the answer to current problem where possible</td>
</tr>
</tbody>
</table>

Enter problem number or STOP:

?

The first thing the RECON program is apt to ask you is to identify the type of terminal you are on. (This familiar interaction is not shown in the above illustration.)

Unlike BERTIE and ARGUE, which allow you to enter your own problems to work on, RECON operates with stored problems only. RECON needs to know immediately which problem set you want. In the example above, I requested Set 1. I could also have typed simply the number 1. (In BERTIE and ARGUE SET and PRO are commands which must be typed out when you request a set or problem number, whereas in RECON you will always be operating in
the given problem file and requesting only problems by number, so it is sufficient to type the number alone.)

If you wish to stop working on a problem before you've finished it in order to choose a different problem, type NEW in response to RECON's prompt, the question mark '?'.

Unlike BERTIE and ARGUE, which allow you to switch back and forth between problems and problem sets simply by giving the PRO and SET commands respectively, you must type NEW to choose a new problem in RECON.

At present, if you want to change problem sets in RECON, you must type STOP, run RECON again, and then request the new problem set when prompted by the query 'Which problem file do you want?'

If you wish to exit RECON at any point, simply type STOP in response to RECON's prompt, '?'

Unlike BERTIE and ARGUE, which allow you to go to work immediately on the task of deriving a conclusion from given premises, RECON guides you through two initial phases of argument reconstruction: the assignment of sentential letters to atomic sentences and the symbolization of the given premises and conclusion of the argument. There are four separate sets of instructions for the RECON program, each of which you get by typing INS at each of four stages in RECON:

- When first entering the RECON program, INS gives you the general instructions and basic commands for the program, as illustrated in the sample interaction above.

- In the first phase of argument reconstruction, INS gives you the specific instructions and commands available for assigning sentential letters to the atomic sentences of the given argument (illustrated in section 3.1 below).

- In the second phase of argument reconstruction, INS gives you the specific instructions and commands for symbolizing the stated premises and conclusion of the given argument (illustrated in Section 3.2 below).

- In the third phase of argument reconstruction, INS gives you the specific instructions and commands for deriving the conclusion from the stated and any needed additional premises (as illustrated in Section 3.3 below).

Throughout the three phases of argument reconstruction in RECON, the LIST command will give you a clean listing of your work on the problem, HINT will give you a hint to its solution, and HELP will give an answer to the current step in the problem (if your instructor has provided hints and answers).
When you have finished a problem or RECON has printed its instructions, the program will instruct you (as above):

Enter a problem number or STOP:

The next three sections provide illustration of what it might look like to work through each of the three phases of the reconstruction of an argument in RECON.

3.1 Phase 1: Assigning Sentence Letters to Atomic Sentences

I will show you what it might look like to run the RECON program, request a problem set, request a problem in that set, and work through the first phase of argument reconstruction (for which I will print the instructions and then get a clean listing of the problem, using the INS and LIST commands respectively).

@Run<PCOK>RECON <CR>

Argument Reconstruction Program

Which problem file do you want? Set 1 <CR>

Type 'INS' for instructions.

There are currently 6 problems available.

Enter problem number or STOP:

? Problem 4 <CR>

I won't pass the course unless I pass the final. But, my God, there is too much material that I just do not know. So, I won't pass the course. RATS! Why didn't I take advantage of office hours and extra-help sessions??

Please type the 3 variable assignments.

P:? INS
The first step in reconstructing this argument is to assign variables to the atomic sentences. The sentences must be assigned in the order in which they appear in the argument. Type the sentences, omitting any unnecessary words.

The following commands are available:

- **INS** Prints instructions
- **STOP** Ends the program
- **NEW** Starts a new problem; ends current one
- **LIST** Gives a clean listing of current problem
- **HINT** Gives aid in solving the current problem
- **HELP** Prints the answer to current problem where possible

P:? **LIST**

After printing its instructions, RECON prompts me again to assign the sentence letter P to an atomic sentence of the argument. We will follow the convention of assigning sentence letters to atomic sentences in the order that the sentences appear in the text of the argument. Before proceeding with the assignments, I requested a clean listing of the argument text using LIST (see the next page for this listing).

Notice that the first atomic sentence in the argument text is

\[ \text{I will pass the course} \]

not

\[ \text{I won't pass the course} \]

which is a negation. The negation is represented explicitly when we symbolize the argument (in Phase 2) by use of the negation sign '~~'. In Phase 1, we assign sentence letters to atomic sentences only.

In the interaction illustrated below, I type in the first assignment correctly, ask for a hint to the second assignment, and type the second and third assignments inaccurately: within bounds, RECON will tolerate and correct spelling or typographical errors, as follows:
I won't pass the course unless I pass the final. But, my God, there is too much material that I just do not know. So, I won't pass the course. RATS! Why didn't I take advantage of office hours and extra-help sessions?!

Please type the 3 variable assignments. [Variables = sentence letters]

P: ? I will pass the course

F: ? HINT

Split the argument into unique atomic sentences. This variable should be assigned atomic sentence 2.

F: ? I pass the final

Unnecessary words or letters deleted.

F: I PASS THE FINAL

M: ? There is too much material that I just do not know

Misspellings corrected.

M: THERE IS TOO MUCH MATERIAL THAT I JUST DO NOT KNOW

Good! All variables have been assigned.

Using the above variables, symbolize the conclusion and 2 premises.

Conclusion: ?

Once all the atomic sentences have been assigned sentence letters, RECON continues to the second phase of reconstruction and asks you to symbolize first the conclusion and then the stated premises of the argument. Before continuing, notice that not all of the material given in Problem 4 was assigned sentence letters. The parenthetical my God

the expletive

RATS!

and the rhetorical question

Why didn't I take advantage of office hours and extra-help sessions?

are (1) not atomic sentences and (2) are not essential to the formal logical structure of the argument. RECON (and ARGUE) problems will often require you to distinguish between logically inessential rhetorical matter and the bare logical bones of an argument. Arguments in everyday life often come disguised in even more complex trappings; part of your task here is to discern the bare logical elements.
3.2 Phase 2: Symbolizing the Conclusion and Stated Premises

We have a clean listing of the foregoing problem below, through Phase 1, the assignment of sentence letters to the argument's atomic sentences. RECON then prompts us for the symbolization of the conclusion and stated premises. In the illustration, I first ask for instructions to this phase, get a clean listing, and then take a stab at symbolization.

I won't pass the course unless I pass the final. But, my God, there is too much material that I just do not know. So, I won't pass the course. RATS! Why didn't I take advantage of office hours and extra-help sessions??

P: I WILL PASS THE COURSE  
F: I PASS THE FINAL  
M: THERE IS TOO MUCH MATERIAL THAT I JUST DO NOT KNOW

Conclusion:?

The conclusion and all of the stated premises of the argument must be symbolized using the above variables. You may introduce new variables and premises while constructing your derivation (Phase 3).

The legal symbols for the connectives are:

\[
\begin{align*}
V & \quad - \quad \text{Disjunction} \\
\& & \quad \text{-} \quad \text{Conjunction} \\
\Rightarrow & \quad - \quad \text{Implication} \\
\Leftrightarrow & \quad - \quad \text{Biconditional} \\
- & \quad - \quad \text{Negation}
\end{align*}
\]

The following commands are available:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>INS</td>
<td>Prints instructions</td>
</tr>
<tr>
<td>STOP</td>
<td>Ends the program</td>
</tr>
<tr>
<td>NEW</td>
<td>Starts a new problem; ends current one</td>
</tr>
<tr>
<td>LIST</td>
<td>Gives a clean listing of current problem</td>
</tr>
<tr>
<td>HINT</td>
<td>Gives aid in solving the current problem</td>
</tr>
<tr>
<td>HELP</td>
<td>Prints the answer to current problem where possible</td>
</tr>
</tbody>
</table>

Conclusion:?

I won't pass the course unless I pass the final. But, my God, there is too much material that I just do not know. So, I won't pass the course. RATS! Why didn't I take advantage of office hours and extra-help sessions??

P: I WILL PASS THE COURSE  
F: I PASS THE FINAL  
M: THERE IS TOO MUCH MATERIAL THAT I JUST DO NOT KNOW

Conclusion:?

Incorrect! Try again.

Conclusion:?

Incorrect! Try again.

Conclusion:?
Premise 1: \( F \rightarrow P \)
Antecedent and consequence reversed.
Incorrect! Try again.

Premise 1: \( P \rightarrow F \)

Premise 2: HELP

Premise 2: \( M \)

Type LIST or hit RETURN to begin your proof of the conclusion.

? LIST

I won't pass the course unless I pass the final. But, my God, there is too much material that I just do not know. So, I won't pass the course. RATS! Why didn't I take advantage of office hours and extra-help sessions??

P: I WILL PASS THE COURSE
F: I PASS THE FINAL
M: THERE IS TOO MUCH MATERIAL THAT I JUST DO NOT KNOW

Conclusion: \(-P\)

Premise 1: \( P \rightarrow F \)
Premise 2: \( M \)

Type LIST or hit RETURN to begin your proof of the conclusion.

?

The first two phases of the reconstruction of the argument in Problem 4 are now complete. To continue to the third phase, deriving the conclusion and providing any additional premise(s) needed for validity, you would simply hit the RETURN bar. RECON will then set the problem up as a derivation task (in the format of BERTIE and ARGUE).

3.3 Phase 3: Deriving the Conclusion -- Adding 'Missing' Premises

In the sample interaction that follows, I will illustrate the use of the following commands available in Phase 3 of RECON:

1. INS, to get instructions for the current phase of RECON
2. ALIST, to get a listing of the sentence letter assignments
3. ENG, to get an English translation of the derivation so far
4. HINT, to read hints on the solution of the problem
5. LIST, to get a clean listing of the problem/derivation so far
6. The Slash command "/", to get RECON to type a line of derivation
7. The DEL N command, to get RECON to delete lines from line N on (in this case,
for example, to go back and try another line of derivation using a different added premise, as suggested in the hints).

We use the same rules and basic commands when doing a derivation in RECON as in BERTIE or ARGUE. RECON sets up the third phase, and we may proceed, as follows:

DERIVE: \(-P\)

1. \(P \rightarrow F\) / PREMISE
2. \(M\) / PREMISE
3. ? INS

Construct a valid derivation of the conclusion, providing any additional premise(s) needed for validity.

The following commands are available:

INS: Prints these instructions
STOP: Ends the program
NEW: Starts a new problem; ends current one
HINT: Gives a hint about solving the problem
DEL N: Erases your proof from line N to the end
REM N: Removes line N only
LIST: Lists your proof with line justifications
LIST M-N: Lists lines M through N of your proof
ENG: Lists your proof with English translations
ENG M-N: Translates lines of M through N of your proof
RULES: Gives a list of the valid rules
ASSIGN: Allows you to introduce new variables and assign their English translations
ALIST: Lists atomic sentences with sentence letter assignments

3 ? LIST

DERIVE: \(-P\)

1. \(P \rightarrow F\) / PREMISE
2. \(M\) / PREMISE
3. ? ALIST

P: I PASS THE COURSE
F: I PASS THE FINAL
M: THERE IS TOO MUCH MATERIAL THAT I DO NOT KNOW

3 ? ENG

DERIVE: \(-P\)

1. \(P \rightarrow F\) I PASS THE COURSE ONLY IF I PASS THE FINAL
2. \(M\) THERE IS TOO MUCH MATERIAL THAT I DO NOT KNOW
3 ? LIST
DERIVE: \(-P\)

1. \(P \Rightarrow F\) / PREMISE
2. \(M\) / PREMISE

3 ? HINT

If you had \(-F\) you could derive \(-P\) by MODUS TOLLENS (MT).

There is a missing premise. What additional premise would CONNECT the given material/premises of the argument with \(-F\) so you could derive \(-F\) and then \(-P\)?

3 Hints remaining.

3 ? HINT

The following additional premise would allow you to get \(-F\):

\(M \Rightarrow -F\)

The following would also allow you to get \(-P\), given \(M\):

\(M \Rightarrow -P\)

But the addition of \(M \Rightarrow -P\) would make premise 1 \(P \Rightarrow F\) superfluous! Why is this not good? Because it wouldn't show the PRESUMED CONNECTION between \(-F\) (failing the final) and \(-P\) (failing the course)!!

2 Hints remaining.

3 ? HINT

A satisfactory reconstruction in valid form would be:

1. \(P \Rightarrow F\) / Premise
2. \(M\) / Premise
3. \(M \Rightarrow -F\) / Premise [ADDED]
4. \(-F\) / MP 2, 3
5. \(-P\) / MT 1, 4

3 ? LIST

DERIVE: \(-P\)

1. \(P \Rightarrow F\) / PREMISE
2. \(M\) / PREMISE

3 ? \(M \Rightarrow -P\) / PREM

4 ? /2 3 MP

4. \(-P\) / 2, 3 MP

CONGRATULATIONS! YOUR DEDUCTION IS COMPLETE.

LINE 1 IS SUPERFLUOUS.

List your deduction or type 'NEW'.

? DEL 3
Using DEL to delete lines 3 on allows us to remove the added premise and replace it with another that will not make the stated premise 1 superfluous. We may then try the derivation again, starting from line 3, as follows:

3 ? LIST

DERIVE: -P

1 P => F / PREMISE
2 M / PREMISE
3 M => -F / PREM
4 /2 3 MP
4 -F / 2, 3 MP
5 /1 4 MT
5 -P / 1, 4 MT

CONGRATULATIONS! YOUR DEDUCTION IS COMPLETE.

List your deduction or type 'NEW'.

? LIST

DERIVE: -P

1 P => F / PREMISE
2 M / PREMISE
3 M => -F / PREMISE
4 -F / 2, 3 MP
5 -P / 1, 4 MT

List your deduction or type 'NEW'.

? ENG

DERIVE: -P

NOT (I PASS THE COURSE)

1 P => F I PASS THE COURSE ONLY IF I PASS THE FINAL
2 M THERE IS TOO MUCH MATERIAL THAT I DO NOT KNOW
3 M => -F IF THERE IS TOO MUCH MATERIAL THAT I DO NOT KNOW
4 -F THEN NOT (I PASS THE FINAL)
5 -P NOT (I PASS THE FINAL)

List your deduction or type 'NEW'.

? NEW

There are currently 6 problems available.

Enter problem number or STOP:
4. Adding New Material to an Argument: The ASSIGN Command

Sometimes, in reconstructing an argument, you may want or find it necessary to add new propositions to an argument in order to explicate unstated assumptions you think essential to the case. The ASSIGN command allows you to introduce new sentence letters and to assign them English translations. Consider the sample argument above: Suppose we thought that the reason I won't pass the final is that I will panic and be unable to reason clearly. We might then introduce these new propositions, and reconstruct the argument, as follows, using the ASSIGN command.

I won't pass the course unless I pass the final. But, my God, there is too much material that I just do not know. So, I won't pass the course. RATS! Why didn't I take advantage of office hours and extra-help sessions??

P: I WILL PASS THE COURSE
F: I WILL PASS THE FINAL
M: THERE IS TOO MUCH MATERIAL THAT I DO NOT KNOW

Conclusion: -P

Premise 1: P => F
Premise 2: M

Type LIST or hit RETURN to begin your proof of the conclusion.

DERIVE: -P

1 P => F / PREMISE
2 M / PREMISE

3 ASSIGN

Variable or constant: ? S

Now enter an English assignment:
? I'm sure to panic
Variable assigned.

Variable or constant: ? R

Now enter an English assignment:
? I'll reason clearly
Variable assigned.

Variable or constant: ? [Hit RETURN to discontinue ASSIGN] <CR>

3 LIST
DERIVE: \( \neg P \)

1. \( P \Rightarrow F \) \(\text{ / PREMISE} \)
2. \( M \) \(\text{ / PREMISE} \)
3. \( ? \text{ ALIST} \)

**P:** I WILL PASS THE COURSE
**F:** I WILL PASS THE FINAL
**M:** THERE IS TOO MUCH MATERIAL THAT I DO NOT KNOW
**S:** I'M SURE TO PANIC
**R:** I'LL REASON CLEARLY

3. \( ? M \Rightarrow S \) \(\text{ / PREM} \)

4. \( ? \text{ ENG 3} \)
3. \( M \Rightarrow S \) IF THERE IS TOO MUCH MATERIAL THAT I DO NOT KNOW THEN I'M SURE TO PANIC

4. \( ? S \Rightarrow \neg R \) \(\text{ / PREM} \)

5. \( ? \text{ ENG 4} \)
4. \( S \Rightarrow \neg R \) IF I'M SURE TO PANIC THEN NOT (I'LL REASON CLEARLY)

5. \( ? F \Rightarrow R \) \(\text{ / PREM} \)

6. \( ? \text{ ENG 5} \)
5. \( F \Rightarrow R \) I WILL PASS THE FINAL ONLY IF I'LL REASON CLEARLY
6. \( ? \text{ LIST} \)

DERIVE: \( \neg P \)

1. \( P \Rightarrow F \) \(\text{ / PREMISE} \)
2. \( M \) \(\text{ / PREMISE} \)
3. \( M \Rightarrow S \) \(\text{ / PREMISE} \)
4. \( S \Rightarrow \neg R \) \(\text{ / PREMISE} \)
5. \( F \Rightarrow R \) \(\text{ / PREMISE} \)

6. \( ? / 2, 3 \text{ MP} \)
6. \( S \) \(\text{ / 2, 3 MP} \)

7. \( ? / 4, 6 \text{ MP} \)
7. \( \neg R \) \(\text{ / 4, 6 MP} \)

8. \( ? / 7, 5 \text{ MT} \)
8. \( \neg F \) \(\text{ / 7, 5 MT} \)

9. \( ? / 1, 8 \text{ MT} \)
9. \( \neg P \) \(\text{ / 1, 8 MT} \)

CONGRATULATIONS! YOUR DEDUCTION IS COMPLETE.
List your deduction or type 'NEW'.

? ENG

DERIVE:  
1 P => F  I WILL PASS THE COURSE ONLY IF I WILL PASS THE FINAL
2 M  THERE IS TOO MUCH MATERIAL THAT I DO NOT KNOW
3 M => S  IF THERE IS TOO MUCH MATERIAL THAT I DO NOT KNOW THEN I'M SURE TO PANIC
4 S => -R  IF I'M SURE TO PANIC THEN NOT (I'LL REASON CLEARLY)
5 F => R  I WILL PASS THE FINAL ONLY IF I'LL REASON CLEARLY
6 S  I'M SURE TO PANIC  . . .  SO
7 -R  NOT (I'LL REASON CLEARLY) . . .  SO
8 -F  NOT (I WILL PASS THE FINAL) . . .  SO
9 -P  NOT (I WILL PASS THE COURSE)

List your deduction or type 'NEW'.

?
IV. The ARGUE Program

The ARGUE program operates with the same commands as BERTIE, but it also has some special commands of its own that are useful for using ARGUE to construct and test your own arguments in English.

In the sample interactions that follow, notice that in the first example I have entered my own conclusion to derive along with my own premise. ARGUE then congratulates me when I have derived the designated conclusion, just as with a stored problem. Notice that after I have completed my derivation, ARGUE still allows me to LIST it or get ENGLISH translations.

Notice in the second example below that when I type 'BEGIN' in response to ARGUE's prompt, the program allows me to pursue any line of derivation: because ARGUE doesn't know in this case what my objective might be, it will let me continue until I type 'NEW' to get a new set of options (or 'STOP' in order to exit the program). The two examples below illustrate the two 'free-form' uses of ARGUE for purposes of entering one's own arguments or derivations. Also illustrated are the use of the slash '/' command to get ARGUE to type a line, and use of the LIST and NEW commands -- all of which are available in BERTIE as well. In addition, the interactions below illustrate three commands that are special to ARGUE (and RECON -- both of which handle ENGLISH argument texts):

ASSIGN: This command allows you to assign English translations to variables and constants so that ARGUE can print out a rough English translation of any line(s) of your argument.

Here it is important to remember the following conventions:
Let F through S stand for either sentences or predicates
Indicate a one or two-place predicate by one or two subscripts
For example: Fy is a one-place, Fxy a two-place predicate
Let x, y, z, or w be individual variables
Let a, b, c, or d be individual constants
If you wish to use the same letter more than once (as in the second example below), you may use numbered letters:
For example: Fix, F2x, F3x...etc
a1, a2, a3....; x1, x2, x3...etc.

ALIST: This command will print out a list of the variable/constant assignments

ENG: This command will print your derivation with English, if you have entered English translations using ASSIGN.
ENG N gives the English for line N
ENG N-M gives the English for lines N through M
EXAMPLE 1: 'Enter your own conclusion to derive...'

@Run<PC0K>ARGUE

Free-form Argument Construction Program

Type 'INS' for instructions, otherwise,

Enter a conclusion to derive, request a problem, or type 'BEGIN'

? G & -P

BEGIN YOUR DERIVATION:

1  ? -{(G => P) / PREMISE
2  ? ASSIGN

Variable or constant: ? G

Now enter an English assignment: ? God has committed a grievous crime

2  ? ASSIGN

Variable or constant: ? P

Now enter an English assignment: ? God will be punished

2  ? /1 IMPL
2  -(-G V P) / 1 IMPL
3  ? /2 DEM
3  G & -P / 2 DEM

CONGRATULATIONS! YOUR DERIVATION IS COMPLETE.

List your derivation, ask for a problem, or type 'NEW'.

? LIST

DERIVE: G & -P

1  -{(G => P) / PREMISE
2  -(-G V P) / 1 IMPL
3  G & -P / 2 DEM
List your derivation, ask for a problem, or type 'NEW'.
? ENG 1

1  \(-(G \Rightarrow P)\)  NOT (IF GOD HAS COMMITTED A GREIVOUS CRIME
     THEN GOD WILL BE PUNISHED)

List your derivation, ask for a problem, or type 'NEW'.
? ENG

3  G & \(-P\)  GOD HAS COMMITTED A GREIVOUS CRIME AND
     NOT (GOD WILL BE PUNISHED)

List your derivation, ask for a problem, or type 'NEW'.
? NEW

EXAMPLE 2: ...or type 'BEGIN'

[After listing my last derivation I typed 'NEW' to begin anew on something else. In response, ARGUE prompts me as usual:]

Enter a conclusion to derive, request a problem, or type 'BEGIN'
? BEGIN

Begin your derivation:

1  ? \(-((x)(P1x \Rightarrow P2x))\)  / PREMISE

2  ? ASSIGN
Variable or constant: ? P1x

Now enter an English assignment: ? x is prosecuted

2  ? ASSIGN
Variable or constant: ? P2x

Now enter an English assignment: ? x is punished

2  ? ALIST

Plx: x IS PROSECUTED
P2x: x IS PUNISHED

2  ? LIST

1  \(-(x)(P1x \Rightarrow P2x)\)  / PREMISE

2  ? /1 IMPL

2  \(-(x)(\neg P1x \lor P2x)\)  / 1 IMPL

3  ? /2 DEM
3  - (x) - (P1x & -P2x)     / 2  DEM

4  ? /3  QN

4  (Ex) (P1x & -P2x)     / 3  QN

5  ? LIST

1  - (x) (P1x => P2x)     / PREMISE
2  - (x) (-P1x V P2x)     / 1  IMPL
3  - (x) - (P1x & -P2x)    / 2  DEM
4  (Ex) (P1x & -P2x)      / 3  QN

5  ? ENG 1

1  - (x) (P1x => P2x)

   NOT (FOR ALL x, IF x IS PROSECUTED THEN x IS PUNISHED)

5  ? ENG 4

4  (Ex) (P1x & -P2x)

   FOR SOME x, x IS PROSECUTED AND NOT (x IS PUNISHED)

5  ? NEW

Enter a conclusion to derive, request a problem, or type 'BEGIN'

At this point you could STOP and exit the program, enter your own conclusion to derive, request a problem from a problem set, or begin again as above. Remember: Whatever English assignments you make within a derivation are forgotten by ARGUE as soon as you go on to a NEW derivation.
When you are working on a stored problem in one of ARGUE's problem sets, you may wish to review the original argument text given at the beginning of the problem. The following is an illustration of the use of the TEXT command for this purpose, along with the ENG and ALIST commands in an ARGUE problem:

Persons have a right to life, allright! And it's morally wrong to kill what's got a right to life. So, it's certainly morally wrong to kill a human fetus!

Supply the UNSTATED GENERAL PREMISE required to make this argument valid and derive the conclusion.

**DERIVE:** \((x)(Fx \Rightarrow Mx)\)

1 \((x)(Px \Rightarrow Rx)\) / PREMISE
2 \((x)(Rx \Rightarrow Mx)\) / PREMISE
3 ? TEXT

Persons have a right to life, allright! And it's morally wrong to kill what's got a right to life. So, it's certainly morally wrong to kill a human fetus!

3 ? ALIST

Px: x IS A PERSON
Rx: x HAS A RIGHT TO LIFE
Mx: IT'S MORALLY WRONG TO KILL x
Fx: x IS A HUMAN FETUS

3 ? ENG

**DERIVE:** \((x)(Fx \Rightarrow Mx)\)

FOR ALL x, IF x IS A HUMAN FETUS THEN IT'S MORALLY WRONG TO KILL x

1 \((x)(Fx \Rightarrow Rx)\)

FOR ALL x, IF x IS A PERSON THEN x HAS A RIGHT TO LIFE

2 \((x)(Rx \Rightarrow Mx)\)

FOR ALL x, IF x HAS A RIGHT TO LIFE THEN IT'S MORALLY WRONG TO KILL x

3 ?
ARGUE Program: Sample Interactions

The following is a print-out of some sample interactions with the ARGUE program. It includes illustration of: (1) running the program; (2) program commands; (3) use of the ENGLISH command to obtain English translations of symbolized material; (4) use of the HINT command to obtain hints for solving the problem; (5) use of the LIST command to obtain a clean listing of the derivation so far; (6) use of the TEXT command to retrieve the statement of the problem and argument text; (7) use of the slash '/ ' command to have the program type out lines or make substitutions according to the legal rules of inference or replacement; (8) use of the SET and PROBLEM commands for gaining access to other problem sets and problems (which one may do at any point while in the program).

Sample student input is what is typed in after the program prompt, which is the question mark '?'. Our sample student in this illustration will first read through the stored hints for the problem and then work through and list the derivation. This is a simple exercise in supplying a 'missing' premise and proving the validity of the resulting argument by deriving the conclusion.

@RUN ARGUE

Free-form Argument Construction Program

Type 'INS' for instructions, otherwise,

Enter a conclusion to derive, request a problem, or type 'BEGIN'

? INS

ARGUE is designed to aid you in learning to construct your own arguments to given conclusions in valid deductive form while allowing you to consider the content and plausibility of your argument.

The following commands are available:

SET N  Opens problem set number N
PRO N   Gives you problem number N
STOP    Ends the program
NEW     Starts a new problem or argument; ends current one
HINT    Gives a hint about solving the problem
DEL N   Erases your proof from line N to the end
REM N   Removes line N only
LIST    Lists your proof with line justifications
LIST M-N Lists lines M through N of your proof
ENG     Lists your proof with English translations
ENG M-N Translates lines of M through N of your proof
RULES   Gives a list of the valid rules
ASSIGN  Allows you to assign English translations to new constants
TEXT    Prints original text of the problem/argument
Enter a conclusion to derive, request a problem, or type 'BEGIN'

? SET 2
There are 5 problems in this set.

Enter a conclusion to derive, request a problem, or type 'BEGIN'

? PROBLEM 2

Persons have a right to life, allright! And it's morally wrong to kill what's got a right to life. So, it's certainly morally wrong to kill a human fetus!

Supply the UNSTATED GENERAL PREMISE required to make this argument valid and derive the conclusion.

Let:  Px = 'x is a person'
     Rx = 'x has a right to life'
     Mx = 'It's morally wrong to kill x'
     Fx = 'x is a human fetus'
     t / u = 'Arbitrary thing t / u'

DERIVE: (x)(Fx = • Mx)

1  (x)(Px =  Rx) / PREMISE
2  (x)(Rx =  Mx) / PREMISE
3  ? HINT

REMEMBER TO USE THE ENG COMMAND IN ORDER TO DISPLAY THE ENGLISH VERSION OF THE ARGUMENT ONCE THE ORIGINAL STATEMENT OF THE ARGUMENT HAS SCROLLED OFF THE SCREEN.
5 Hints remaining.

3 ? ENGLISH

DERIVE: (x)(Fx =  Mx)

FOR ALL x, IF x IS A HUMAN FETUS THEN IT'S MORALLY WRONG TO KILL x

1  (x)(Px =  Rx)
    FOR ALL x: IF x IS A PERSON THEN x HAS A RIGHT TO LIFE
2  (x)(Rx =  Mx)
    FOR ALL x: IF x HAS A RIGHT TO LIFE THEN IT'S MORALLY WRONG TO KILL x
3  ? HINT
You must supply the tacit assumption needed to make the argument valid as an EXPLICIT PREMISE. Isn't it obvious? Either of these will do:

(a) Human fetuses are persons
(b) Human fetuses have a right to life

But while either will make the argument VALID, (b) makes premise (1) superfluous whereas (a) posits a connection between persons and rights.

4 Hints remaining.

3 HINT

Since the original argument posits a connection between being a person and having a right to life, your reconstruction of the argument should preserve this connection: the argument seems tacitly to assume that it is ON THE BASIS OF BEING PERSONS that human fetuses have a right to life.

The argument tacitly assumes that HUMAN FETUSES ARE PERSONS.
So, state this!
3 Hints remaining.

3 HINT

Use UNIVERSAL INSTANTIATION (UI) on the premises in order to display their underlying sentential structure. You must instantiate to pseudo-names (t) in order to later derive the given conclusion by UNIVERSAL GENERALIZATION: What goes for EVERYTHING x goes for ARBITRARY THING t by UI; and what goes for ANY ARBITRARY THING t goes for EVERYTHING x by UG.
2 Hints remaining.

3 HINT

Take 'Human fetuses are persons' (x) (Fx = Px) as an explicit premise

Get:
Pt = Rt by UI
Rt = Mt by UI
Pt = Mt by HS
Ft = Pt = Mt by UI and HS
1 Hints remaining.

3 HINT

Given: IF ANY ARBITRARY THING t is a human fetus, it's wrong to kill t
Then: For EVERYTHING x: If x is a human fetus, it's wrong to kill x

Thus, you can get (x) (Fx = Mx) from Ft = Mt by UG
where Ft = Mt was previously derived using UI
0 Hints remaining.

3 ? LIST
Persons have a right to life, allright! And it's morally wrong to kill what's got a right to life. So, it's certainly morally wrong to kill a human fetus!

Supply the UNSTATED GENERAL PREMISE required to make this argument valid and derive the conclusion.

Let:

\( P_x = 'x\) is a person' \n\( R_x = 'x\) has a right to life' \n\( M_x = 'It's morally wrong to kill x' \n\( F_x = 'x\) is a human fetus' \n\( t / u = 'Arbitrary thing t / u' \n
\( (x)(Fx = Mx) \)
List your derivation, ask for a problem, or type 'NEW'.

? LIST

DERIVE: \((x)(Fx = Mx)\)

1 \((x)(Px = Rx)\) / PREMISE
2 \((x)(Rx = Mx)\) / PREMISE
3 \((x)(Fx = Px)\) / PREMISE
4 Pt = Rt / 1 UI
5 Rt = Mt / 2 UI
6 Ft = Pt / 3 UI
7 Pt = Mt / 4, 5 HS
8 Ft = Mt / 6, 7 HS
9 \((x)(Fx = Mx)\) / 8 UG

List your derivation, ask for a problem, or type 'NEW'.

? SET 51
There are 9 problems in this set.

Enter a conclusion to derive, request a problem, or type 'BEGIN'

? PROBLEM 1

Is preventing serious hurt to other people the only legitimate ground for justifying coercion or prohibition by law? Is HURT the only form of HARM?

Well, advertising the pleasures and techniques of sodomy on a large billboard in public is surely offensive even if it’s not seriously hurtful to people. But if preventing serious hurt to others is the only legitimate ground for justifying prohibition by law, then (i) ONLY what is seriously hurtful may legitimately be prohibited and (ii) what is offensive but not seriously hurtful may NOT be prohibited. Therefore, serious hurt to others is not the only ground for justifying prohibition by law.

Be sure all the stated and additional required premises are symbolized. Then DERIVE the conclusion. Use Indirect Proof: CP + REDUCTIO.

Let: a = the act of advertising sodomy in public
d = the act of preventing actual hurt to other people
Gx = x is the only legitimate ground for justifying prohibition
Lx = x may legitimately be prohibited
Hx = x is seriously hurtful to other people; Ox = x is offensive

DERIVE: \(-Gd\)

1 Oa & \(-Ha\) / PREMISE
2 ? PROBLEM 4
About my skinning my own dog alive -- when it's done secretly, say, in the privacy of my basement, it is neither hurtful to people nor offensive to people. So, although it is morally objectionable, my skinning my dog alive may NOT legitimately be prohibited so long as it's done secretly.

SYMBOLIZE the stated premise of this argument below, and SUPPLY the unstated premises required to obtain the conclusion. One of the missing premises is a tacitly assumed principle of legitimate prohibition: you must construct the conditions it alleges to be NECESSARY and/or SUFFICIENT for justifying coercive prohibition.

Let:  
Lx = x may legitimately be prohibited  
Hx = x is hurtful to persons  
Ox = x is offensive to persons  
Sx = x is done secretly  
Mx = x is morally objectionable  
a = my skinning my dog alive

DERIVE: Ma & (Sa = -La)