McMaster University
MECHTRON 2MD3: Data Structures and Algorithms for Mechatronics

Winter 2023
Assignment 2
Due: Feb. 8, 2023 at 23:59

Instructions
All questions annotated "short answer" should be typed and submitted in a single pdf document with the file name "1234567-asg1.pdf" where 1234567 is your student ID. Each question annotated "programming" should be submitted as a single C++ source file named "1234567-asg1-x.cpp" where 1234567 is your student ID and x is the question number. All code in programs and short answers should be formatted and commented. Please ensure your source files compile and run properly before submitting.

Questions

1. [short answer, 3 marks] Describe in detail how to swap two nodes *x* and *y* (and not just their contents) in a singly linked list *L* given references only to *x* and *y*. You can assume that you also have a reference to the head of the singly linked list, and to the header and trailer for the doubly linked list. No assumptions can be made about the relative location of nodes x and y in the list. Repeat this exercise for the case when *L* is a doubly linked list. Which algorithm takes more time?

2. [short answer, 3 marks]: Draw the recursion trace for the execution of function ReverseArray(A,0,4) (Code Fragment 3.39 in Goodrich text) on array A={4,3,6,2,5}.

3. [programming, 3 marks] Write a tail recursive C++ function that finds both the minimum and maximum values in a c++ vector of int values without using any loops. Include a main method that tests your function by creating the vector A={1, -2, 0, 6, 66, -7} and printing the min and max values on a single line like this:

   ```
   Min -7 Max 66
   ```

4. [programming, 10 marks] Modify lecture_demos_ch03/singly_linked_list_demo.cpp in the following ways:

   a. Add a recursive function to the SLinkedList class that counts and returns the number of nodes.

   b. Add a function to the SinglyLinkedList class to print node elements in order. For a list with string element "one", "two", "three", your function should output a single line like this:

one->two->three

c. Add a function to the SinglyLinkedList class to append another singly linked list to the end of a list object. Since calling this function effectively combines the two lists, you must be careful to avoid errors when both lists fall out of scope and have their destructors called.

d. Add a recursive function to the SinglyLinkedList class to reverse the list. Do this by manipulating pointers, not by copying node data.

e. Include a main method that tests all your new functions using two linked lists with string data. Your program should produce exactly this output:

   List 1 has 4 nodes:
   one->two->three->four
   List 2 has 3 nodes:
   five->six->seven
   After appending list 2 to list 1, list 1 has 7 nodes:
   one->two->three->four->five->six->seven
   After reversing list 1, it looks like this:
   seven->six->five->four->three->two->one

5. [programming, 16 marks] Modify DLinkedList class from lecture_demos_ch03/doubly_linked_list_demo.cpp to store game score entries similar to Section 3.1.1 of the Goodrich textbook. Each node must store a string and integer for the name and score of each game score entry. The list must maintain data in sorted order from best to worst score. Add the following methods:

   a. int DLinkedList::Size() which recursively counts the number of scores stored in the list.

   b. void DLinkedList::AddScoreInOrder(std::string name, int score) which adds a game score entry.

   c. void DLinkedList:: RemoveScore(int index) which removes the i^{th} score from the list. Indexing should be such that 0 refers to the best score. Do this as efficiently as possible.

   d. bool DLinkedList:: UpdateScore(std::string name, int score) which updates the score associated with the given name. This function should return true if the name was found and updated, and false otherwise.

   e. void DLinkedList:: Print() which recursively prints the scores from best to worst in the following format: {name,score}->{name,score}->{name,score}…

f. Copy constructor.

g. Overload the assignment (=) operator

h. void DLinkedList::OrderByName() which changes the order of the list to alphabetical by name.

Your modified DLinkedList class will be tested with the following main method:

```
int main() {
  DLinkedList scores;
  scores.AddScoreInOrder("Jeff", 7);
  scores.AddScoreInOrder("Jen", 9);
  scores.AddScoreInOrder("Ilya", 3);
  scores.AddScoreInOrder("Sara", 10);
  scores.AddScoreInOrder("Sam", 11);
  // Test size function
  cout << "Number of scores is " << scores.Size() << endl;
  scores.Print();
  // Test remove function
  scores.RemoveScore(3);
  cout << "Number of scores is now " << scores.Size() << endl;
  scores.Print();
  // Test update function
  if (scores.UpdateScore("Jeff",6))
    scores.Print();
  // Test copy construcor
  DLinkedList scores_copy_1(scores);
  scores.UpdateScore("Jen",5);
  scores.Print();
  scores_copy_1.Print();
  // Test assignment operator overload
  DLinkedList scores_copy_2 = scores_copy_1;
  scores_copy_1.UpdateScore("Jen",5);
  scores_copy_1.Print();
  scores_copy_2.Print();
  // Test OrderByName function
  scores_copy_2.OrderByName();
  scores_copy_2.Print();
}
```

6. [programming, 6 marks] You are given a singly linked list of integer nodes **sorted** in increasing order. Write a program that represents the linked list with a 2-dimensional array. Your program should output the minimum integer which is greater than or equal to *x*.

The **input** to your program is as follows: The first line should receive integers *n*, *first,* and *x* which represent the number of elements in the list, the index of the first element, and the integer *x*. The next *n* lines of input each give $value_i$ and $next_i$ which represent a node's value and where it points (indices start from 1). If $next_i = -1$ then it is the tail of the list and does not point to any other cell. Your program should **output** the minimum value that is greater than or equal to *x*, or *-1* if there is no such number.

Example

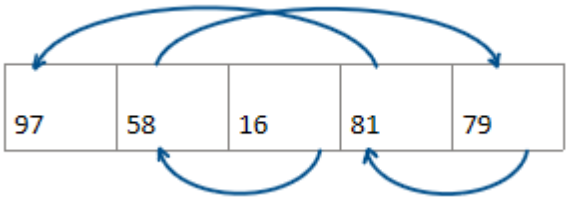
Input:

```
5 3 80
97 -1
58 5
16 2
81 1
79 4
```

Output:

```
81
```

Hint: You do not need to know how to sort. The linked list associated with the example input looks like this:



The End.