Name - Aniket Kumar
Registration no. - 12108348
Roll no. - B62

Q.No. - 1

Q1. write a code to reverse the elements of stack and then find the index of minimum valued number using array.

=>
```
#include <stdio.h>
#define TRUE 1
#define FALSE 0

// declaring the functions to be used.
void Push(int);
int Pop();
void PrintStack();
void reverse();
void MinElement();

// declare stack
int top = -1;
int arr_stack[20];
int SIZE = 0;

int main(){
    int temp; // for scanning the input
    // inserting elements in stack
    Printf("Enter the number of elements (not more than 20): ");
    Scanf("%d", &SIZE);
    Printf("Enter the elements of stack:\n");
    for(int i=0; i<SIZE; i++){
        Scanf("%d", &temp);
        Push(temp);
    }
    // Displaying the originally inserted value from stack
    Printf("Original stack\n");
    PrintStack();
```

```
    reverse(); // for reversing the stack elements.
    // displaying reversed stack
    Printf("\n Reversed Stack\n");
    PrintStack();
    // displaying minimum valued number's index
    MinElement();
    return 0;
}
// function definitions
// for checking if stack is full or not
int isfull(){
    if(top >= SIZE-1)
        return TRUE;
    else
        return FALSE;
}
// checks if stack is empty or not
int isEmpty(){
    if(top==-1)
        return TRUE;
    else
        return FALSE;
}
// Inserting an element to the stack and increment top index.
void Push(int num){
    if(isfull())
        Printf("stack is full...\n");
    else{
        top = top+1;
```

Name - Aniket Kumar
Registration no. - 12108348
Roll no. - B62

Q.No - 1

```
        arr_stack[top] = num;
    }
}

// removing top element from stack and derement top index.
int Pop(){
    if(isEmpty())
        Printf("Stack is Empty... \n");
    else{
        top = top-1;
        return arr_stack[top+1];
    }
}

// for displaying all the elements of stack
void PrintStack(){
    if(top == -1){
            Printf("Stack is Empty... \n");
    }
    else{
        for(int i=top; i>=0; i--){
            //start traversing from top element.
            Printf("%d ", arr_stack[i]);
        }
        Printf("\n");
    }
}

// Inserting element to the bottom of the stack
void InsertAtBottom(int item){
        if(isEmpty()){
                Push(item);
```

```
        else {
            // Store top most element in another variable and recursively
            // insert the rest of the stack using InsertAtBottom
            int top = Pop();
            InsertAtBottom(item);
            //finally insert the top element again
            Push(top);
        }
    }
// reversing the stack elements
void reverse(){
    if(!isEmpty()){
        // Keep Popping top element until stack is Empty.
        int top = Pop();
        reverse();
        // After removing, insert top element to the bottom
        InsertAtBottom(top);
    }
}
void MinElement(){
    int index = 0;
    for(int i=top; i>=0; i--){
        if(arr_stack[i] < arr_stack[index]){
            index = i;
        }
    }
    Printf("\n minimum element is Present at index %d and is
    %d(th) element from the top.\n", index, top-index+1);
}
```

Name - Aniket Kumar
Registration no. - 12108348
Roll no. B62

Q. No. - 1

Output:

Enter the number of elements (not more than 20): 5
Enter the elements of stack:
32 45 67 12 56
original stack
56 12 67 45 32

Reversed Stack
32 45 67 12 56

Minimum element is present at index 1 and is 4(th) element from the top.

Q2 write a code to delete the minimum valued item among N numbers from a queue.

⇒
```
#include <stdio.h>
#define MAX_SIZE 100
void insert();
void delete_min();
void display();
int arr_queue[MAX_SIZE]; // declaring queue
int front=0,rear=0;
int main(){
    int n;
    // intesting queue elements
    Printf("Enter the no of queue elements: ");
    Scanf("%d", &n);
    for(int i=0; i<n; i++){
        insert();
    }
    // deleting minimum valued item from queue and display
    // rest elements of the queue.
    delete_min();
    return 0;
}
//function for inserting elements into queue at the ~~front~~ rear.
void insert(){
    int item;
```

```
        if (rear == MAX_SIZE)
            Printf("\n ## Queue reached max!);
        else{
            Printf("\n Enter the value to be inserted : ");
            scanf("%d", &item);
            Printf("# Position : %d, Inserted value : %d", rear+1, item);
            arr_queue[rear++] = item;
        } // Insert element at rear and increment rear by 1.
}

// function to delete minimum valued item.
void delete_min(){
        if(front == rear)
            Printf("\n## Queue is Empty!");
        else {
            int min_element, index = 0;
            // displaying the queue before deleting.
            display();
            // finding the index of minimum valued item
            for(int i = front; i < rear; i++){
                if [arr_queue[i] < arr_queue[index]{
                    index = i;
                }
            }
            // displaying minimum valued item.
            min_element = arr_queue[index];
            Printf("\n Minimum element is %d.\n", min_element);
            Printf("deleting it..\n");
```

```
    // deleting the element
        for(int i= index; i<rear; i++){
            arr_queue[i] = arr_queue[i+1];
        }
        rear = rear-1; //decrement in rear after deleting.
        display(); // displaying after deletion.
    }
}

//function for displaying queue items.
void display(){
    Printf("\n ## Queue size: %d \n", rear);
    for (int i= front; i<rear; i++)
        Printf("%d ", arr_queue[i]);
}
```

Output:

```
Enter the no of queue elements: 5
Enter the value to be inserted: 32
# Position: 1 , Inserted value: 32
- - - - - .
- - - - - .

# Queue size: 5
 32 45 67 12 56
Minimum element is 12
Deleting it..

## Queue size: 4
 32 45 67 56
```

Q3. write a code to insert a node in queue using linked list.

⇒
```
#include <stdio.h>
#include <stdlib.h>

struct node{    // declaring queue using linked list.
      int data;
      struct node *next;
};
struct node *front;
struct node *queue;

void insert();
void display();
void main(){
      int choice;
      while(1){
       Printf("\n chose option");
       Printf("\n 1. Insert an element \n 2. display the queue
            \n 0. Exit \n");
       Printf("\n Enter your choice: ");
       Scanf("%d", &choice);
       Switch(choice){
            Case 1:
                 Insert();
                 break;
            Case 2:
                 display();
                 break;
```

```
            Case 0:
                exit(0);
                break;
            default:
                Printf("\n Enter valid choice ! \n");
        }
    }
}
// function for inserting node in queue using linked list.
void Insert(){
    Struct node *Ptr;
    int item;
    // dynamically allocating memory.
    Ptr = (Struct node *)malloc(sizeof(Struct node));
    if (Ptr == NULL){ // Queue is full
            Printf("\n overflow \n");
            return;
    }
    else{
        Printf ("\n Enter value: ");
        Scanf ("%d", &item);
        Ptr -> data = item;
        if (front == NULL) //inserting node in empty queue
        {
            front = Ptr;
            rear = Ptr;
            front -> next = NULL;
            rear -> next = NULL;
        }
```

```
        else // inserting node next to Previous node
        {
          rear->next = Ptr;
          rear = Ptr;
          rear->next = NULL;
        }
        Printf("Inserted!\n");
      }
    }
// function to display the elements of queue.
void display(){
    Struct node *Ptr;
    Ptr = front;
    if (front == NULL)
        Printf("\nEmpty queue\n");
    else {
        Printf("values inside Queues are: ");
        while (Ptr != NULL){
            Printf("%d ", Ptr->data);
            Ptr = Ptr->next;
        }
        Printf("\n");
    }
}
```

Output:

```
Queue operation.
1. Insert an element
2. Display the queue.
0. Exit
```

Enter your choice: 1
Enter value: 43
Insested!
...
...
Enter your choice: 2
values inside Queue are: 43 46