COMP 11 – Introduction to Computer Science – Fall 2002

Project pp6 – Version 2

Note: The original handout had several errors in the sample output that are corrected below. JGS

Programming project due the week of 10/21

Due dates:
Section 01 (Block H): Wednesday 10/23 11:00 pm
Section 02 (Block E): Thursday 10/24 11:00 pm

Submissions may begin on Monday 10/21 at 5pm.

Project pp6: 8 points for correctness and 1 point for style.
Perform a more detailed car wash simulation for one car

Overview

This is the third part in the semester long Car Wash project. Here you will extend the simulation you developed in project pp3. This will require the use of a structure and functions, which was not part of pp3. As with pp3, you must create a loop that simulates time where each iteration of the loop simulates one minute of operation. Also, like pp3, there will be only one car. However, the car wash will have 2 washing stations, each with its own cashier.

While the concept of having two washing stations and only one car is silly, this will prepare you for the next project in which there will be many cars and multiple washing stations.

For pp6, there are 6 possible states for the simulated car. We need a state for the car being in each of the cashiers and each of the washing stations. The car wash has only one queue that feeds into the 2 cashiers and stations.

- elsewhere: The car is driving around the city and is not at the car wash.

- queued: The car is in the queue waiting to pay a cashier.

- paying0: The car’s driver is paying the cashier for washing station 0 (we computer scientists love to begin numbering at zero!).

- paying1: The car’s driver is paying the cashier for washing station 1.

- washing0: The car is actually being washed in station 0.

- washing1: The car is actually being washed in station 1.

The interpretation of the above states and the rules of operation are the same as in pp3 (except you now have 6 states instead of 4).

In addition to keeping track of the state of the car, you must also keep track of the following.

- The number of times the car was washed.

- The total number of minutes spent waiting in the queue.
- The total number of minutes spent at the cashier (the cashiers may operate at different speeds).
- The total number of minutes spent being washed (ditto, different stations may require different times).

All the data for a single car **must** be stored in a single structure, i.e., you must define a struct that represents the car and that contains all the data about that car. Do not use classes – use C++ structs. All updates of data about the car must update this structure.

For each washing station, you must keep track of the following. All costs will be in dollars (floating point).

- Hourly cost of paying the cashier (this is the cashier’s salary).
- Hourly cost of operating the washer unit in the station (this is the salary of the people working in the washing unit).
- Incremental cost per car wash (for soap, water, etc.).

As with the car, all data about a given washing station must be stored in a C++ struct and you must update that struct whenever its data changes.

Also, you must keep track of data about the car wash as a whole in a C++ struct. You must track the following.

- Price of a car wash (same for both stations)
- Number of hours of operation for the given day.
- The lunch hour.
- Total number of washes for the given day.
- Total income.
- Total cost (salaries, soap, water, etc.) for the day.

If the car begins it queue time during the lunch hour, the time in the queue is doubled (popular time for car washes). The hours are numbered beginning at zero.

Your main function **must** look like the following.

1. Declare and initialize variables.
2. Call function(s) to read some or all of the data.
3. Loop – one iteration per simulated minute.
   (a) Call function(s) to read data (if needed).
   (b) Call function(s) to simulate one minute.
   (c) Update the simulated clock.

In other words, your main function should **only** contain function calls, a single loop and an update to the clock. Everything else happens in various functions you write. You may need additional structures to make this work.

Note that the requirements for using structures, functions, and a simple main function will be checked. If your program fails to meet this, you will lose half credit. As with style, we will examine your first submission only.
Output from your program

Your program must print out the progress of your car similar to pp3 except that it should only print the car’s state for minute zero and for any other minute that it changes state. For example, if the car is elsewhere for the first three minutes before it enters the queue, you should print out that it is in the elsewhere state in minute 0 and that it is in the queued state for minute 3. Do not print the car’s state for minutes 1 and 2.

It must also print the following statistics after the simulation is over.

- Average number of minutes the car spent in the queue (float).
- Average number of minutes the car spent at a cashier (float).
- Average number of minutes the car spent being washed (float).
- Profit or loss in dollars of the car wash for the simulated day. A positive number is profit and a negative number is loss (float).
- Profit/loss per hour of operation (float).
- Profit/loss per car wash performed (float).

Inputs to your program

Here are the inputs to your program, which your program must read before starting your simulation.

1. The number of hours the car wash is open for this simulated day. (An integer > 0). Your simulation will operate minute by minute. As with pp3, please number the first minute as minute 0.
2. The lunch hour. (An integer that is ≥ 0 and < the number of hours open.)
3. Price of a car wash. (A float > 0)
4. Number of minutes the car will spend in the queued state whenever it is about to get washed. (An integer > 0)
5. Number of minutes it takes to pay the cashier for station 0. (An integer > 0)
6. Number of minutes it takes to pay the cashier for station 1. (An integer > 0)
7. Number of minutes it takes to wash a car in station 0. (An integer > 0)
8. Number of minutes it takes to wash a car in station 1. (An integer > 0)
9. Hourly cost for the cashier in station 0. (A float > 0)
10. Hourly cost for the cashier in station 1. (A float > 0)
11. Hourly cost for operating the washing station for station 0. (A float > 0)
12. Hourly cost for operating the washing station for station 1. (A float > 0)
13. Incremental cost per wash for station 0. (A float > 0)
14. Incremental cost per wash for station 1. (A float > 0)
15. The number of the minute that the car gets washed the first time and the station it uses (0 or 1), followed by the number of the minute for the second wash and the station, etc. This will be terminated by the following sentinel:

-1 -1

You need not read the second -1 if you do not wish to. As with pp3, there will be enough time in between each wash for the car to complete its wash. Note in particular that the car might be washed in the very first minute, i.e., in minute 0, or immediately after its previous wash (i.e., it might go from a washing state directly to a queued state).

Sample run

Here is a sample input.

```
10 4 6.00 5 1 2 6 3 9.00 10.50 16.00 24.00 1.00 1.00
60 0
250 1
-1 -1
```

The state of the car is printed one per line. On each such line is a ‘#’ character, the number of the minute, and the name of the state. The ‘#’ indicates to the grading program that it should examine that line. Here is the output that your program should print if given the above input. After the simulation is done, you should print out the summary statistics. Precede each such number by a ‘$’, even if the value is not a dollar value.

```
Car wash simulation for one car.
Number of hours open: 10
The lunch hour is: 4
Price of a car wash: 6.00
Number of minutes spent in the queue: 5
Number of minutes spent paying cashier 0: 1
Number of minutes spent paying cashier 1: 2
Number of minutes spent being washed in station 0: 6
Number of minutes spent being washed in station 1: 3
Hourly cost for cashier 0: 9.00
Hourly cost for cashier 1: 10.50
Hourly cost for washing in station 0: 16.00
Hourly cost for washing in station 1: 24.00
Incremental cost per wash for station 0: 1.00
Incremental cost per wash for station 1: 1.00
Next time the car will be washed: 60
In station: 0
# 0 elsewhere
# 60 queued
# 65 paying0
# 66 washing0
# 72 elsewhere
Next time the car will be washed: 250
In station: 1
```
# 250 queued
# 260 paying 1 <= Note: Queue time is doubled.
# 262 washing 1
# 265 elsewhere
Next time the car will be washed: -1
In station: -1
Average number of minutes spent in queue: $7.5
Average number of minutes spent at cashier: $1.5
Average number of minutes spent being washed: $4.5
Profit/loss for the day: $ -585.00
Profit/loss per hour: $ -58.50
Profit/loss per car wash: $ -292.50

Note that the lines beginning with '
' and the numbers preceded by '$' are the only output that the grading program will examine. All other output will be discarded during automatic grading. However, the entire output of your program will be examined for style. The exact appearance of the other lines does not matter as long as it is easy for someone who is using your program to understand. Your program must be able to handle any number of car washings, as long as they fit into the hours open.

**Style**

To get the style point, you must satisfy all of the following requirements.

- Produce output that is easily read.
- Have an introductory comment at the top of your program and preceding each function.
- Have a comment for each variable describing its purpose.
- Separate functions by blank lines.
- Within each function, separate logical sections by blank lines.
- Have a comment at the start of every such section.
- Choose meaningful names for variables and functions.

Your first submission will be used for style grading, even if your best grading score is on a different submission.

**How to submit your program**

Assume that your program is in a file named pp6.cpp. To submit it, type:

```
provide comp11 pp6 pp6.cpp
```

from an Andante computer. You can submit up to 3 times. Your grade will be the highest grade you receive out of all your submissions.