ENGG1340 / COMP2113, Assignment 3

Due Date: Nov 30, 2020 23:59

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

- General Instructions
- Problem 1: Game of Elimination in C++ (45 marks)
- Problem 2: STL - Log Analyzer (50 marks)

Total marks: 100 marks

- 5 marks for proper code comments, indentation and use of functions
- 95 marks for program correctness

General Instructions

Read the instructions in this document carefully.

In this assignment you will solve 2 tasks and a tester would automatically test your submitted program. So 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.

Sample test cases are provided with each 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.

Input and output format

Your C++ programs should read from the **standard input**. Also, your program output should be printed through the **standard output**. If you failed to follow the instructions, the tester may not able to give a score for your program. Additionally, you should strictly follow the sample output format (including space, line breaker, etc.), otherwise, your answer might be considered as wrong.

How to use the sample test cases

Sample test cases in text file formats are made available for you to check against your work. Here's how you may use the sample test cases. Take Problem 2 test case 3 as an example. The sample input and the expected output are given in the files `input2_3.txt` and `output2_3.txt`, respectively. Suppose that your program is named "2", do the followings at the command prompt of the terminal to check if there is any difference between your output and the expected output.

```
./2 < input2_3.txt > myoutput.txt
diff myoutput.txt output2_3.txt
```

Testing against the sample test cases is important to avoid making formatting mistakes. The additional test cases for grading your work will be of the same formats as given by the sample test cases.

Coding environment

You must make sure that your program can compile, execute and generate the required outputs on our standard environment, namely, the gcc C++11 environment we have on the CS Linux servers (academy*). As a programmer/developer, you should always ensure that your code can work perfectly as expected on a target (e.g., your client's) environment, not only on yours.

While you may develop your work on your own environment, you should always try your program (compile & execute & check results) on our standard environment before submission.

Make sure the following compilation command is used to compile your programs:

```
g++ -pedantic-errors -std=c++11 [yourprogram].cpp
```

Submission

Name your C++ programs as the following table shows and put them together into one directory. Make sure that the folder contains only these source files (`*.cpp`) and no other files. **Compress this directory as a `[uid].zip` file where [uid] is your university number** and check carefully that the correct file have been submitted. We suggest you to download your submitted file from Moodle, extract them, and check for correctness. Resubmission after the deadline is not allowed.

A maximum of 5 marks will be deducted if you fail to follow the submission instructions strictly.

Note that code template for Problem 2 is provided, and you should base your work on it.

Problem	Code files provided?	Files to Submit
1	Create your own `1.cpp`	`1.cpp`
2	`2.cpp`	`2.cpp`

Late submission

If submit within 3 days after the deadline, 50% deduction. After that, no mark.

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.

Discussion forum

You are not alone! If you find yourself stuck on something, post your question to the course forum. 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.

Problem 1: Game of Elimination in C++

Consider a prize to be awarded to a winner among n > 0 contestants by a game of elimination. The n contestants first line up in a line, and are assigned the numbers 1 to n one by one.

The host of the game then decides on an integer k > 1, and starting from contestant 1, the host counts k contestants down the line and eliminates the k-th contestants from the circle. He then continues to count for another k contestants, and eliminates the k-th contestants from the line. When he counts to the end of line, he will continue counting from the beginning of the line. This process repeats until there is only one contestant remains who will be the winner of the prize.

For example, if n = 7 and k = 3,
The initial line: 1234567, count to 3 and contestant 3 is eliminated
Line now becomes: 124567, continue counting from 4 and contestant 6 is eliminated
Line now becomes: 12457, continue counting from 7 and contestant 2 is eliminated
Line now becomes 1457, continue counting from 4 and contestant 7 is eliminated
Line now becomes 145, continue counting from 1 and contestant 5 is eliminated
Line now becomes 14, continue counting from 1 and contestant 1 is eliminated
Winner is contestant 4

Write a **C++ program** which implements a circular linked list (see Module 8 Guidance Notes p.92) to determine which contestant will win the prize.

Program input. Two input numbers n and k (n > 0 and k > 1).

Program output. The winner of the game.

Requirement: You will need to implement the circular linked list on your own, i.e., you may NOT use any STL containers or external linked list libraries.

Idea:

- Use the program `build_list_forward.cpp` of Module 8 as a basis for your work. The program essentially has built a linked list; to turn it into a circular linked list, you just need to point the next pointer of the last node to the head of the list.
- You will need to build a circular linked list containing the numbers 1 to n
- You will then need to traverse through the linked list and remove a node once you have traversed for k nodes.
- After removing a node, you should check if there remains only one node in the list. If yes, the winner is identified.
- Remember to free all memories used by the linked list before the program terminates.

Hint:

- Study carefully `build_list_forward.cpp` and `build_list_sorted.cpp` of Module 8. They should contain the main building blocks for you to finish this task.
- It is expected that you will only need to write no more than 20-30 lines of additional code, after reusing the appropriate codes in `build_list_forward.cpp` and `build_list_sorted.cpp`. Of course, it is OK if you write more than that.

Sample Test Cases

User inputs are shown in blue.

1_1:

```
7 3
4
```

1_2:

```
5 2
3
```

1_3:

```
1 6
1
```

1_4:

```
2 5
2
```

1_5:

```
100 5
47
```

1_6:

```
1000 8
185
```

1_7:

```
1200 23
473
```

1_8:

```
100000 9
42610
```

Problem 2: STL - Log Analyzer

You are provided with a template program `2.cpp`. Complete the program and the program will read and process a web log data file from user input (`cin`). The program will then print out the 5 most popular pages, and the 5 most active users and the corresponding pages they have visited.

The log file consists of records of three types, each record occupies exactly one line. Here is the format of these three types of record:

Page record: `PAGE <page id> <page url>`
User record: `USER <user id>`
Visiting record: `VISIT <page id>`

A **page** record represents a page on the web server. A **user** record represents a user that accesses the system. A **visiting** record represents a visit by the user indicated by the most recent user record. Here is a sample data file:

```
PAGE 1288 /library
PAGE 1282 /home
USER 20686
VISIT 1288
VISIT 1282
USER 20687
VISIT 1288
```

In this case, we have 2 pages (`/library` and `/home`) on the server. Two users have accessed the server, one (#20686) visiting 2 pages (`/library` and `/home`) and the other (#20687) visiting 1 page (`/library`).

You can assume the followings regarding the data file:

- The data file always consists of the three types of records only
- The page ids are unique across all PAGE records
- The user ids are unique across all USER records
- The page ids are unique across all VISIT records of a user
- VISIT records will only appear after the first USER record
- The page id in a VISIT record appears only after its corresponding PAGE record
- There will be at least 5 users and 5 pages

The followings have been implemented for you:

- A `Page` structure, each `Page` object consists of the id, path and a counter to count the number times it is being visited

```
struct Page {
    int id;
    string path;
    int counter;
    Page(int id, string path) {
        this->id = id;
        this->path = path;
        counter = 0;
    };
};
```

- A STL vector for organizing the `Page` objects

```
vector<Page> pages;
```

- A `User` structure, each `User` object consists of the id and the pages the user visited

```
struct User {
    int id;
    vector<string> visits;
    User(int id) {
        this->id = id;
    };
    void add_visit(int page_id) {
        ...
    };
    int size() const {
        return visits.size();
    };
    void print_visits() {
        ...
    }
};
```

- A STL vector for organizing the `User` objects

```
vector<User> users;
```

- A function to overload `<` operator for the comparison of 2 pages based on their ids

```
bool operator<(const Page & a, const Page & b) {
    return (a.id < b.id);
};
```

You are required to complete the missing functions in the template program to make it work.

Note: The functions `Page()` and `User()` in the structs `Page` and `User` above are called **struct constructors**. Refer to Module 10 "C Programming (Part 3)" for the discussion on the struct constructor functions. It works the same in C++.

Sample Test Cases

You are provided with 4 sample test cases. We will use the first test case and detail the test run below.

Assume we have `input2_1.txt` in the current directory. If we run the program like this (assuming that we are working on Linux):

```
$ g++ -pedantic-errors -std=c++11 2.cpp -o 2
$ ./2 < input2_1.txt
```

The program will print:

```
*** 5 most popular pages ***
36:/msdownload
32:/ie
17:/products
17:/search
13:/sitebuilder
*** 5 most active users ***
23:10068
- /activeplatform
- /activex
- /automap
- /frontpage
- /gallery
- /ie
- /intdev
- /isapi
- /java
- /msdownload
- /musicproducer
- /office
- /promo
- /sbnmember
- /search
- /sitebuilder
- /spain
- /vbasic
...
[snipped]
...
11:10019
- /athome
- /clipgallerylive
- /games
- /isapi
- /kb
- /logostore
- /msdownload
- /msoffice
- /ntserver
- /products
- /search
```

Note:

- If two pages are equally popular, the pages are ordered lexicographically.
- If two users visit the same number of pages, the users are ordered by their id ascendingly.
- The list of pages a user visit must be printed in ascending order.
- You can choose not to use the provided template program and implement in your own way. However, your program must contain the provided Page and User structures and use STL vectors for storage.
- Once again, you MUST use STL vectors in your program. Your program will be checked and your score will be 0 if we find that you use traditional arrays to store pages and users in your implementation.