Colorado Mesa University
Computer Science and Engineering

CSCI112 Computer Science 2 2021

Assignment 2 20 points

- Due dates (Hard deadlines):
  Part I March 13, 11:00 PM
  Part II March 21, 11:00 PM – Demo & discussion required

Aim:

To manipulate pointers, use some dynamic memory and start programming with complex classes and Linked Lists.

Requirements:

Over the next two assignments we are going to solve a simulation problem. Here is the description.

The people that run the Motor Registry Office (MRO) have a problem. They are concerned that people do not spend enough time waiting in lines to appreciate the privilege of owning and driving a car. The current arrangement is as follows:

- When people walk in the door, they must wait in a line to sign in.
- Once they have signed in, they are told either to stand in line for registration renewal, or to wait until they are called for license renewal.
- Once they have completed their desired transaction, they must go and wait in line for the cashier.
- When they finally get to the front of the cashier's line, if they expect to pay with a cheque, they are told that all cheques must get approved. To do this, it is necessary to go to another counter to get their cheque stamped, then rejoin the cashier's line at the end.

The object of part 2 will be to write a complete simulation of this procedure to provide the MRO with some statistics. In Part 2 you will be asked to do some designing of this task.

Here we will implement the cashier's line.

Part 1A: (8 points)

As customers of the MRO complete their renewals, they have to go to the cashier's to pay their fee. Some will want to pay by cheque. Others will pay cash.

To simulate this waiting line, each customer will arrive at the cashier's line according to data stored in a file. A sample file will be available. The data will consist of the time of arrival at the cashier's line in seconds after the opening time, followed by a method of payment indicator (either $ for cash or C for cheque).
As a customer arrives at the waiting line, one of two things will happen:
a) there is no-one there and the person gets served straight away;
b) there is at least one person in the line so the new customer must join the end of the line.

The cashier can serve a person paying cash in 30 seconds. To send a person to the cheque stamping counter takes 10 seconds. It takes a customer no time at all at this counter before they rejoin the cashier's line.

All processing times should be specified as `const` values.

When a person finally finishes their transaction we know what time it is and how long it has taken.

When the data has run out, report the total number of customers and the average time it took a customer to carry out the payment (use the chrono lib functions and namespace if you like).

Some clues:

We are performing an event-driven simulation. There are only two events that take place in this problem:

1. a customer arrives, which occurs at the time specified in the file;
2. a customer finishes being served, at a time calculated from the start time.

Other events are incidental: joining the line (happens on arrival if there's a customer being served), starting the payment process (happens immediately on arrival if no-one in line, or when the previous customer finishes). Of course, a cheque customer will rejoin the line at the end of the time required by the cashier, but then can be considered a cash customer.

Use a linked list of structs to store the customers as they join the line. The leaving time for a customer can also be in the struct, but will not be determined until they start being served as a cash customer.
Dynamic memory allocation and deletion must be used.

Have a clock which ticks over to the next of the two possible events.

When a customer finishes, add their total time to an accumulating total ready for the average to be calculated.

You should print out a running commentary like:

```
10:  Customer 1 arrives
     starts being served (cash)
23:  Customer 2 arrives
     joins the line
40:  Customer 1 finishes
     Customer 2 starts being served (cheque)
45:  Customer 3 arrives
     joins the line
50:  Customer 2 finishes
     joins the line as a cash customer
     Customer 3 starts being served (cash)
```

and so on.

Part I B: (3 points)

Please do not attempt this part until after you have thoroughly tested your code.

(a) Part 2 will have a different input, as customers will arrive at the MRO with two additional information: one of two tasks (L for license renewal, R for registration renewal) and a name. Describe a suitable data structure for a customer.
(b) There will be more than one linked list, not all of which are of the join at end/leave at head type. (The license renewals are called in name order.) Describe the different linked lists.
(c) As the customers will be moving from one line to another, and hence should not be destroyed until they finally pay, describe a suitable linked list node.
(d) Describe the different events that the complete simulation will have to handle.
(e) At each of the events given in (d), describe what happens to the customer.

Submissions: . . . . . (any deviation - will result in no grading)

If you elect to do only part 1 (max 11/20)

Submit all the three source files for Part 1 A and the text file for Part I B as follows:
1 compressed archive when uncompressed will contain 2 folders nmaed as follows:
Code → this will contain your 3 code files
Design → This will contain your text file for part B

If you do both parts (max 20/20)

Submit **all of the above as per instructions** +
A second compressed archive when decompressed will contain 1 only folder named:
P1-2-Code → Containing **all** the source files for your program

Assignment deadlines will not change! This is a long and complex assignment start now!