Assignment 1

COS 212

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA
Denkleiers • Leading Minds • Dikgopolo tša Dihlalefi

Department of Computer Science
Deadline: 27/02/2017 at 12:00

Objectives:

- Revision of linked lists, stacks and queues.
- Writing a working program in Java.
- Learning about the differences between C++ and Java.
- Investigating the differences and relationships between stacks and queues.

General instructions:

- This assignment should be completed individually, no group effort is allowed.
- The output of both your C++ and Java programs will be evaluated.
- Be ready to upload your assignment tasks well before the deadlines as no extensions will be granted.
- Task 1 is divided up into a number of steps to assist you. Follow each step in order to complete the assignment and produce the final deliverables for marking.
- You are NOT allowed to import any Java packages with classes that support linked list-, stack- or queue-like functionality (e.g. `ArrayList`). Doing so you will receive 0.
- The differences between C++ and Java have been deliberately left vague or omitted (e.g. Java and pointers, Java and destructors, etc.) from this specification. It is up to you to discover and investigate these differences and compensate for them.
- Your code will may inspected to ensure that you've followed the instructions.
- If your code does not compile you will be awarded a mark of 0.
- You will be afforded two opportunities to upload your submissions for automarking.

Plagiarism:

The Department of Computer Science regards plagiarism as a serious offence. Your code will be subject to plagiarism checks and appropriate action will be taken against offending parties. You may also refer to the the Library's website at `www.library.up.ac.za/plagiarism/index.htm` for more information.

After completing this assignment:

Upon successful completion of this assignment you will have implemented your own basic data structures in C++ and Java.

Task 1

Download the archive `Assignment1Code.tar.gz` and untar it in an appropriate directory. The archive contains the following files:

- main.C
- CircularList.h
- CircularList.C
- Stack.h
- Stack.C
- Queue.h
- Queue.C
- Deque.h
- Deque.C

These files contain partial C++ source code for a various simple data structures. You will have to complete the implementation for these classes by reading the comments in the code. You will also have to write your own makefile and add it to your final submission. Your makefile should produce an executable called `main`.

You must write your own code in `main.C` to test your implementation. Your main will be replaced for marking purposes.

You may NOT modify any aspect of the classes other than providing implementation for the functions in the implementation files. You may not add members or change the names of the functions or classes. The header files will be overwritten for marking purposes. Be sure to follow the output conventions exactly as given in the source file comments.

Step 1: Completing the CircularList class

Take a moment to inspect the `CircularList` class. You will notice that the `CircularList` class has a single pointer named `tail` which will point to the last node in the list. The CircularList class should be implemented as a `singly linked circular list`. You must implement all of the functions by following the instructions in comments within the source code files you were given.

Step 2: Completing the Stack class

The next structure to complete is the `Stack` class. You will notice that the `Stack` class contains a pointer to a `CircularList` object. You must store your stack's elements in this list. Complete the `Stack` class by following the instructions in comments within the source code files you were given.

Step 3: Completing the Queue class

The next structure to complete the `Queue` class. You will notice that the `Queue` class uses a stack to store its elements, similar to how the `Stack` class stored elements in a circular list. Complete the `Queue` class by following the instructions in comments within the source code files you were given.

Step 3: Completing the Deque class

The final step is to complete the `Deque` class. You will notice that the `Deque` class contains a queue object to store its elements in, similar to how the `Queue` class was defined in terms of a stack. Complete the `Dequeue` class by following the instructions in comments within the source code files you were given.

Task 2

For this step you will have to translate your C++ code into Java code. Create an additional class called `Tester`. You have to place your `main` in this class to test your code. Rewrite all of your C++ code into Java code, minding the naming conventions. Test your implementation thouroughly. Your `Tester` file will be replaced for marking purposes so ensure that you follow the naming conventions as in your C++ code. Java does not support a number of features that C++ allows. There is however a Java equivalent for everything that cannot be directly translated into Java. The `Object` class in Java will have an answer to the majority of these features.

You must create your own makefile and submit it along with your Java code in which your code should be compiled with the following command:

```
javac *.java
```

Your makefile should also include a `run` rule which will execute your code if typed on the commandline as in `make run`.

Submission Instructions:

For your submission, you must tar your Java code and C++ code, including the makefiles, into two separate archives. Your Java archive should be named sXXXJava.tar.gz and your C++ sXXXCplusplus.tar.gz, where XXX is your student/staff number. You will notice that this assignment has two upload links. Your archives must be uploaded to the appropriate links, Assignment 1 C++ for the C++ part and Assignment 1 Java for the Java part, before the deadline.