COMP2113 Programming technologies

ENGG1340 Computer Programming II

Quiz Mar 27, 2023

12:30-14:20

Time Allowed: 110 minutes

Total: 15 marks

Submission

Please complete this quiz through Moodle. No late submission will be accepted.

This quiz involves some console input/output. You are reminded that the evaluation system on Moodle (a.k.a. VPL) evaluates your program with a full score under the condition that your program output is an exact match with the expected output. In other words, any additional or missing space character, tab character, newline character, etc. will be treated as errors during the evaluation of your program and will lead to 0 mark. If your submission cannot be run in the evaluates system, you will get 0 mark as well.

We will grade your programs with another set of test cases. Therefore, you are advised to make more test cases on your own for testing your program.

Q.1 Shell Scripts (7 marks)

Part (a) (3 marks)

Please write a shell script `getWords.sh`.

- The script will take one command line argument.
- The script will read the content of a file specified by the argument.
- It will then find and print out every word (case sensitive) in the file, one word per line.
- All words will be printed only once in the order they appear in the file.

Assumptions:

- You can assume that the input file will only consist of words separated by spaces, tabs or newlines (in Linux format).
- The script will print nothing if the input file does not contain any word, or if there is not enough argument specified.

Hints:

- You may use "gred -w" to match whole words only. That is, "apple" will not be matched with "pineapple".

Assume we have `wordlist.txt` in the current directory, with the following content:

```
apple pineapple
APPLE Orange     orange
orange Apple
```

If we run the script like this:

```
$ ./getWords.sh wordlist.txt
```

The script will print:

```
apple
pineapple
APPLE
Orange
orange
Apple
```

Part (b) (4 marks)

Please write a shell script `getKeyword.sh`.

- The script will take one command line argument.
- The script will read the content of a file specified by the argument.
- It will then find and print out the most occurring words (case sensitive, may not be unique) and the number of occurrences of these words in the file, one word and its frequency per line.
- If there are more than one words sharing the highest frequency, print them and their frequency in alphabetical order.

Assumptions:

- You can assume that the input file only consists of words separated by spaces, tabs or newlines (in Linux format).
- The script will print nothing if the input file does not contain any words, or if there is not enough argument specified.

Hints:

- You can make use of the commands “sort” and “uniq” to get the number of occurrences of each word if the words are listed one by one on each line.
- For the “uniq” command, using the flag -c can return the row count.
- For the “sort” command, you can use “-r” to sort in reverse order, “-n” to sort a file numerically used –n option, “-k” to sort on a certain column. For example, use “-k 2” to sort on the second column.
- You can make use of the command “head –n x” to return the first x rows.
- In shell script, you can use () to declare an array and ${} to extract elements by index.

Example:

```
LIST=('Jan' 'Feb' 'Mar' 'Apr' 'Jun' 'Jul' 'Aug')
```

```
echo ${LIST[0]}
echo ${LIST[1]}
echo ${LIST[2]}
```

Output:

```
Jan
Feb
Mar
```

Sample Run:

Assume we have **wordlist.txt** in the current directory, with the following content:

```
apple pineapple
APPLE Orange     orange
orange Apple
```

If we run the script like this:

```
$ ./getKeyword.sh wordlist.txt
```

The script will print:

```
orange 2
```

Q.2 C++ Programming (4 marks)

Write a program to accept an integer from the user. This integer can be positive or negative. The program then reverses the digits in the input integer and the sign. All trailing zeros should be ignored. We assume all inputs are proper integers and not overflowed.

Part of the program is given below:

```
#include <iostream>

using namespace std;

int reverse(int N) {
   // Several missing statements

}

int main() {

   int N;
   cin >> N;

   cout << reverse(N) << endl;

   return 0;
}
```

Copy the above into your program and complete the missing statements in the reverse function.

Sample runs of the program are as follows. Your output format should be exactly the same. In all the test cases below, input is in the first line while output is in the second line.

Test 1:
```
123
-321
```

Test 2:
```
-456700
7654
```

Test 3:
```
9876543
-3456789
```

Q.3 C++ Programming (4 marks)

Let n be a positive integer and let S(n) denote the number of divisors of n.

For example, S(1) = 1, S(4) = 3, S(6) = 4.

A positive integer p is called antiprime if S(n) < S(p) for all positive n < p. In other words, an antiprime is a number that has a larger number of divisors than any number smaller than itself. And the smallest antiprime is 1.

Given an integer b (negative, positive or 0), your program should output the smallest antiprime that is larger than or equal to b.

Input specification:

• An integer b.

For 30% of test cases, b <= 100

For 80% of test cases, b <= 1000

For 100% of test cases, b <= 100000

Output specification:

• The smallest antiprime more than or equal to b.

Hint: You may need to implement a function to represent S(n) in your program.

In all the test cases below, input is in the first line while output is in the second line.

Test 1:

```
11
12
```

Test 2:

```
12
12
```

Test 3:

```
-10
1
```

Test 4:

```
700
720
```

Test 5:

```
45678
50400
```

~ End of Quiz ~