Homework 0: Getting Started

Due:

The section headings below give the names of handouts or sections in the *User's Guide to Project Vincent* that you should read to help yourself with the problems.

1 Vincent Account Information Sheet, User's Guide to Project Vincent: 3.1

1. (10 points; basic) Obtain a Project Vincent user account. If you have trouble (this is likely if you just arrived at ISU), note the message and go to 197 Durham; and let us know you are having problems, and hand this problem in as soon as you get your account.

   Then write down on a piece of paper your family name, your given name, the last 6 digits of your University ID number (i.e. your social security number), your Project Vincent login ID, your local phone number (if you have one), and your section (A1, A2, B1, B2, or C1). The format should look like the following example

   ```
   Family Name  Given Name Last 6 of ID  login     Phone     Section
   Leavens,     Gary       123456        leavens   294-1580  A1
   ```

   We will use this information for grading reports, class lists, and as a means to contact you. (We won't let anyone see your phone number except the course staff.)

2 User's Guide to Project Vincent: 3.2-3.5, How to Access the On-line Workshop for Project Vincent

2. (0 points; suggested practice) On a Project Vincent workstation, run the On-line Workshop for Project Vincent, as described in the handout *How to Access the On-line Workshop for Project Vincent*.

3 Emacs Reference Card, User's Guide to Project Vincent: 4.1–4.2

3. (0 points) Start the `emacs` program and run the on-line tutorial for emacs by typing control-h, then a "t", and then following directions. This will take a while.

   You do not need a workstation for this, but if you are not using a workstation, be sure that you read the information in the handout "Getting to and Using Project Vincent without a Workstation" in the sections about using the ISN and emacs. You'll have to remember that where the tutorial says `C-s`, you will type `C-\`, and where it says `C-q`, you will type `C-^`.

   If you already know emacs or another editor (e.g., vi) you may skip this problem.

4. (5 points; extra credit) What does `M-x` mean in emacs documentation?

   What are the commands for moving up and down in a buffer without using the cursor (arrow) keys?

   What do you type in emacs to cancel a command?

   What sequence of commands do you type in emacs to move a line from one place to another?

How do you undo mistakes in emacs?

How do you create a file using emacs?

What are the names of the commands in emacs apropos "scheme"?

What does the emacs command `C-x 4 f` do?

4 User's Guide to Project Vincent: 4.1–4.2, 3.4.7–3.4.8, 3.4.15, 3.5, How to Run Scheme

5. (3 points; basic) Create a file `~/.startup.tty` by editing it to contain the following:

```
# Add the Chez Scheme locker, so I can run Chez Scheme by just typing scheme
# The -n tells add (really attach) not to bother to authenticate me.
add -n scheme

# Add the Com S 227 locker, so I can get files relating to the course
add cs227
```

   Check that your file ends in a newline, by typing `cat .startup.tty`; if the prompt for the next command does not start in the usual place, you are missing a newline: edit the file and hit "return" after the last character in the file.

   Check that the file is *exactly* as shown. Computer programming requires a great deal of precision typing, and this is practice for it, even though the lines that start with `#` are comments.

   In Unix, files whose names start with a dot (.) are hidden and not shown by a normal "ls" command. To show them use `ls -a`. Try `ls -a` now.

   Log out and then log in again. Check that now you can start Chez Scheme by typing `scheme` at the vincent prompt. Try typing some input to Scheme; for example, try `(expt 2 0.5)`. Type `(exit)` or C-d to get out of Scheme.

   When your file is complete, print it, and hand in a copy to your TA.

6. (5 points) Edit your `~/.emacs` file so that it contains the following lines.

```
;; tell emacs where to find the files chez-scheme.el* and chez-xscheme.el*
(setq load-path
      (cons "/home/cs227/emacs" load-path))

;;; tell emacs what files should be loaded for what commands
(autoload 'chez-xscheme "chez-xscheme"
          "mode for chez scheme program interaction" t)
(autoload 'scheme-mode "chez-xscheme"
          "mode for chez scheme programming and interaction" t)

;;; tell emacs that scheme files end in .ss
(setq auto-mode-alist
      (append '(("\\.ss$" . scheme-mode))
      auto-mode-alist))
```

   Note that the double quotes `"` are single characters.

   Check that you typed these lines in correctly by starting a fresh emacs, editing a file named `foo.ss`, and checking that you are in Scheme mode in the buffer for that file. In Scheme mode, look at the output of `C-h m` (control-h then "m") and play with typing in some code from the "course policies and procedures" handout; use tabs and see what emacs does.

   Hand in a printout of your `.emacs` file. (This is required even if you use some other editor.)

7. (5 points; basic) Make a directory, called `bin` for historical reasons, to contain the Unix shell script that will be made below.

   ```
   mkdir ~/bin
   ```

   Now create a file `~/bin/scheme` that contains the following text.

   ```
   #!/bin/sh
   # scheme -- customized start to chez scheme

   /home/scheme/bin/dec/scheme /home/cs227/lib/springer-friedman.ss $*
   ```

   Change the unix file mode (permissions) on the file you just created to make it executable. To do this execute the following commands from the Vincent Shell.

   ```
   cd ~/bin
   chmod +x scheme
   cd
   ```

   The `cd` (change directory) command changes your shell's current working directory. When given no arguments, it returns you to your home directory.

   Check that this has been done correctly as described in the handout "How to Run Scheme". Then hand in a printout of your file `~/bin/scheme`.

8. (0 points; suggested practice) To give you more practice with emacs and to personalize your environment more, try the following.

   The news readers on Project Vincent, and some other programs use a file named `~/.signature`. Make a file `.signature` in your home directory. It should contain your street address, your e-mail address, and (if you wish) your phone number. It should be no more than 3 lines long.

   The `finger` program on Project Vincent prints a file, `~/.plan` when someone uses `finger` to locate you. See the manual page for `finger` and page 34 of the *User's Guide to Project Vincent*. You can put whatever you like in this file, but it typically should include the same kind of information as in your `.signature` file, and in addition a few witty sayings or some other stuff you want people to learn about you. Try `finger leavens@iastate.edu` for a (boring) example. Use the `finger` command on yourself to see if it has the information correct. Use the `chfn` command to change the information in the top part (before the words `Plan:`) if it is incorrect or missing. Try fingering your friends.

   The `.cshrc.mine` file in your home directory can be used to customize the Unix shell. Create this file and put in it the following line.

   ```
   alias rm 'rm -i'
   ```

   Make sure the file ends in a newline. This says to make the `rm` command (for removing files) always ask for confirmation; it may save your neck sometime. If you are a DOS user, you can put other lines in the file to alias your favorite DOS commands to their Unix equivalents. For example, you might want to use the following.

   ```
   alias dir ls
   alias copy cp
   alias rename mv
   ```

5 User's Guide to Project Vincent: 11.1

9. (0 points) Start an e-mail message to yourself, but typing the command `comp` at the Vincent prompt. Send the message to yourself by filling in your login name after the characters "`To:`" on the same line, and then put as the subject the words "a test message from ... guess who!." Then in the body of the message, following the line `--------` but not on that line, put the text of your message:

   ```
   Dear me,
      I'm learning all about Unix.
   I think it's neat

      I have to be sure to get this
   exactly right, as computer programming
   takes great precision.  Even a one character
   mistake can cause my program to fail!

           Sincerely,
           <My name goes here>
   ```

   Put your name in place of `<My name goes here>`.

   (The subject and signature are important for the reader of your mail. I've often gotten mail from students without a signature, from whom I then have no idea who they are until I look up their login id in my files.)

   After you get the text of the message correct, send it to yourself by typing (assuming you are using emacs as your editor) C-x C-c, answering `y` to the question about saving the file `draft`, and then answering with `send` to the `What now?` prompt.

10. (0 points) Check your mail as described in section 11.1.3 of the *User's Guide to Project Vincent* to be sure you got it right.

11. (5 points; basic) Your TA will send you a piece of e-mail (after you have told them your login id). Reply to it by e-mail as directed in the message. Be sure to include a subject and a signature. Send a carbon copy to yourself, so that you have a record of the message in case it gets lost.

6 User's Guide to Project Vincent

12. (0 points; suggested practice) Use emacs to read the file `/home/cs227/doc/reading-news.txt`. Create the files that are suggested and try reading the news for our class. Post the message "I can read the news" to our news group, with subject "can read" and summary "I can read it". Read the news and see who else can read it.

13. (0 points; suggested practice) Read the rest of chapter 3, and chapter 15 of the *User's Guide to Project Vincent*.

14. (0 points; suggested practice) Use emacs to type in homework for another class you are taking. Print it and see how impressed your teacher is with typewritten work!