

University of Newcastle
Discipline of Computer Science and Software Engineering
Semester 2, 2011 - SENG1120/6120

Assignment 2 – version 2

Due using the Blackboard Turnitin submission facility:
9:00AM – 10 October 2011

Changes:

- Inclusion of UML diagram request.

NOTE: *The important information about submission and code specifics at the end of this assignment specification.*

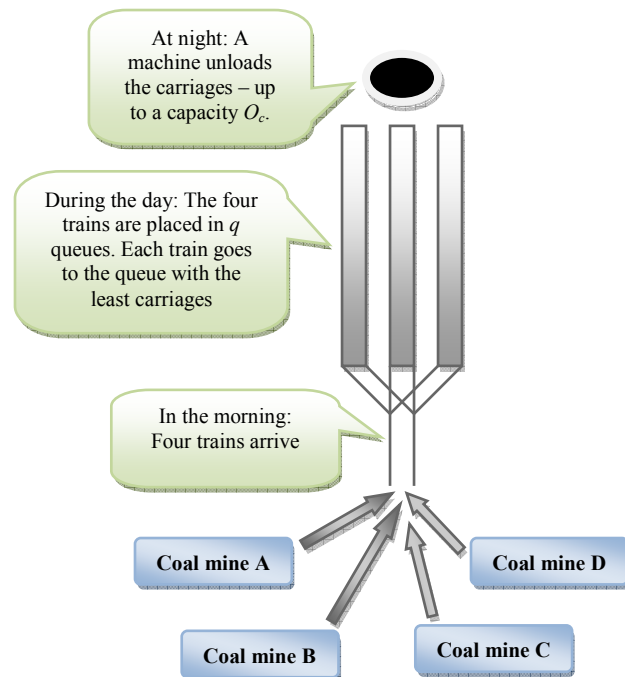
In lectures we have discussed the use of templates to provide the compiler with blueprints for functions and classes. These are used by the compiler to produce code that implements functions and/or classes that are suitable for the type(s) to which you apply them in your program code.

Assignment Task

Your task in this Assignment is to simulate the coordination of coal transportation at a track junction. The goal is to have a decision support tool to design a new unloading facility. Managers want to know the size of the track junction that has to be built to support production outflow from 4 coal mines.

The situation is as follows. There are 4 coal mines operating and trains depart from those mines once a day, arriving at the track junction in the morning. Then, they are placed in the q queues, following the dispatch rule in which *each train goes to the queue with the least number of carriages*. In the afternoon/night, the carriages are unloaded up to a capacity O_c .

Your task is to simulate the arrival of the trains and their unloading. The goal of the simulation is to analyse the impact of *different numbers of queues* and *machines with different unloading capacities* in the length of the queue formed by the coal carriages.



Technical information:

- Every day in the morning, four trains will arrive, with the following characteristics:
 - o **Number of carriages in each train:** random, in the interval [10, 20].
 - o **Amount of coal in each carriage:** random, in the interval [4,000, 5,000]
- Unloading capacity: known, variable O_c .
- Number of queues: known, variable q
- Dispatch rule:
 - o Enqueue: As trains arrive, their carriages will be placed in the queue that currently has the least number of carriages.
 - o Dequeue: Unloading occurs after all trains for the day have arrived. One carriage is taken for each queue, in sequence (1, 2,..., q , 1, 2,..., q , 1, and so on), until the unloading capacity for the day, in tons, O_c is reached.

To run the simulation you will have to:

- 1) Create a counter for the number of days.
- 2) Create q queues to store the carriages' data.
- 3) For each day, generate four trains and add their carriages to the queues, according to the dispatch rule.
- 4) After the trains were added, unload one carriage from each queue in sequence, until the amount of coal given by the unloading capacity O_c has been unloaded.
- 5) Go back to 3, repeating for 30 days of transportation/unloading.

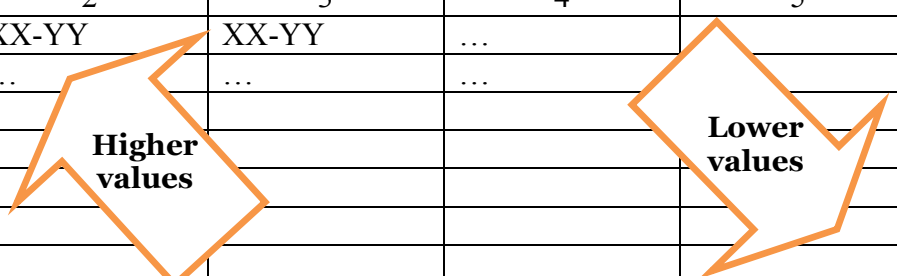
Which results your demo file should present:

Your demo should iterate through 2 up to 5 queues, and unloading capacities from 50,000 to 350,000; in 50,000 steps. After finishing, your demo should present a table (in the command prompt line), showing the results for the 30-day simulations (you do not have to align every column perfectly). The table should be as follows:

	Number of queues - q			
O_c	2	3	4	5
50,000				
100,000				
150,000				
200,000				
250,000				
300,000				
350,000				

The numbers in the table should depict to the average length of the queues; and maximum number of carriages in any of the queues during the 30-day period (counted when the trains for the day have been enqueued, i.e. before dequeuing starts). To have an idea about what sort of results are expected, the table is likely to have this configuration:

	Number of queues - q			
O_c	2	3	4	5
100,000	XX-YY	XX-YY	...	
150,000	
200,000				
250,000				
300,000				
350,000				
400,000				



Where XX is the average length of the queues, and YY is the maximum number of carriages in any of the queues.

To simulate the coal transportation, you will produce a **template class** for each of:

1. Node, which is then used as a component in the construction of
2. LinkedList, which is then used as a component in the construction of
3. Queue

You can create any other classes, in addition to those, if you believe it makes the code more generic, or easy to understand (e.g. a class to manage the simulation might be a clever design solution, though not necessary).

Node, LinkedList and Queue should be dynamic (use pointers). You should create a class Carriage, which will contain the information of the carriage cargo. Node will store instances of Carriage. It should not be necessary for me to note that these template classes should be programmed by you, and definitely *not* obtained from any alternate source. You may use hints provided in the lecture slides *only* in writing your code.

UML diagram:

You should provide a UML diagram showing the classes contained in your assignment. Remember that the main file (the file that contains the `int main()` function) is not a class so it should not be in the diagram. Private member variables and public methods should be listed, as well as the connections between classes with corresponding multiplicities. The diagram can be created using any suitable software, including Word, Rational Rose, etc.

Your submission should be made using the Assignments section of the course Blackboard site. **Incorrectly submitted assignments will not be marked. Assignments that do not use the specified class names will not be further marked.** You should provide all your .h, .cpp and .template files and, and a Makefile. Also, provide an UMLDiagram.doc file and, if necessary, a readme.txt file containing any instructions for the marker. Each program file should have a proper header section including your name, course and student number, and your code should be properly documented. Finally, a completed Assignment Cover Sheet should accompany your submission.

COMPRESS ALL THE FILES INTO A SINGLE .ZIP AND THEN SUBMIT.

This assignment is worth 10 marks of your final result for the course.