CSC 122 001 Computer Science II
Julius Ranoa

Midterm – Practical Exam.
Objectives:

1. Create a template linked list with methods to add/remove/print the list.
2. Modify the above linked list to include a member function to insert a new item at a specified position. A position of 0 means that the value will become the first item on the list, a position of 1 means that the value will become the second item on the list, and so on. A position equal to or greater than the length of the list means that the value is placed at the end of the list. Call this member function *"insertAtPos"*.
3. Demonstrate the usage of the above linked list using strings and char from *main.cpp*.

Screenshot of runtime:

```
Now testing with characters...
New linked list with 7 members: a b c d e f g
Removed first three members: d e f g
The character most recently removed is: c
Linked list after insertion to index 0: 1 d e f g
Linked list after insertion to index 3: 1 d e 4 f g
Linked list after insertion to index 100: 1 d e 4 f g X

Now testing with strings...
New linked list with 7 members: Ant Bird Cat Dog Eagle Fox Goat
Removed first three members: Dog Eagle Fox Goat
The string most recently removed is: Cat
Linked list after insertion to index 0: ONE Dog Eagle Fox Goat
Linked list after insertion to index 3: ONE Dog Eagle FOUR Fox Goat
Linked list after insertion to index 100: ONE Dog Eagle FOUR Fox Goat LAST

Done.
```

Files included: (1) main.cpp, (2) LinkedList.h

main.cpp

```cpp
#include <iostream>
#include <string>
#include "LinkedList.h"

void testWithChars();
void testWithStrings();

int main() {
    std::cout << "Now testing with characters... \n";
    testWithChars();
    std::cout << "\n";

    std::cout << "Now testing with strings... \n";
    testWithStrings();
    std::cout << "\n";

    std::cout << "Done. ";
    return 0;
}

void testWithChars() {
    LinkedList<char> characterList;
    // Make a linked list of 7 characters.
    char charTest[] = { 'g', 'f', 'e', 'd', 'c', 'b', 'a' };
    for (char c : charTest) {
        characterList.addItem(c);
    }
    std::cout << "New linked list with 7 members: ";
    characterList.print();
    std::cout << "\n";

    // Remove the first three characters.
    std::cout << "Removed first three members: ";
    for (int i = 0; i < 2; i++) characterList.popItem();
    char charLastPopped = characterList.popItem();
    characterList.print();
    std::cout << "\n";
    std::cout << "The character most recently removed is: " << charLastPopped << "\n";

    // Insert at Position Tests
    characterList.insertAtPos('1', 0);
    std::cout << "Linked list after insertion to index 0: ";
    characterList.print();
    std::cout << "\n";

    characterList.insertAtPos('4', 3);
    std::cout << "Linked list after insertion to index 3: ";
    characterList.print();
    std::cout << "\n";

    characterList.insertAtPos('X', 100);
    std::cout << "Linked list after insertion to index 100: ";
    characterList.print();
    std::cout << "\n";
```

```
}

void testWithStrings() {
    LinkedList<std::string> characterList;
    // Make a linked list of 7 strings.
    std::string stringTest[] = { "Goat", "Fox", "Eagle", "Dog", "Cat", "Bird", "Ant" };
    for (std::string s : stringTest) {
        characterList.addItem(s);
    }
    std::cout << "New linked list with 7 members: ";
    characterList.print();
    std::cout << "\n";

    // Remove the first three characters.
    std::cout << "Removed first three members: ";
    for (int i = 0; i < 2; i++) characterList.popItem();
    std::string lastPopped = characterList.popItem();
    characterList.print();
    std::cout << "\n";
    std::cout << "The string most recently removed is: " << lastPopped << "\n";

    // Insert at Position Tests
    characterList.insertAtPos("ONE", 0);
    std::cout << "Linked list after insertion to index 0: ";
    characterList.print();
    std::cout << "\n";

    characterList.insertAtPos("FOUR", 3);
    std::cout << "Linked list after insertion to index 3: ";
    characterList.print();
    std::cout << "\n";

    characterList.insertAtPos("LAST", 100);
    std::cout << "Linked list after insertion to index 100: ";
    characterList.print();
    std::cout << "\n";
}
```

LinkedList.h

```
//
// Created by TheLoneWoof on 3/8/18.
//

#ifndef EXAM_MIDTERM_LINKEDLIST_H
#define EXAM_MIDTERM_LINKEDLIST_H

#include <iostream>

// TEMPLATE DEFINITION

template <class T>
class LinkedList {

private:
    struct ListItem {
        T value;
        ListItem * next;
        ListItem(T val, ListItem * ptr) {
            value = val;
            next = ptr;
        }
    };
    ListItem * firstItem;

public:
    LinkedList() {
        firstItem = nullptr;
    }

    // Objective 1. Add, Remove, and Print.
    void addItem(T val);
    T popItem();
    void print();

    // Objective 2. Insert at position method.
    void insertAtPos(T, unsigned);
};

// IMPLEMENTATION

template <class T>
void LinkedList<T>::addItem(T val) {
    firstItem = new ListItem(val, firstItem);
}

template <class T>
T LinkedList<T>::popItem() {
    // Pop the item in front of the list.
    ListItem * tempReference;
    T poppedItem;

    // Make duplicates of the first item.
    tempReference = firstItem;
```

```
    poppedItem = firstItem->value;

    // Fix the list.
    firstItem = firstItem->next;
    tempReference->next = nullptr;
    delete tempReference;

    return poppedItem;
}

template <class T>
void LinkedList<T>::print() {
    ListItem * temp;
    temp = firstItem;
    while (temp) {
        std::cout << temp->value << " ";
        temp = temp->next;
    }
}

template <class T>
void LinkedList<T>::insertAtPos(T item, unsigned index) {
    if (index == 0) {
        // If index == 0, then add item to the beginning.
        addItem(item);
    } else {
        // If index > 1, then insert item between lists.
        ListItem * prevItem, * currItem, * newItem;
        int i = 1;

        prevItem = firstItem;
        currItem = firstItem->next;
        // Loop through i or until last of list.
        while (i < index && currItem) {
            i++;
            prevItem = currItem;
            currItem = currItem->next;
        }

        // Insert item.
        newItem = new ListItem(item, currItem);
        prevItem->next = newItem;

        // Dereference the pointers.
        newItem = nullptr;
        prevItem = nullptr;
        currItem = nullptr;
    }
}


#endif //EXAM_MIDTERM_LINKEDLIST_H
```