Creative Software Programming Assignment#6 (week-6)

Every assignment will be announced on **Thursday** and should be sumitted by next **Tuesday**.

In this week **Handed out will be Oct 8, 2020, Due Oct 13, 2020**

1. Linked List (85%)

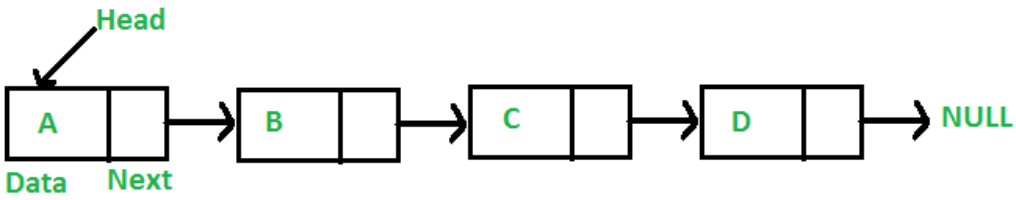
We are going to create a simple linked list data structure. First, check the node structure by referring to `node.h`. Also in `list.h` there is a simple skeleton code written.

A linked list refers to nodes in a connected form, where each node points to the next node.

The List class has a special node called **head**. The next of head node will point to the first element in the list. The next of the first element of the list points to the second element of the list, and in this way we can iterate through the list repeatedly until nullptr(0x0) appears.

Assignment Structure

- week-6
  - node.h
  - list.h
  - main.cc
  - CMakeLists.txt




```
#include <string>
#include <functional>

#include "node.h"

template <typename T>
class List {
public:
    List() : count(0) {
        head = new Node<T>(0, nullptr);
    }
    ~List() {
        // TODO: write your code here
        // release remain nodes before delete head node
        delete head;
    }

    void push_front(T value) {
        // TODO: write your code here
```

```
        // create new node with value
        // and add node to front of list
    }
    void push_back(T value) {
        // TODO: write your code here
        // create new node with value
        // and add node to back of list
    }
    T pop_front() {
        // TODO: write your code here
        // remove front node(not head)
        // and return its value
        // if try to remove head node return 0
    }
    T pop_back() {
        // TODO: write your code here
        // remove back node(not head)
        // and return its value
        // if try to remove head node return 0
    }

    size_t size() {
        // TODO: write your code here
        // return current items in list (except head)
    }

    void traverse(const std::function<void(const Node<T>&)>& f) {
        for (Node<T>* node = head->next; node != nullptr; node = node->next) {
            f(*node);
        }
    }

private:
    Node<T>* head;
    size_t count;

    // OPTIONAL: you can write helper functions
};
```

2. Double Linked List (15%)

The list implemented in Problem 1. is called a single linked list where each node points only to the next node. If each node has not only the next but also the previous node, prev, it is a double linked list. Use the node structure below to implement a double linked list.

You can start by copying the previously created list.h file to double_linked_list.h. See below for double_linked_node.h.

Assignment Structure

- week-6
  - double_linked_node.h
  - double_linked_list.h

```cpp
template <typename T>
class Node {
public:
    Node() {}
    Node(T value) : value(value) {}
    Node(T value, Node* next, Node* prev) : value(value), next(next), prev(prev)
{}

    T value;
    Node* next;
    Node* prev;
};
```

```cpp
#include <string>
#include <functional>

#include "double_linked_node.h"

template <typename T>
class List {
public:
    List() : count(0) {
        head = new Node<T>(0, nullptr, nullptr);
        tail = new Node<T>(0, nullptr, nullptr);

        head->next = tail;
        tail->prev = head;
    }
    ~List() {
        // TODO: write your code here
        // release remain nodes before delete head node
        delete head;
        delete tail;
    }

    // ...

private:
    Node<T>* head;
    Node<T>* tail;
    size_t count;

    // OPTIONAL: you can write helper functions
};
```