CSC3002: Introduction to Computer Science

Assignment 3

Assignment description:

You are supposed to complete the *.cpp* and *.h* files provided as requirement shown in the problems. Please pack your whole project files into a single .zip file, name it using your student ID (e.g. if your student ID is 123456, then the file should be named as 123456.zip), and then submit the .zip file via Blackboard System.

You should use the simple project containing C++ Standford Library and start your programing.

Please note that, the teaching assistant may ask you to explain the meaning of your program, to ensure that the codes are indeed written by yourself. Please also note that we may check whether your program is too similar to your fellow students' code using Moodle.

Please refer to the BB system for the assignment deadline. For each day of late submission, you will obtain late penalty in the assignment marks. If you submit more than three days later than the deadline, you will receive zero in this assignment.

Detailed description on assignment requirement is stated in the last few pages.

Question 1 and Question 2:

Q1:

In the queue abstraction presented in this chapter, new items are always added at the end of the queue and wait their turn in line. For some programming applications, it is useful to extend the simple queue abstraction into a ***priority queue,*** in which the order of the items is determined by a numeric priority value. When an item is enqueued in a priority queue, it is inserted in the list ahead of any lower priority items. If two items in a queue have the same priority, they are processed in the standard first-in/first-out order.

Using the linked-list implementation of queues as a model, design and implement a **`pqueue.h`** interface that exports a class called **`PriorityQueue`**, which exports the same methods as the traditional **`Queue`** class with the exception of the **`enqueue`** method, which now takes an additional argument, as follows:

```
void enqueue(ValueType value, double priority);
```

The parameter **`value`** is the same as for the traditional versions of **`enqueue`**; the **`priority`** argument is a numeric value representing the priority. As in conventional English usage, smaller integers correspond to higher priorities, so that priority 1 comes before priority 2, and so forth.

Q2:

Use the algorithm from section 16.5 to implement the **`PriorityQueue`** class so that it uses a heap as its underlying representation. To eliminate some of the complexity, feel free to use a vector instead of a dynamic array.

Question 3:

1.

Eliminate the recursion from the implementation of `depthFirstSearch` by using a stack to store the unexplored nodes. At the beginning of the algorithm, you simply push the starting node on the stack. Then, until the stack is empty, you repeat the following operations:

1. Pop the topmost node from the stack.
2. Visit that node.
3. Push its neighbors on the stack

2.

Take your solution from the preceding exercise and replace the stack with a queue. Describe the traversal order implemented by the resulting code.

Question 4

Complete the definition of the `Employee` class hierarchy by adding the necessary instance variables to the private sections and the implementations for the various methods. Design a simple program to test your code.

Question Requirement

Q1:

Fill the file *Q1_pqueue_list.h*.

You should use linked-list based implementation and includes all the methods in the implementations of list-based queue of the textbook.

So that when the we run the test file (do not change the codes in this file) *TestPoriorityQueue.cpp* in which *Q1_pqueue_list.h* is included we can have the printed result like:

```
Priority queue test succeeded
```

Q2:

Fill the file *Q2_pqueue_heap.h*.

You should use heap based implementation and includes all the methods shown in the implementation of list-based queue in the textbook.

So that when the we run the test file (do not change the codes in this file) *TestPoriorityQueue.cpp* in which *Q2_pqueue_heap.h* is included we can have the printed result like:

```
Priority queue test succeeded
```

Q3:

1. **Reimplements the depth-first search algorithm using an explicit stack in file "*StackDFS.cpp*"**
2. **Reimplements the breadth-first search algorithm using an explicit queue in file "*QueueBFS.cpp*"**

So that when we run the test file "AirlineGraph.cpp", we can have the results like: (if Seattle is typed in)

```
Los Angeles -> Dallas
Portland -> San Francisco, Seattle
Seattle -> Portland, Boston
San Francisco -> Portland, Dallas, Denver
New York -> Atlanta, Boston
Dallas -> San Francisco, Los Angeles, Atlanta, Denver
Boston -> New York, Seattle
Denver -> San Francisco, Chicago, Dallas
Chicago -> Denver, Atlanta
Atlanta -> New York, Dallas, Chicago
Starting city: Seattle
Depth-first search:
Seattle
Boston
New York
Atlanta
Chicago
Denver
Dallas
Los Angeles
San Francisco
Portland
Breadth-first search:
Seattle
Portland
Boston
San Francisco
New York
Dallas
Denver
Atlanta
Los Angeles
Chicago
```

Q4:

Write codes in file *employee.cpp* and *employee.h*.

Of them, *employee.h* and *employee.cpp* should include the declaration of class Employee and its subclasses including *HourlyEmployee*, *CommissionedEmployee*, and *SalariedEmployee*, and their corresponding implementations.

So that if we run the test file "TestEmployeeClass.cpp", we can have the results like:

```
Bob Cratchit: 90
Mr Fezziwig: 150
Ebenezer Scrooge: 500
```

Assignment Submission

After you test all your questions, zip the whole project in one file named XXX.zip (XXX is your student ID and your name is not required) and then submit it to the BB system.

Marking scheme

For each question:

- **0.6 Marks will be given to students who have submitted the program on time.**
- **0.6 Marks will be given to students who wrote the program that meet all the requirements of the questions**
- **0.6 Marks will be given to students who programs that can be compiled without errors and warnings**
- **0.6 Marks will be given to students whose programs produce the correct output if their programs can be compiled.**
- **0.6 Marks will be given to students who demonstrate good programming habit and style.**

Q&A

1. **How to check whether the assignment is submitted on time?**

We directly check it through Blackboard system

2. **How to check whether the assignment can be compiled?**

   Note that we strongly suggest you to use the QT IDE since your code tested on other IDE may not work on QT. If you want to use other IDE, please test your code again on QT to avoid such errors. We also strongly suggest you to use the Standford Library if necessary instead of STL or other similar libraries.

   If the QT IDE generates any errors when building your project, 0.4/0.6 mark will be missed for each question. If any warnings are produced but your code can run, 0.4/0.6 mark will be missed for all 3 questions.

3. **How to check the assignment gives correct output?**

We will run your code if it can be compiled (warnings are tolerated). You test code should be correct and output is also correct. User-friendly test code and

output is preferred.

4. How to check the programming style?

- **Code layout.**

  Beautiful !!!!

  The following points should be pay attention to:

  a. The indentation used

  b. {} used for function, for structure, while structure, switch structure.

  c. meaningful variable name.

We strongly suggest you to adhere to the coding style used in our textbook, which will be followed by us to check your code.