ENGG1340 Programming Technologies / COMP2113 Computer Programming II

Assignment 3

Deadline: 13 April (Saturday), 2024 23:59

If you have any questions, please post to the Moodle discussion forum on Assignment 3.

> **Problem 1 (30 marks)**
>
> **Problem 2 (35 marks)**
>
> **Problem 3 (35 marks)**

Total marks: 100 marks

General Instructions

Please read the instructions in this document carefully.

In this assignment, you will solve 3 tasks, and a tester will automatically test your submitted program. Therefore, if your submitted files and program outputs do not conform to our instructions given here, your programs cannot be evaluated and you will risk losing marks totally.

One sample case is provided with every task in this document. Note that the test cases may or may not cover all boundary cases for the problem. It is also part of the assessment whether you are able to design proper test cases to verify the correctness of your program. We will also use additional test cases when marking your assignment submission.

Coding environment

For all the problems, make sure the following compilation command is used to compile your program:

```
g++ -pedantic-errors -std=c++11 [your_program_name].cpp –o vpl_execution
```

Submission

Name your programs as the following table shows, and submit them to the corresponding VPL tester. **You will risk receiving 0 marks for this assignment if you submit incorrect files.** Resubmission after the deadline is not allowed.

Filename	Description
1.cpp	Problem 1
2.cpp	Problem 2
3.cpp	Problem 3

Late submission

If you submit within 3 days after the deadline, there will be a 50% mark deduction. After that, no mark will be given.

Evaluation

Your code will be auto-graded for technical correctness. In principle, we use test cases to benchmark your solution, and you may get zero marks for not being able to pass any of the test cases. Normally partial credits will not be given for incomplete solution, as in many cases the logic of the programs are not complete and an objective assessment could be difficult. However, your work may still be considered on a case-by-case basis during the rebuttal stage.

Academic dishonesty

We will be checking your code against other submissions in the class and from the Internet for logical redundancy. Please be reminded that no matter whether it is providing your work to others, assisting others to copy, or copying others will all be considered as committing plagiarism and we will follow the departmental policy to handle such cases. Please refer to the course information notes for details.

Getting help

You are not alone! If you find yourself stuck on something, post your questions to the course forum, send us emails or come to our support sessions. We want this assignment to be rewarding and instructional, not frustrating and demoralizing. But we don’t know when or how to help unless you ask.

Discussion forum

Please be careful not to post spoilers. Please don’t post any code that is directly related to the assignments. However, you are welcome and encouraged to discuss general ideas on the discussion forums. If you have any questions about this assignment, you should post them in the discussion forums.

Problem 1

Please write a C++ program which meets the following requirements.

- The program will read a whole line from the standard I/O.
- It will then print the length of this line.
- It will print the sub-sequence following the rightmost whitespace.
- It will count the vowels (a, e, i, o, u, case-insensitive) in the sub-sequence.
- Finally, it will print the statistical results with the following format.

  `<char>: <num>`

  , where `<char>` and `<num>` denote each individual vowel and its count. Besides, a colon `:` and a whitespace ` ` are used to connect these two items.

Assumptions:

- The line only contains the characters in the ASCII Character Set.
- The line is not empty, and its length is less than `100`.
- The number and positions of whitespaces are totally not restricted, which means they can appear anywhere.
  - If there is no whitespaces in the line, the whole line should be analyzed for vowels.
  - If the rightmost whitespace appears at the end of the line, an empty string should be analyzed for vowels. Apparently, the counts of all vowels should be zero in this case.
- After `u: <num>`, please print a `\n`.

Sample:

Assume we have a line with the following content:

```
an orange
```

The program will print:

```
9
orange
a: 1
e: 1
i: 0
o: 1
u: 0
```

Problem 2

Suppose you are engaged in an adventure activity in the ruins of an ancient civilization and come across a wide river that you need to cross. Luckily, there is a decaying wooden bridge over the river, although some of the planks have fallen off. This scene can be modelled as shown in the figure below, with you on the lefthand side of the river and wanting to reach the other side. The complete bridge consists of 3x5 wooden planks, where each square represents a plank. The number 0 indicates a missing plank, while 1 indicates an intact plank. To minimize risks, you decide to only walk across planks that are connected by one edge. However, before taking actions, you are wondering whether this bridge can be passed or not.

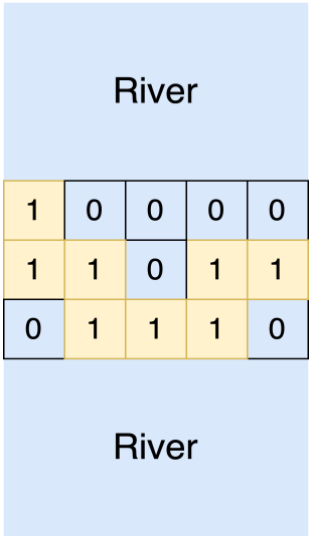

Figure 1

Please write a C++ program to determine if the bridge can be passed.

- The program will read a line (i.e. the path of a txt file) from the standard I/O.
- The program will read the content of this file, containing the shape of plank array and the status of each plank.
- If the bridge can be crossed, the program will return 1. Otherwise, it will return 0.

Assumptions:

- The first line consists of two integer numbers(i.e., the number of rows and columns respectively) to represent the array shape of the planks.
- The following lines consists of 0-1 sequences to denote the status of each plank.
- All the numbers in the same line are separated by a whitespace.
- Please use `cout << "1" << endl;` or `cout << "0" << endl;` to match the output format exactly.

Sample:

- Take the above figure as an example. The input file should be

  ```
  3 5
  1 0 0 0 0
  1 1 0 1 1
  0 1 1 1 0
  ```

  The bridge has 3 rows and 5 columns. The status of the planks in the first row is `1 0 0 0 0`, meaning that only the first plank can be passed.

- You can pass the bridge along the green path. Therefore, your program should print `1`.

Attention: For the solution, you are strongly suggested to use a recursive function. It will not only enhance your understanding of recursive functions but also result in a more concise and organized solution compared to using multiple loops.

Problem 3

Please write a C++ program which meets the following requirements.

- Read two lines (i.e. two integer sequences) from the standard I/O, and organize them into two separate linked lists.
- Sort them in ascending order, and print the ordered integers, respectively.
- Merge them into one linked list.
- Delete negative numbers and all duplicates such that each non-negative integer appears only once, and print the final linked list.

Assumptions:

- Typically, each line only consists of integers separated by random number of whitespaces.
- In some cases, each line could optionally be empty or full of whitespaces, denoting an empty linked list.
- Whitespaces may appear at the beginning or end of the sequences.
- The head is the leftmost integer, while the tail is the rightmost integer.
- For any empty linked list, you still need to print a `\n` to represent its existence.

Sample:

1. Assume we have two lines with the following content:

```
      2 4 3    1
-1  7    3  5 2
```

2. The following figures (i.e., Figure 2 and 3) show vanilla and sorted linked lists, respectively.

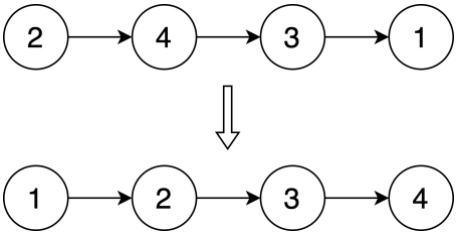

Figure 2

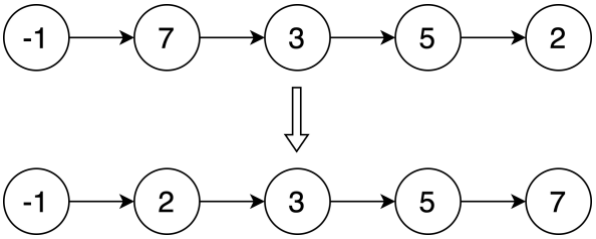

Figure 3

3. After the merging and deletion, the final linked list is shown in Figure 4.

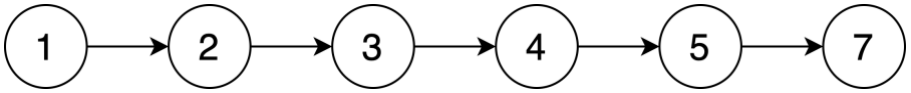

Figure 4

4. Hence, the output should be

```
1 2 3 4
-1 2 3 5 7
1 2 3 4 5 7
```

Hint: Here is a demo to read a line from the standard I/O, and convert each segment to an integer before printing every number. You may need it, especially the usage of `stringstream` and `stoi`, to read integers efficiently.

```
#include <iostream>
#include <string>
#include <sstream>
using namespace std;


int main()
{
```

```
    string line;
    getline(cin, line);
    stringstream ss(line);

    int num;
    while (ss >> num)
    {
        cout << num << endl;
    }

    return 0;
}
```

Attention: Even if this problem can be solved with other approaches, only linked lists are suggested for your solution to enhance your understanding of linked lists.