National University of Computer and Emerging Sciences

FAST School of Computing **Spring-2023** **Islamabad Campus**

CS-2001: Data Structures

Serial No:

Sessional Exam-II

Total Time: 1 Hour

Total Marks: 60

Monday, 10th April, 2023

Course Instructors

Amna Irum, Madiha Umar, Urooj Ghani, Rohail Gulbaz, Bilal Khalid

_____

Signature of Invigilator

Student Name	Roll No.	Course Section	Student Signature

DO NOT OPEN THE QUESTION BOOK OR START UNTIL INSTRUCTED.

Instructions:

1. Attempt on question paper. Attempt all of them. Read the question carefully, understand the question, and then attempt it.
2. No additional sheet will be provided for rough work. Use the back of the last page for rough work.
3. If you need more space write on the back side of the paper and clearly mark question and part number etc.
4. After asked to commence the exam, please verify that you have **11** different printed pages including this title page. There are a total of **4** questions.
5. Use permanent ink pens only. Any part done using soft pencil will not be marked and cannot be claimed for rechecking.

	Q-1	Q-2	Q-3	Q-4	Total
Marks Obtained					
Total Marks	**10**	**20**	**15**	**15**	**60**

Question 1 [1+8+1 Marks]

Convert the given string *A*(B/(C-D)+E)*(F^G)* to prefix notation by following the proper algorithm and show all steps.

a) Reverse String:

)G^F(*)E+)D-C(/B(*A
(G^F)*(E+(D-C)/B)*A

_____

b) Infix to Postfix:

Symbol	Postfix String	Stack
(		(
G	G	(
^	G	(^
F	GF	(^
)	GF^	
*	GF^	*
(	GF^	*(
E	GF^E	*(
+	GF^E	*(+
(	GF^E	*(+(
D	GF^ED	*(+(
-	GF^ED	*(+(-
C	GF^EDC	*(+(-
)	GF^EDC-	*(+
/	GF^EDC-	*(+/
B	GF^EDC-B	*(+/
)	GF^EDC-B/+	*
*	GF^EDC-B/+*	*
A	GF^EDC-B/+*A	*
	GF^EDC-B/+*A*	

c) Output string:

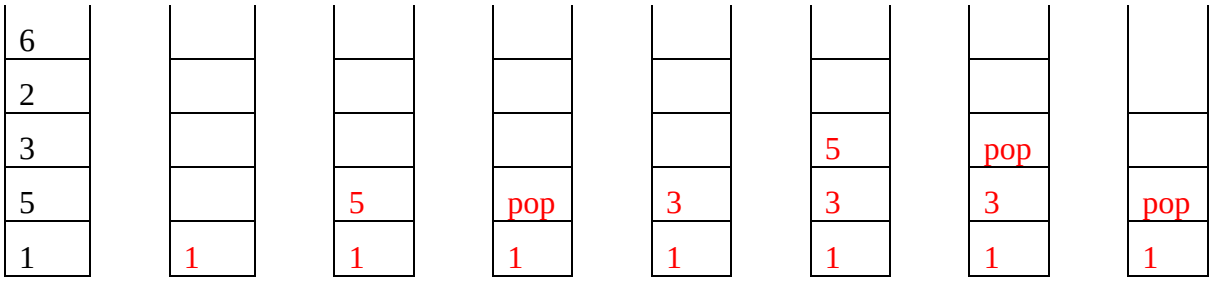
A+/B-CDE^FG

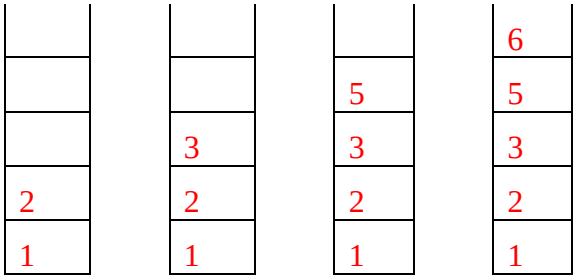
_____

Question 2 [20 Marks]

Given a Singly Linked List based Stack populated with random integers. You are required to sort the Stack in descending order using Recursion and show all traces of InsertWithSort() function after each call. Complete the code by doing compulsory rough work in the provided area.

Before





Line #	Code
1	struct stack{
2	int data;
3	struct stack *next;
4	};
5	class Sorting
6	{
7	public:
8	void push(struct stack **s, int x){
9	stack *p = new stack;
10	p->data = x;
11	p->next = *s;
12	____ *s = p;_____
13	}

```
14      int pop(struct stack **s){
15          int x=(*s)->data;
16          stack *temp=*s;
17          __ (*s) = (*s)->next; _____________
18          delete temp;
19          return x;
20      }

21      void InsertWithSort(struct stack **s, int x){
22          if(*s==NULL || __ x >= (*s)->data _________){
23              push(s, x);
24          }
25          else{
26              int temp = pop(s);
27              InsertWithSort(s, x);
28              __ push(s, temp); _____________
29          }
30      }

31      void sort(struct stack **s){
32          if(__*s!=NULL_____________){
33              int x = pop(s);
34              sort(s);
35              InsertWithSort(s, x);
36          }
37      }
38  };

39  int main(){
40      Sorting obj;
41      stack *top=NULL;
42      obj.push(&top, 1);
43      obj.push(&top, 5);
44      obj.push(&top, 3);
45      obj.push(&top, 2);
46      obj.push(&top, 6);
```

```
47      obj.sort(&top);
48      return 0;
49  }
```

Select the correct option for each blank.

Line #12

a) p=*s;

b) s = p;

c) *s = p;

d) **s=p;

e) None of the above

Line #17

a) *s = s->next;

b) s = s->next;

c) s->next=temp;

d) (*s) = (*s)->next;

e) None of the above

Line #22

a) x <= (*s)->data

b) x < (*s)->data

c) x >= (*s)->data

d) x <= s->data

e) None of the above

Line #28

a) push(s, x);

b) push(*s, temp);

c) push(**s, temp);

d) *s->data = x;

e) None of the above

Line #32

a) *s==NULL

b) *s!=NULL

c) s!=NULL

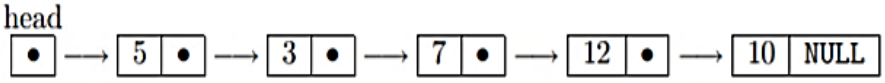
d) *s->data==NULL

e) None of the above

Rough Work:

Question 3 [5+5+5 Marks]

a) Consider the linked list of integers represented by the following diagram:



Draw a diagram of the above list after the following lines of code have been executed:

```
Node* prev = head->next;

Node* nodeToInsert = new Node;

nodeToInsert->item = 4;

nodeToInsert->next = prev->next;

prev->next = nodeToInsert;
```

Ans:

5->3->4->7->12->10->null

Rough Work:

b) The following function reverse() is supposed to reverse a singly linked list. There is one line missing at the end of the function, select the appropriate option.

```
void reverse( )
{
   Node* prev   = NULL;
   Node* current = head;  //assign head of the linkedlist to current pointer
   Node* next;
   while (current != NULL)
   {
      next= current->next;
      current->next = prev;
      prev = current;
      current = next;
     }

     /*ADD A STATEMENT HERE*/


}
```

Rough Work:

a) head = prev;
b) head = current;
c) head = next;
d) head = NULL;

c) Given an array-based queue.

44			33	66	88	99		

I. What are the index values of front and rear pointers.

Front=_____ Rear=_____

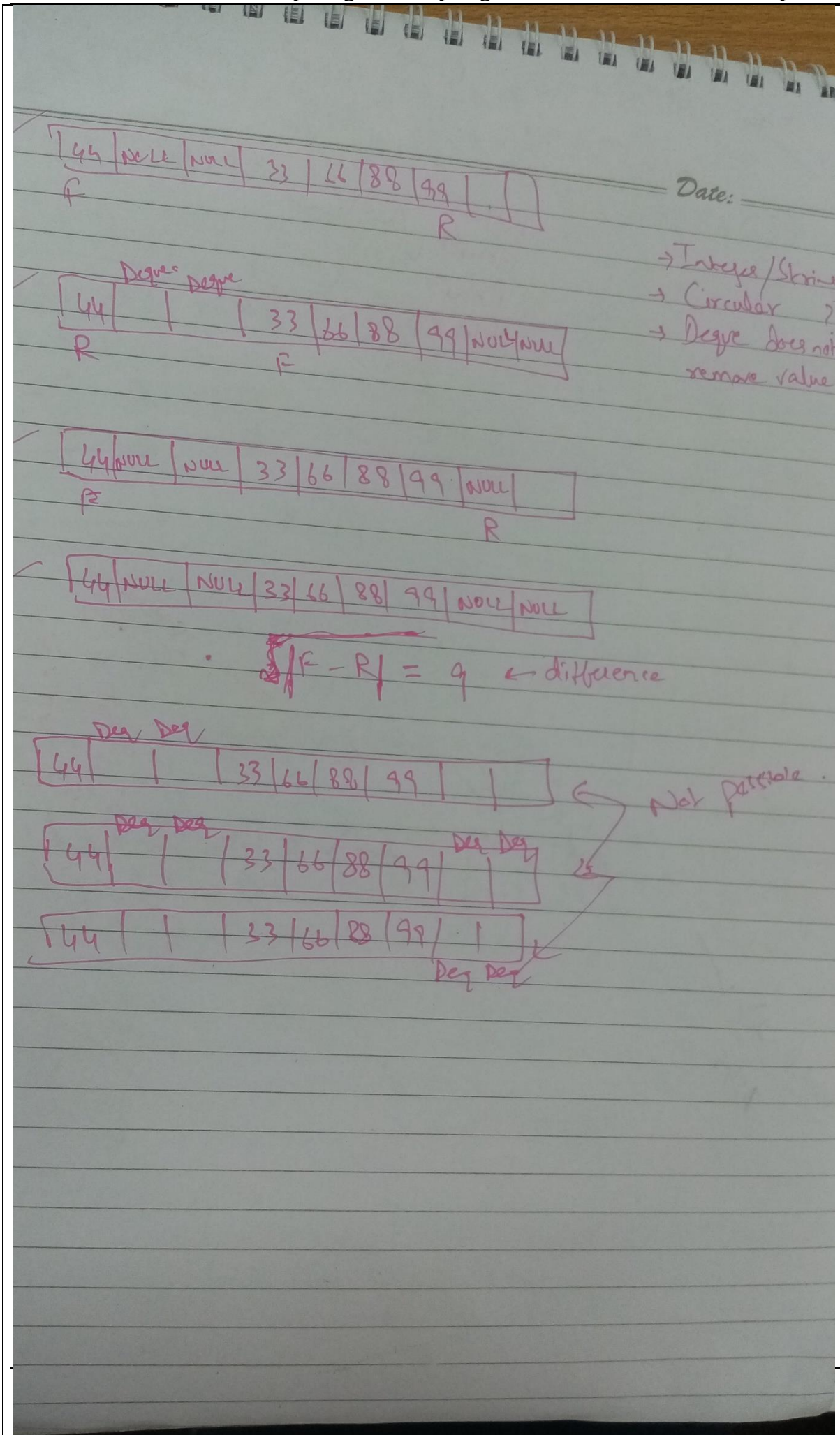
II. What are the index values after first dequeue operation is performed in the above-mentioned queue.

Front=_____ Rear=_____

III. What are the index values after second dequeue operation is performed in the above-mentioned queue.

Front=_____ Rear=_____

Rough Work:


Date:
→ Integer/String
→ Circular
→ Deque does not remove value
Deque Deque
F
R
|F - R| = 9 ← difference
Not possible.
Deq Deq

Question 4 [5+10 Marks]

a) What will be the linked list after the following function, if the original linked list is as follows.

6->13->11->6->13->11->5->4->5->10->NULL

Provide traces of each iteration

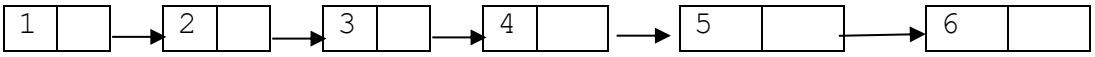
```
void function(struct Node* start)
{
	struct Node *ptr1, *ptr2, *nod;
	ptr1 = start;
	while (ptr1 != NULL && ptr1->next != NULL)
	{
		ptr2 = ptr1;
		while (ptr2->next != NULL)
		{
		    if (ptr1->data == ptr2->next->data)
		     {
				nod = ptr2->next;
				ptr2->next = ptr2->next->next;
				delete (nod);
		      }
		      else
				ptr2 = ptr2->next;
		}
		ptr1 = ptr1->next;
	}
}
```

Linked List after each iteration:

6->13->11->13->11->5->4->5->10->NULL
6->13->11->11->5->4->5->10->NULL
6->13->11->5->4->5->10->NULL
6->13->11->5->4->10->NULL
6->13->11->5->4->10->NULL
6->13->11->5->4->10->NULL

Rough Work:

b) Consider following LinkedList



Write a print function that prints a node and then skips two nodes to print another node. For example:

Nodes:	**Printing:**
NULL	Nothing
1->NULL	1
1->2	1
1->2->3	1
1->2->3->4	1, 4
1->2->3->4->5	1, 4

```
void print() {

node *ptr=head;

while(_____ptr!=NULL_____)
{
    cout<<ptr->data;

    if(__ptr->next==NULL_____ || __ptr->next->next==NULL_____)
    {
        ptr=___NULL_____
    }
    else
    {
```

```
        ptr= ___ptr->next->next->next;_________
    }
 }

}
```

Rough Work: