Name:

|  |  |  |
| --- | --- | --- |
| Problem | Points | Score |
| 1 | 50 |  |
| 2 | 60 |  |
| 3 | 70 |  |
| 4 | 80 |  |
| 5 | 90 |  |
| 6 | 100 |  |
| Total | 100 |  |

Notes:

1. For this exam you are allowed to open a terminal window on your computer, you are allowed to web surf with Google, but you cannot use online chat or other interactive services.
2. Your code and results should be placed in directories p01, p02, …, p06. It must compile with a Make file and run with the interface requested for each problem to get credit. In each directory, the name of the executable must be myprog.py.

**The first step in this exam is to create a workspace in the following directory:**

**/data/courses/ece\_1111/current/exams/exam\_04/lastname\_firstname**

**Set the permissions using “chmod -R u+rwx,g-rwx,o-rwx <lastname\_firstname>” so only you have read and write permission to this directory. Create six subdirectories within this directory: p01, p02, …, p06. Put ALL your code in these directories. Do not touch your files after the exam is over.**

This exam is structured similar to the way exam no. 3 was structured. Start with problem 1. When done, copy your code to the next problem and continue editing it. Only turn in code that is completely working. I will grade the highest level you submit. If it doesn’t work, you will get a 50 for this exam (we call this the “you didn’t debug your code and wasted my time” penalty). Therefore, what you submit must work and meet the stated requirements.

**Goal:** Implement a windowed grep function that produces identical behavior to this command:

nedc\_999\_[1]: sd /data/courses/ece\_1111/current/exams/exam\_04/picone\_joseph/

/data/courses/ece\_1111/current/exams/exam\_04/picone\_joseph

nedc\_999\_[1]: grep -i -B 2 -A 2 "medication" x1.txt

line 3

line 4

**meDication** John Smith

line 6

line 7

nedc\_999\_[1]: grep -i -B 2 -A 2 "John Smith" x2.txt

line 3

line 4

meDication **John Smith**

line 6

line 7

The options “B” (number of lines before) and “A” (number of lines after) control the window size.

**Problem No. 1**: Write a program with this interface:

myprogr.py <word> filename

The program should loop over the file, reading one line at a time, and print any line that contains the target word. For example:

myprogr.py medication x1.txt

Should print the one line in x1.txt that contains the word “medication”. Note that matches must be case-insensitive (ignore case). Do not read the file into memory. Read it line by line. There are reasons for this that become obvious below.

**Problem No. 2**: Enhance your program so that it stores data in a buffer that is N lines long. Add a command line argument:

myprogr.py N <word> filename

where N is the number of lines to be stored (the window size).

The program should keep the last 5 lines read in a vector of character strings (a list or any other comparable data structure is fine as long as the order of the lines is preserved). Print the contents of this buffer using the following format. I should clearly see the data shifting through your buffer.

… reading line no. 27 …

buf[0] = … line 1

buf[1] = … line 2 …

buf[2] = … line 3 …

buf[3] = … line 4 …

buf[4] = … line 5 …

… reading line no. 28 …

buf[0] = … line 2

buf[1] = … line 3 …

buf[2] = … line 4 …

buf[3] = … line 5 …

buf[4] = … line 6 …

**Problem No. 3**: Only print N lines centered around the line that matches the target word. In other words, your program will produce something very close to the grep examples above – it prints N lines centered around each match that it finds. If multiple matches are found in the file, it prints N lines for every match:

… found <word> at line no. ??? …

buf[0] = … line 2

buf[1] = … line 3 …

buf[2] = <… this line contains the target word …>

buf[3] = … line 5 …

buf[4] = … line 6 …

Assume the value of N is odd, or arbitrarily decide where the center is if N is even.

**Problem No. 4**: Modify the program of problem 3 so that it prints the matching text on the line in which the match occurred in uppercase. Only the matched word should be printed in uppercase, and it should be printed in uppercase whether or not the input was lowercase, uppercase or mixed case. The rest of the line should be printed as is. This is what some call “highlighting the match.”

**Problem No. 5**: Modify the interface of problem no. 4 so that two target words can be specified:

myprogr.py 5 <word1> <word2> filename

Only print a match if both words exist on the matching line.

**Problem No. 6**: Modify the behavior of problem no. 5 so that two target words can be appear anywhere in the buffer of N lines:

myprogr.py 5 <word1> <word2> filename

Only print a match if both words exist somewhere in your N line buffer. Highlight each word found by printing it in uppercase.