COSC1076 | Advanced Programming Techniques

Tutorial/Lab | Week 10

Overview

The Week 10 tutorial/lab is to revise content from the last few weeks.

- Linked Lists
- Recursion
- Binary Search Tree

Tutorial Questions

The tutorial questions relate to the following C++ program:

LinkedList.h

```
class LinkedList {
public:

   LinkedList();
   ~LinkedList();

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

    // ... more methods (see previous labs)

private:
   Node* head;

   int size(Node* node);
};
```

LinkedList.cpp

```
int LinkedList::size(Node* node) {
   int retVal = 0;
   if (node == nullptr) {
      retVal = 0;
   } else {
      retVal = 1 + size(node->next);
   }
   return retVal;
}

int LinkedList::size() {
   return size(head);
}
```

1. When implementing a recursive function, what are the two key "components" to find?
2. Describe how the recursive implementation of the `size` method works. Use a *static reasoning* perspective.
3. Implement a recursive version of the `get` method.
4. Describe and illustrate the properties of a Binary Search Tree.

Assignment 2 Progress Update

Update your tutor on the progress of your group. You might consider such things like: ***The assignment is due NEXT WEEK (week 11)!*** For Thursday and Friday labs, this will be the last update before the assignment is due. However, there will still be an update next week for what your group did to get the assignment completed.

- ***What has each individual done in the past weeks?***
- ***What is the goal of each individual in the group for the coming week?***
- Do you have a design of the classes, data structures, and functions for the assignment?
- Have you completed Milestone 2?
- What will your group do for Milestone 3?
- Make sure you remember Milestone 4!

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. The below ADT defines a Linked List, as we have been using in the past few weeks.

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);

    void deleteFront();
    void deleteBack();

    void addAt(int i);
    void deleteAt(int i);

private:
    Node* head;
```

```
};
```

Implement ***recursive*** version of the methods:
(a) `clear`
(b) `addBack`
(c) `deleteBack`

2. The following ADT defines a binary search tree, that makes use of shared pointers. Implement the methods. You may use recursion if you wish:
(a) `clear`
(b) `contains`
(c) `add`

BST_Node.h

```
#include <memory>

class BST_Node {
public:

   BST_Node(int data);
   BST_Node(const BST_Node& other);
   ~BST_Node();

   int   data;

   std::shared_ptr<BST_Node> left;
   std::shared_ptr<BST_Node> right;
};
```

BST.h

```
#include "BST_Node.h"

#include <memory>

class BST {
public:

   BST();
   ~BST();

   void clear();
   bool contains(const int data) const;
   void add(const int data);

private:
   std::shared_ptr<BST_Node> root;
};
```