CSE 203 - Data Structure

Midterm Exam

Spring 2020 Time: 2 hours

Name:
Id:
Section:

1. (a). **[5 Points]** Write a function to check whether the numbers stored in a one dimensional array of size n are organized in a palindrome order not. For example,
The following array will return **true,**

1	2	3	3	2	1

false for following array

1	2	3	3	2	6

Function Signature:

```
        bool isPalindrome (int * array){

                //write the code
            }
```

(b). **[20 Points]** Suppose, you have a memory of size 128 KB (128X 1024 = 131072 Bytes) and we know that an integer variable (int) only takes 4 bytes of memory.
Now,

i. Write a program that will take 32000 numbers randomly and store them into an array **A.** After storing all the numbers into the array print the square of each number.

ii. Then, in the same program, you need to take 30000 random numbers and store them into a different array **B.** After storing all the numbers into the array B, print their cubes of each number.

2. **[25 Points]** Given two singly linked lists of any size, the task is to create a new linked list using those linked lists. The condition is that the greater node among the i th nodes of both linked lists will be added to the new linked list. For example, if the two linked lists are:

Head1: 5->2->3->8->9
Head2: 1->7->4->5

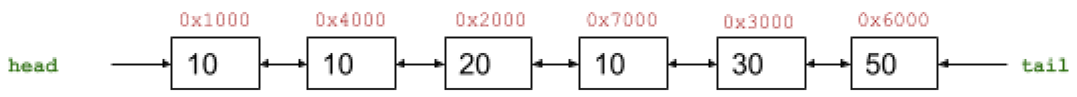
Then 1st node of Head1 is compared with 1st node of Head2, 5 is bigger so it will added to the new list. Then 2nd node value of Head1 is compared with 2nd node value of Head2, here 7 is bigger so it will be added to the new list and so on. If at any moment, one list finishes then the remaining items of the list will be added to the new list as it is. So after the execution of your function the new list would look like this:

newList: 5->7->4->8->9

Write function node* merge(node* A, node* B) that will create a new singly linked list with a new head using A and B using the above condition. Then return the head pointer of the new list to the calling function.

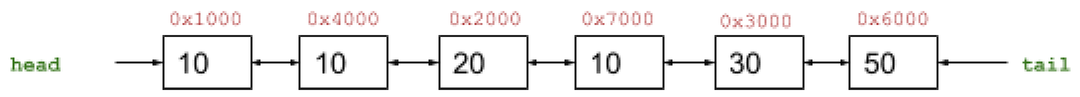
```
node* merge(node* A, node* B)
{
     //Your Answer
}

int main()
{
     node* Head1, Head2;
     //CodeRandomly creating two linked list with Head1, and
     //Head2.
     //The lengths might not be equal
     node* newList = merge(Head1, Head2);
}
```

3. **(a) [5 Points]** Given the *head and *tail pointers of the doubly linked list and a value x, write a function which will return the address of the node whose data value is equals to x. If multiple

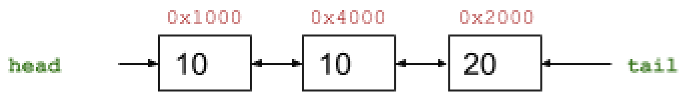


occurrences of x are found then your function should return the last occurance of the node with value x. For example, if your link list looks like the one below:
If the value 10 is given as a value x then the address 0x7000 should be returned from your function.

(b) [20 Points] Given a doubly linked list with pointers *head and *tail and a value x, delete all the nodes starting from the last occurrence of the value x till the end of the list (including the node with value x). For example, if the linked list looks like the one below:



If the value 10 is given then your function to delete the last three nodes (in this example only, your function should work in any given linked list flawlessly). After the execution of your function, the above list should look like this:



If the value is found only at the last position then only that node will be deleted. On the other hand, if the value appeared only at the first position then the whole list will be deleted.

4. **[25 Points]** Write a function that will verify whether a string is in $(01)^n(10)^n$. You are only allowed to use only a stack. You cannot use any array, queue, linkedlist.

   Hints: Use STL (stack <char> st) to use stack operations.
   For example,

Sample Input	**Sample Output**
010101101010	true
0101101010	false
010110011010	false

Function Signature:

```
bool is01n10n(string str){

        //write the code

}
```