COSC1076 | Advanced Programming Techniques

Tutorial/Lab | Week 07

Overview

The Week 06 tutorial/lab is to revise content from the last few weeks.

- ADTs
- Vector
- Linked Lists

Tutorial Questions

The following header file snippets (only the classes are listed) are the simple definition of a Linked List as discussed in the Week 06 lecture.

Node.h

```
class Node {
public:

   Node(int data, Node* next);
   Node(Node& other);

   int    data;
   Node*  next;
};
```

LinkedList.h

```
#include "Node.h"

class LinkedList {
public:
   LinkedList();
   ~LinkedList();

   int size();
   void clear();
   int get(int i);

   void addFront(int data);
   void addBack(int data);

private:
   Node* head;
};
```

Consider the following questions:

1. Describe a `vector` and compare it to a primitive array.
2. Use the linked list ADT to answer the following questions. (That is, you should not modify the linked list itself)
   (a) Write a code snippet to create a new Linked List
   (b) Write additional code to add three integers to the Linked List
   (c) Write additional code to print out the contents of the Linked List in order
3. For questions 2a and 2b, draw the picture of the Linked List after every step.

4. For question 2c, discuss the efficiency of the implementation.
5. Consider the following questions about the Linked List ADT.
   (a) Why is the `int` type used for the `get`, `addFront`, and `addBack` methods?
   (b) Why is the `Node` class *not* used in the public methods of the `LinkedList` class?
   (c) Discuss the design aspects of making the fields of the `Node` class public, rather than private.

Group Registration for Assignment 2

Assignment 2 will be conducted in groups of 4. These groups ***must*** be formed only within your lab. This is because progress updates for assignment 2 will be presented to your tutor during lab times.

Groups for Assignment 2 must be ***confirmed*** with your tutor by the ***your week 7 lab***.

If you have formed your group, inform your tutor.

Assignment 2 Progress Update

In this update you will need to have a brief discussion with your tutor about:

- How your group will manage their codebase. That is, will you have a git repository?
- How will your group co-ordinate/discuss issues?
- Does your group have a planning schedule of when to get work done?
- How will your group split up tasks?

Lab Questions

It is a good idea to attempt the lab questions before coming to class. The lab might also be longer than you can complete in 2 hours. It is a good to finish the lab at home.

You should demonstrate your work to your tutor.

Exercises

1. Write a C++ program that:
   (a) Creates a C++ STL Vector of integers
   (b) Fills the vector with 100 integers from 0 to 99
   (c) Prints out the contents of the vector as a comma separated list on a single line

2. Using the Linked List ADT provided in the tutorial questions, implement all of the methods. *Do not modify the header files.* That is implement the:
   (a) Constructor
   (b) Deconstructor
   (c) `size`
   (d) `clear`
   (e) `get`
   (f) `addFront`
   (g) `addBack`

   If you are having trouble with this, many of these methods were discussed in the lecture, so review the Echo360 recording.

3. The implementation of the size method takes *linear time*, because it must go through the entire linked list. A more efficient method would be to keep track of the size of the list in the ADT.
   Modify the linked list ADT by:
   (a) Add a field to the `LinkedList` to store the size of the list
   (b) Modify the methods to correctly keep track of the size. You will need to modify, the constructor, the deconstructor, `size`, `clear`, `addFront`, and `addBack`.
   (c) What is the efficiency of your modified implementation?

4. (Extension) It is also possible to made the `addBack` method more efficient, by keeping a pointer to the last node in the linked list.
   Modify the linked list ADT by:
   (a) Add a field to the `LinkedList` to store the end of the list
   (b) Modify the methods to correctly keep track of the end of the list. You will need to modify, the constructor, the deconstructor, `size`, `clear`, `addFront`, and `addBack`.
   (c) What is the efficiency of your modified implementation?