CSCI 2270: Data Structures - Midterm I
Part II (Coding Problems)

Spring 2021: Friday February 26, 5 PM - 8 PM

Instructions

1. This portion of the exam has two coding questions. Each question is worth 30 points.
2. You are provided with starter code for both questions. For the first question, you need to design an appropriate function prototype (within the provided starter code). For the second question, the prototypes are provided.
3. Make sure to use the C++11 standard for compilation (`g++ -std=c++11`). If you have doubts or issues with your local system, compile in `coding.csel.io`.
4. You need to write code for the function asked in the question. You can write helper functions if desired. You do not have to complete the entire class implementation. For example, if the question asks to sum all the nodes in a Linked list, you should only write the code for that function. You do not have to be concerned about other class member functions, such as insertion or list creation.
5. Read the specification given in the questions. Your code should be well-written in terms of commenting and indentation. Good coding includes (but not limited to):
   - meaningful names of the functions and variables,
   - proper indentations, and
   - legible comments.
6. Describe your logic clearly in coding comments. In case your implementation is not fully correct, your comments may be helpful to get you some partial credit.
7. You are required to upload a single zip file with the two sub-directories (folders): `Q1` and `Q2`. Include all of the relevant code (including any starter code) for question 1 and question 2 to the folders `Q1` and `Q2`, respectively. When finished, submit the **combined** zip file (that has both aforementioned folders) to the specified Canvas submission link.
8. In addition to being tested on several test-cases, the grading staff will also visually inspect your code for correctness, partial correctness, and memory leaks.

1. $\boxed{\text{30 points}}$ You are required to write a function `trim` that takes three arguments: an integer array, its length, and a target value; and returns `void`. Your algorithm is supposed to modify the array (and its length) by removing all of the elements of the array from (and including) the first occurrence of the target value. If the target value is not present in the array then this function keeps the array unchanged (i.e. nothing happens). Your function must deallocate (free) any heap memory that is no longer required.

Example A.

```
Input array = {3, 2, 5, 1, 0, 8, 0, 4}
Array length = 8
Target value = 0
// Call to function trim
Array after trimming = {3, 2, 5, 1}
Array length = 4
```

Example B.

```
Input array = {3, 2, 5, 1, 0, 8, 0, 4}
Array length = 8
Target value = 33
// Call to function trim
Array after trimming = {3, 2, 5, 1, 0, 8, 0, 4}
Array length = 8
```

Notes. You must implement your code in a single file named `Q1.cpp` and your code must compile with the following command in a standard Linux or Mac terminal.

```
g++ -std=c++11 Q1.cpp
```

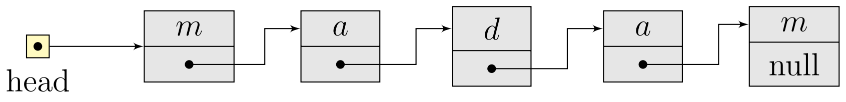
2. $\boxed{\text{30 points}}$ You are given a *singly linked list* class `SLL` (partial) implementation where each node consists of a single key value of `char` type. You are required to implement two member functions `char atIndex(int i)` and `bool palindrome()`.

   1. The function `char atIndex(int i)` returns the character at the index `i` of the linked list. Note that we, the computer scientists, count from 0. Hence, the head of the list is the 0 index node, the next element is the 1 index node, and so on. Your function should terminate gracefully by printing an error message and returning the null character `'\0'` if *i* is not a valid index (e.g. `i` is either negative or greater than or equal to the length of the linked list). You can safely assume that the null character `'\0'` is not a member of the linked list.
   2. The function `bool palindrome()` returns true if the sequence of characters of the linked list is a *palindrome*. Recall that a palindrome is a sequence of characters which reads the same backward as forward, such as tacocat or racecar. For the purpose of this question, we assume that empty lists are palindromes, and so are lists with only one node. **Hint: The `atIndex` function.**

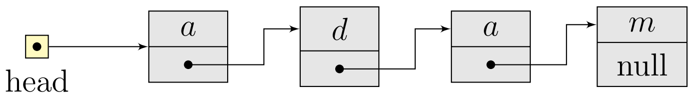
Consider the following definition of the the linked list class:

```
struct Node{
    char key; /* Key of the node */
    Node *next; /* Pointer to the next node in the list */
};
class SLL {
    private:
        Node* head;
    public:
        /* implemented in the starter code */
        SLL(); // constructor
        ~SLL(); // destructor

        void displayList();
        void insert(Node* afterMe, char newValue);

        /* To be implemented by you */
        bool palindrome();
        char atIndex(int i);
};
```

Example A. Let `ll` be an object of the `SLL` class. If `ll` has the following configuration:



Then, `ll.palindrome()` should return `true`, while `ll.atIndex(0)` and `ll.atIndex(4)` should both return `m`. On the other hand, `ll.atIndex(-1)` and `ll.atIndex(8)` should return the null character (`'\0'`) and print an error message. Similarly, if `ll` has the following configuration:



then `ll.palindrome()` should return `false`.

Within your function, you are allowed to construct a local linked list if it helps you to solve the problem. However, you must deallocate (free) any heap memory that is no longer required. You are not allowed to use an array, a `vector`, or any of the `string` library functions.

The code must compile with the following in a standard Linux or Mac terminal.

```
g++ -std=c++11 SLL.cpp Q2driver.cpp
```

If you have doubts or issues with your local system, compile in `coding.csel.io`.