Programming project due the week of 9/30

Due dates:
Section 01 (Block H): Wednesday 10/2 11:00 pm
Section 02 (Block E): Thursday 10/3 11:00 pm

Submissions may begin on Monday 9/30 at 5pm.

Project pp2: 6 points for function and 1 point for style.
Perfom the car wash simulation for one car

This is the second part in the semester long Car Wash project. Here you will actually perform part of the simulation. This will require reading in some parameters of the simulation and performing the simulation itself. The simulation must use a loop, where each iteration simulates exactly one minute of simulated time. There will be exactly one car in this simulation. There are exactly 4 possible states for the simulated car. In other words, at each minute, the car must be in exactly one of these states.

- 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 the cashier.
- paying: The car's driver is paying the cashier for the car wash.
- washing: The car is actually being washed.

The above order is important. The car begins in the elsewhere state. At some time it enters the queued state. For the queued, paying and washing states, the car stays in the given state for a fixed number of minutes where the number of minutes will be an input parameter. After the queued state, it enters the paying state, and next the washing state. After the washing state, it returns to the elsewhere state, unless it is supposed to return immediately to the car wash (some drivers always wash their car twice in a row for maximum cleanliness). After that, it may get washed several times.

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

1. The number of simulated minutes for this run. (An integer > 0) Please number the first minute as minute 0 (not 1), the second as 1 (not 2), etc. So if there are 100 minutes, they will be numbered 0-99. This numbering scheme may seem odd but it will be useful on later car wash projects.
2. Number of minutes spent in the queued state whenever it is about to get washed. (An integer > 0)
3. Number of minutes spent in the paying state. (An integer > 0)
4. Number of minutes spent in the washing state. (An integer > 0)
5. Number of times the car will get washed.
6. The number of the minute that the car gets washed the first time, followed by the number of the minute for the second wash, etc., up to the number of washes. 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.

Here is a sample input.

20 1 2 5 2 2 11

This means the simulation runs for 20 minutes, each time the car is washed it spends 1 minute in the queue, 2 minutes paying, and 5 minutes washing. It gets washed twice, at minute 2 and at minute 11.

Your program should print the state of the car in each simulated minute. The state of each minute is printed on its own 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.

```
Car wash simulation for one car.
Number of simulated minutes: 20
Number of minutes spent in the queue: 1
Number of minutes spend paying the cashier: 2
Number of minutes spent being washed: 5
Number of times the car will be washed: 2
Next time the car will be washed: 2
# 0 elsewhere
# 1 elsewhere
# 2 queued
# 3 paying
# 4 paying
# 5 washing
# 6 washing
# 7 washing
# 8 washing
# 9 washing
Next time the car will be washed: 11
# 10 elsewhere
# 11 queued
# 12 paying
# 13 paying
# 14 washing
# 15 washing
# 16 washing
# 17 washing
# 18 washing
# 19 elsewhere
```

Note that the lines beginning with ‘#’ are the only lines that the grading program will examine. All other lines 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.
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.
- Have a comment for each variable describing its purpose.
- Separate your program into logical sections separated by a blank line.
- Have a comment at the start of every such section.
- Choose meaningful names.

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 pp3.cpp. To submit it, type:

```
provide comp11 pp3 pp3.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.