University of Exeter

College of Engineering, Mathematics

and Physical Sciences

ECM2433

The C Family

Continuous Assessment

Date Set:	25th January 2021
Date Due:	18th February 2021
Return Date:	11th March 2021

This CA comprises 30% of the overall module assessment.

This is an **individual** exercise and your attention is drawn to the College and University guidelines on collaboration and plagiarism, which are available from the College website.

This is the first coursework for this module and tests your skills and understanding of programming in C.

Please Turn Over

Problem Statement

You are working for the Post Office and have been given the task of modelling how customers queue within one branch. You are going to do this by writing an ANSI standard C program to simulate this queuing system.

For those of you not familiar with a Post Office branch, customers arrive at random intervals and all join a single first-in-first-out (FIFO) queue. Each customer is served at one of a small number (hopefully greater than zero) of service points. The customer at the head of the queue goes to the first available service point and is served for some period of time, before exiting the branch. At closing time, no further customers can join the queue, but the service points stay open until all customers already in the queue have been served. If a customer arrives and the queue is already full, then they will leave without being served, and if they have been waiting too long in the queue, then they will give up, and, again, leave without being served.

Your simulation could answer questions like

- If N customers on average arrive at each time interval and there are S service points, then
  (a) How long, on average, does it take to clear the queue at closing time?
  (b) What is the optimum number of service points required to serve customers? (You will need to think about what "optimum" means in this case.)
  (c) How many customers, on average, can be served in one day?
  (d) How long, on average, does each customer wait?
  (e) How many customers, on average, give up waiting and leave without being served?

Note the use of the words "on average". Since the simulation is dependent on random numbers, each individual run of the simulation will provide one set of results, but those results might be skewed by a particularly favourable or unfavourable sequence of random numbers, so each simulation will need to be run a number of times and the results summarised, for example showing the mean and standard deviation. Since this is a C program, you should easily be able to repeat each simulation a large number of times (1 000 times say).

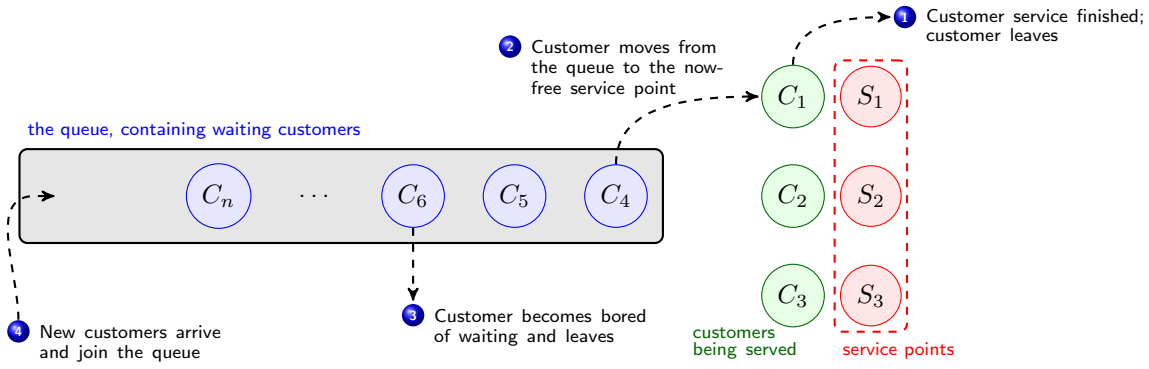

Figure 1: A diagram showing the operation of the Post Office. Here there are three service points, all serving customers. (1) At some point the customer labelled C_1 finishes being served and leaves the Post Office - a *fulfilled* customer. (2) There is now a free service point, so the customer at the head of the queue goes there to be served, leaving the queue. (3) One of the customers (labelled C_6) has decided that they have waited too long and decides to give up - a *timed-out* customer. (4) More customers arrive and join the queue, if there is room for them. If a customer arrives and finds the queue full, then they leave and become an *unfulfilled* customer.

The Task

Your task is to write a well-structured, ANSI standard C program to simulate one branch of the Post Office, using the details described below.

Running the simulator

Your executable program must be called `simQ` and should accept a number of parameters from the command line, as follows:

1. `fileIn`: The name of a text file containing the parameters that will be used to drive the simulation. The details of these parameters are described below, and an example of how the file might look is given on ELE.
2. `numSims`: The number of times the simulations must be repeated, using the parameters specified in the `fileIn` file.
3. `fileOut`: The name of the file which should be generated by the simulation, containing the output results.

For example, the command

```
simQ parameters01.txt 1000 results01.txt
```

should load the parameter values from a file called `parameters01.txt`, run a simulation based on these parameters 1 000 times (using different random numbers in each run), and then output the results of those simulations to a file called `results01.txt`.

One simulation

A simulation contains four entities, as follows:

1. The single queue, which may contain zero or more customers at any one time. For one simulation, the maximum number it can contain is stored in the parameter `maxQueueLength`; if the value of this parameter is -1, then there is no maximum. This is a FIFO queue.
2. One or more service points. For one simulation, the number of service points is stored in the parameter `numServicePoints`. Each service point can serve only one customer at a time, but at a given timepoint it might not be serving any customer if the customer queue is empty.
3. One Post Office branch, which contains the queue and the service points.
4. Customers, which arrive at random, queue up until they are at the head of the queue, are served at the next available service point, and then leave; these are referred to as *fulfilled* customers. If the queue is full when they arrive (i.e. the number of people in the queue is equal to the parameter `maxQueueLength`), then they leave immediately, and are referred to as *unfulfilled* customers. Each customer has a different tolerance for queueing; once they have been queuing for that amount of time without being served, then they leave and are referred to as *timed-out* customers.

There is a clock which measures discrete time intervals that controls the whole system. At each time interval the following happens or may happen, in this order:

1. Zero or more fulfilled customers have finished being served and leave the system.
2. A service point that is not serving a customer will start to serve the next available customer, and that customer will leave the queue.
3. If a customer in the queue has reached their waiting tolerance limit, then they leave the queue and become a timed-out customer.
4. Zero or more customers arrive into the system, and either join the back of the queue, or, if the queue is full, leave the system as an unfulfilled customer.

The Post Office closes its doors after a given period of time (controlled by the `closingTime` parameter), after which it no longer lets in any new customers (customer just stop arriving into the system). The service points continue serving customers until those remaining in the queue have all been served.

All of this is controlled by random numbers, possibly sampled from different types of distributions:

- The number of customers who arrive in a given time interval.
- The amount of time it takes to serve a particular customer.
- Each individual customer's waiting tolerance limit, i.e. the number of time intervals that the customer is prepared to stand in the queue.

The values required to control these random numbers (for example, perhaps a mean and standard deviation), are included in the parameter file input into the simulation (i.e. `fileIn`).

Simulator output

If the simulator is run just once (`numSims`=1), i.e. using a command like

```
simQ inputParameters01.txt 1 outputResults01.txt
```

then the results written to the output file (called `outputResults01.txt` in this example) must be as follows, in this order:

1. A list of the parameter values read in from the input file (called `inputParameters01.txt` in this example).
2. One record per time interval from the start until all customers have been served, showing:
   - The time interval number.
   - The number of customers currently being served.
   - The number of people currently in the queue.
   - The numbers of fulfilled, unfulfilled and timed-out customers so far.
3. A marker showing when closing time has been reached.
4. The time it takes from closing time until all remaining customers have been served.
5. The average waiting time for fulfilled customers over the whole simulation (i.e. the number of time intervals they spend in the queue).

If the simulator is run more than once (`numSims`>1), i.e. using a command like

```
simQ inParams.txt 1000 outResults.txt
```

then the results written to the output file (called `outResults.txt` in this example) must be a summary of all 1 000 simulations, as follows, in this order:

1. A list of the parameter values read in from the input file.
2. The average of each of the following across all 1 000 simulations:
   (a) Number of fulfilled customers.
   (b) Number of unfulfilled customers.
   (c) Number of timed-out customers.
   (d) The average waiting time for fulfilled customers.
   (e) The time it takes from closing time until all remaining customers have been served.

Report

As well as the simulator, you must submit a report (maximum 4 pages of A4). This must include the following:

- A description of the design decisions you have made when developing your program and any assumptions you have made.
- A description of an experiment you have performed using your simulator, and the results you obtained, including a discussion about the meaning of those results. You must include example output from at least one run of your code that performs multiple simulations (i.e. where `numSims`>1).

If your submitted code does not work fully as expected, then describe the details in the report, including any testing you have performed to narrow down what is causing the problem.

Deliverables

The deliverables for this coursework comprise the source code for your simulator, instructions for compiling and linking it into an executable, an example input parameter file (i.e. `fileIn`) and your report, zipped together and submitted via EBART. Your submission must contain the following:

- The source code must be written in well-structured, ANSI standard C.
- There must be a Linux shell script for compiling and linking your code, called `compileSim` and nothing else, and must compile your code to an executable called `simQ` and nothing else.
- You must include one example input parameter file, called `testInput.txt` and nothing else.
- The report, as detailed in the previous section.

Your program must compile, link and execute on one of the two Linux servers we are using on this module (`emps-ugcs1` and `emps-ugcs2`) and will be tested on one of those servers using parameters other than the ones you have included.

Submission must be made by 12pm (noon) on the date indicated on the front page of this document.

Marking Scheme

Criteria	**Marks**
Program basics *To what extent is your program a well-constructed and well-formatted ANSI standard C program? Does your program compile and link without errors and warnings? To what extent does it trap and report run-time errors? To what extent does your program correctly read in the command line parameters?* Processing of the parameter file *To what extent does your program correctly read the data from the input parameter file specified in the command line parameters and output the values to the output file?*	40
Function of the simulator *To what extent does the program make a suitable representations of the entities in the system? (Top marks will only be awarded if the queue is implemented as a linked list.) To what extent does the program correctly and efficiently perform the simulations and output the results?*	40
Report *To what extent does the report clearly and concisely describe your design decisions and assumptions? To what extent does it describe a plausible experiment, its outputs and the meaning of those outputs? Does the report include the specified example output? If the code does not work fully, to what extent have the errors been investigated and described in the report? Are the spelling, grammar, language and presentation of the report clear, formal and professional and appropriate to the intended audience?*	20
Total	100

Penalties

You will be penalised 10 marks if the executable code is not called `simQ`, or the Linux shell script for compiling and linking it is not called `compileSim`, or an example parameter file called `testInput.txt` containing all the required parameter values has not been included.

If your report exceeds 4 pages, only the first 4 pages will be marked.