[redacted]

[redacted]

[redacted]onal University of Computer and Emerging Sciences, Lahore Campus

Course:	Data Structures	Course Code:	CS 2001
Program:	BS(CS, DS, SE, R)	Semester:	Fall 2023
Duration:	90 Minutes	Total Marks:	
Paper Date:	2-Oct- 2023	Page(s):	
Section:	ALL	Section:	
Exam:	Midterm Exam 1	Roll No:	

[redacted]

[redacted]tion Notes: Answer in the space provided. You can use rough sheets, which **will not be marked.** Do not use pencil or red ink to answer the questions. In case of confusion or ambiguity, make a reasonable assumption. Questions are not allowed.

Ques[redacted]n 1: **(Marks: 5+5)**

a) Give an estimate of T(N) for each line of the following code. Also, give time complexity (in Big-Oh notation). Compute the tight bounds.

```
void format(int num, int * mem, int & size)
{
        size = 0;
        while (num > 0) {
                mem[size++] = num % 2;
                num = num / 2;
        }
}
void print(int n) {
        int* arr = new int[n];
        int s = 0;
        for (int i = 1;i <= n;i++)
                format(i, arr, s);
                for (int j = s - 1;j >= 0;j--)
                        cout << arr[j];
                cout << endl;
        }
        delete[] arr;
}
```

T(n) = 1+1+3+2+N+1+1+1+1+
(N + N)xN-1 *log N
3N + logN ≥ N x (N-1)
1.5 =
= O(N^1 logN)
= num-1
= N-1
= O(N)

b) Dry run and give the output for the following code snippet.

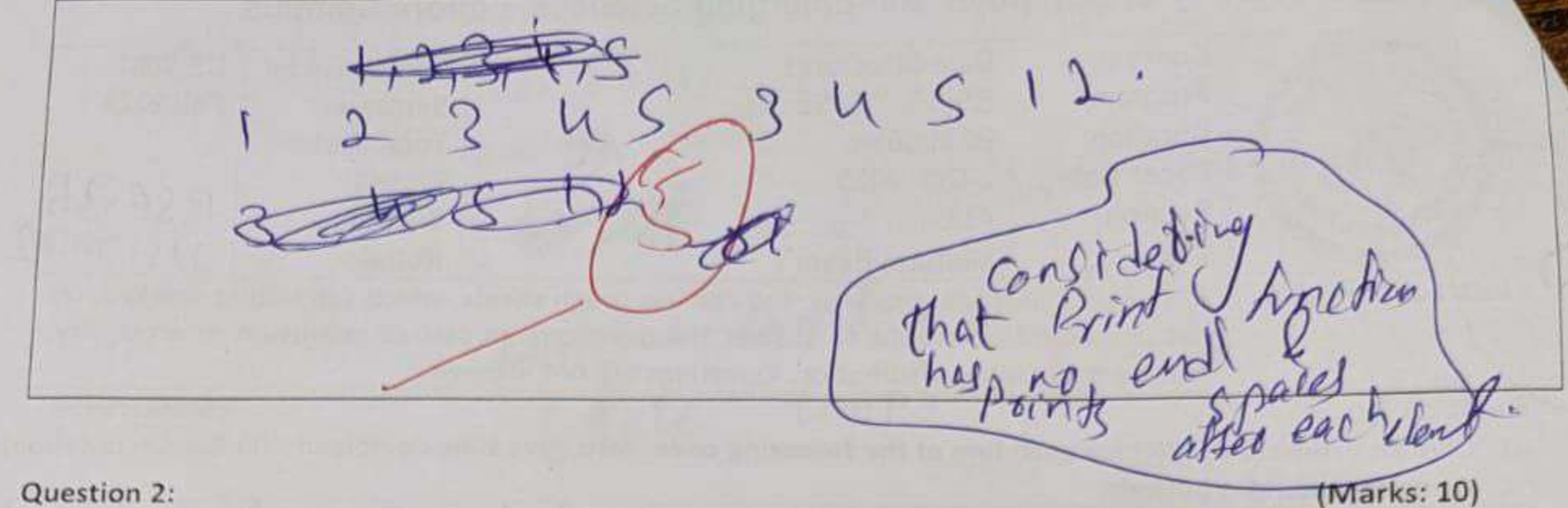
```
void LINKEDLIST:: whatsThisFor(int pos) {

if (pos == 0) return;
        Node* curr = head;
        Node* temp = curr;
        while (temp->next != NULL)
                temp = temp->next;
        temp->next = curr;
        curr->previous = temp;
        int count = 1;
        while (count <= pos) {
                head = head->next;
                temp = temp->next;
                count++;
        }
        temp->next = NULL;
        head->previous = NULL;
}
```

```
int main()
{
        LinkedList L1; // doubly linked list
        for (int i = 1;i <= 5;i++)
                L1.insertAtTail(i);
        L1.printList();
        L1.whatsThisFor(2);
        L1.printList();
}
```


[redacted]

Name:_____ Roll #: _____



Question 2: **(Marks: 10)**

Given a template-based doubly linked list class ***DLList,*** write the function ***RemoveSubList*** that takes a doubly linked list DLL1 as input and removes the first occurrence of DLL1 from the doubly linked list pointed by (*this). The function returns false if DLL1 does not exist in the doubly linked list.
Note that the function only removes if the entire list DLL1 is present in the doubly linked list.

```
void main() {
      DLList<int> DLL2, DLL1;
      int no,n, m;
      cin>>n>>m;
      for (int i = 0; i <n; i++){
           cin>>no;
           DLL2.insertStart(no);
      }
      for (int i = 0; i <m; i++){
           cin>>no;
           DLL1.insertStart(no);
      }
      DLL2. RemoveSubList(DLL1);
}
```

```
template<class T>
class DLList {
      public:
                ...
          bool RemoveSubList(DLList<T> DLL1)

      private:
           DLLNode *head, *tail;
};
```

Some examples to understand the working of the function bool RemoveSubList(DLList<T> DLL1)

If the list DLL2 is 2 3 3 3 5 **6 7** 8 and the input list DLL1 is **6 7,**
then RemoveSubList returns true, and DLL2 becomes **2 3 3 3 5 8.**

If the list DLL2 is 2 3 3 3 5 6 7 8 and the input list DLL1 is **5 6 1,**
then RemoveSubList returns false and ***makes no change to list DLL2*** as *complete list DLL1 5 6 1* is not found.

If the list DLL2 is 2 3 3 **3 5 6** 7 8 and the input list DLL1 is **3 5 6,**
then RemoveSubList returns true, and list DLL2 becomes **2 3 3 7 8.**

You can assume that class DLList is implemented with dummy (sentinel) nodes. **If you use any function (search or delete) of the DLList class, provide its code.**

Note: Code should be efficient (space and time), well structured and properly commented.

0635

Name: _____ Roll #: _____

```
bool RemoveSublist (DLList<T> DLL1)
{
  if ( DLL1.Head == NULL || this->Head == NULL)
  {
    return false;
  }
  bool flag = false;
  DLLNode<T>* del = NULL;
  DLLNode<T>* temp1 = DLL1.Head;
  DLLNode<T>* temp2 = this->Head;
  while ( temp1->next != NULL || temp2->next != NULL)
  {
    if ( temp2->data != temp1->data )
    {
      temp2 = temp2->next;
    }
    if ( temp2->data == temp1->data)
    {
      del = temp2;
      temp2 = temp2->next;
      temp1 = temp1->next;
      while( temp2->data == temp1->data)
      {
        if (temp2->next != NULL)
        {
          temp2 = temp2->next;
        }
        if (temp1->next != NULL)
        {
          temp1 = temp1->next;
        }
      }
      if ( temp1 != DLL1.Tail)
      {
        temp1 = DLL1.Head;
        del = NULL;
      }
      else {
        del = del->prev;
```

H→2→3→3→3→5→6→7→8

H→6→7→null

del to 3

t1

next = null

3