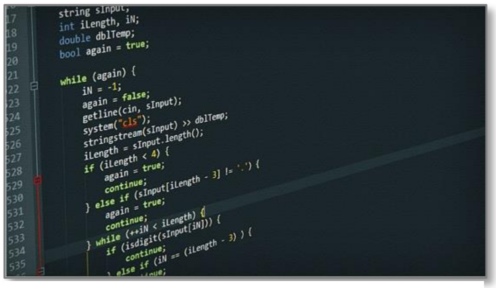
Programming Assignment: Warm-up

Description

This assignment will ask you to review basic Unix and C skills to help you determine your level of readiness for this course. Refer to any Unix and C books and/or online documentation you deem helpful as you complete the following problems.



```
string sInput;
int iLength, iN;
double dblTemp;
bool again = true;

while (again) {
    iN = -1;
    again = false;
    getline(cin, sInput);
    system("cls");
    stringstream(sInput) >> dblTemp;
    iLength = sInput.length();
    if (iLength < 4) {
        again = true;
        continue;
    } else if (sInput[iLength - 3] != '.') {
        again = true;
        continue;
    } while (++iN < iLength) {
        if (isdigit(sInput[iN])) {
            continue;
        } else if (iN == (iLength - 3) ) {
```

Problems

[1] In the following code, the first printf() reached produces the output “14” but the second printf()can cause a bus error or segmentation fault. Why?

```
main ( )
{ int *p;
funct ( p);
printf(“%d\n” , *p) ;
}
funct (int *p2)
{
p2=( int *)malloc(4);
*p2=14;
printf (“%d\n” , *p2) ;
}
```

[2] Write a C program that reads a text file and prints out any words that begins with a user given string. the filename should be given at the command line as an argument. the program should prompt the user for the search string. the program should then read the file one word at a time and front out the word if its first N bytes match the search string, where N is the length of the search string.

[3] Explain the purpose of the following Unix commands: ls, cat, rm, cp, mv, mkdir, cc.

[4] Using your favorite editor, create a small text file. Use cat to create another file consisting of five repetitions of this small text file. Use wc to count the number of characters and words in the original file and in the one you made from it. Explain the result. Create a subdirectory and move the two files into it.

[5] Write, compile, and execute a C program that prints a welcoming message of your choice.

[6] Write, compile, and execute a C program that prints its arguments.

[7] Using getchar() write a program that counts the number of words, lines, and characters in its input.

[8] Create a file containing a C function that prints the message "hello, world". Create a separate file containing the main program which calls this function. Compile and link the resulting program, calling it hw.

[9] Look up the entries for the following topics in your system’s manual; the cat command, the printf function, and the write system call.

[10] Write a function that computes some basic statistics for a list of numbers and stores those results in parts of a struct. In particular, given this definition:

```
          struct numlist {  float *list;   /* points to list of numbers */
                            int   len;     /* number of items in list   */
                            float min,     /* the minimal value in list */
                                  max,     /* the maximal value in list */
                                  avg;     /* the mean of the numbers   */

};
```

write a function `compute_stats(struct numlist *listptr)` that takes as an argument a pointer to a struct numlist with list and len already initialized and computes and fills in the other three members.

[11] This problem has 2 parts.
Note*: If you are not able to do the first part, you are not prepared to take this class. If you find the second part extremely tricky, you will find the assignments for the course difficult and potentially more time consuming than you expect.*

Part 1 - Write a program that prints a range of lines from a text file. The program should take command line arguments of the form:

```
lrange 10 20 filename
```

will print lines 10 through 20 of the named file. If there are not enough lines in the file, the program should print what it can.

Part 2 - Write a program called last10 that prints the last ten lines of a text file. The program can be used from the command line with:

```
last10 filename or
last10
```

If there is no filename, last10 processes standard input.

Submission

The submission structure is as follows:

1. Complete your draft of the programming assignment.
2. Use feedback received from instructor to improve your code.
3. Complete your final version of the programming assignment.
4. Submit your final version.

Deliverables

Use the following naming convention for your files, noting changes to the submission location and extension:

- **Draft:** LastnameFirstname-Coursenumber-Assignment title.txt
  (e.g.: SmithJane-CS43203-Warm-up-Draft.txt).

- **Final**: LastnameFirstname-Coursenumber-Assignment title.txt
  (e.g.: SmithJane-CS43203-Warm-up-Final.txt).