

Instructions

Complete the problems assigned below and turn in your answers by the due time above. To receive full credit, you must show both the relevant command(s) and output. If a question asks about what a command does, or what the output would be, give both the command and the output. Demonstrate that you actually performed the command, and didn't just guess.

Homework Problems

1. Modify your **trm**, **trecov**, and **tempty** programs from homework 4 to handle the following additional requirements (you do not need to turn in several versions of the programs - just the final version that handles all of the requirements below). Hint: adding and validating one feature at a time minimizes problems.
 - a. The Trash Can should be created as needed, and should live in the current directory. It should also be removed when no longer needed. Consider calling it **.trash** so that it is hidden from view!
 - b. Handle any number of files/directories as arguments. Do this without using loops.
 - c. Handle filenames that contain spaces. Hint: search for **\$@** in the **sh** man page
 - d. Add appropriate usage messages to the programs. These are output whenever improper arguments are supplied.
 - e. Do not hard-code the program's name in the program.
 - f. Exit with appropriate exit codes; consider normal exits and error exits.
 - g. Define and consistently use exit codes - place their definitions within comments at the top of the programs.
2. Write a program that takes one argument, a number, and outputs its largest decimal place in English. Here are the requirements:
 - a. The program outputs one of **One**, **Ten**, **Hundred**, **Thousand**, ... up to **Hundred Million** depending on the size of the number. For example, for the number 999, the program outputs **Hundred**, because the number only uses at most hundred's decimal place. For the number 8 you would output **One**, and for the number 2998 the output would be **Thousand**. Issue an error message if the value exceeds the limitations of the program.
 - b. The program *must* use nested **if** / **else** statements to solve the problem, but it cannot use **elif**.
 - c. The program must be consistently indented and formatted using one tab per indent level.
 - d. Handle incorrect syntax to your program.
 - e. Exit with appropriate exit codes; consider normal exits and error exits.
3. Rewrite the program from problem 2 above by removing the nested **if** statements, using instead **elif** statements. You should find this version much more readable.
4. Rewrite the program from problem 2 once again, this time using a single **case** statement instead of any **if** statements.
5. Write a program that shows three versions of the same basic conditional statement that outputs the word **Less** if the single numeric argument passed on the command line is less than 10. Use an **if** statement for one version, the **&&** operator for the second, and the **||** operator for the third. Each statement will only output the word **Less**; the program is silent in all other cases.

Extra Credit Problems - Do one or more of the Exercises on pages 186-187 for extra credit. Extra credit will only be received if you have completed the required problems above.