LAB DIGITAL ASSIGNMENT 1

REG. NO-19BCE0811

NAME-AKSHAT SRIVASTAV

```c
#include<stdio.h>//inflix to postflix using stack
char stack[20];
int top = -1;
void push(char x)//providing an arguement
{
    stack[++top] = x;//going from last to first using stack
}

char pop()
{
    if(top == -1)//condition ti pop the expression from left to right
        return -1;
    else
return stack[top--]; 15
}

int priority(char x)//providing conditions of priority.
{
    if(x == '(')//least priority
        return 0;
    if(x == '+' || x == '-')
        return 1;
    if(x == '*' || x == '/')//highest priority
return 2; 25    }

main()//calling the main function
{
    char exp[20];
    char *e, x;//using pointers
    printf("Enter the expression :: ");
    scanf("%s",exp);
    e = exp;
    while(*e != '\0')//providing the condition
    {
        if(isalnum(*e))//if we have a character or number
            printf("%c",*e);
        else if(*e == '(')
            push(*e);//putting the open bracket sign
        else if(*e == ')')
        {
            while((x = pop()) != '(')//if we have ')' we pop everything upto '('.
                printf("%c", x);
        }
        else
        {
            while(priority(stack[top]) >= priority(*e))//the reverse case
                printf("%c",pop());
            push(*e);
        }
        e++;
    }
    while(top != -1)//if we dont't star from left the above everything is popped out.
    {
        printf("%c",pop());
    }
}
```

```c
#include<stdio.h>//insertion sort
int main()
{
    int n,array[1000],c,d,t,flag=0;//calling the variables
    printf("enter the number of elements\n");
    scanf("%d",&n);
    printf("enter %d elements\n",n);
    for(c=0;c<n;c++)
        scanf("%d",&array[c]);//taking input
    for(c=1;c<=n-1;c++)
    {
        t=array[c];
        for(d=c-1;d>=0;d--)//same array but the empty one element entered from last
position
        {
            if(array[d]>t)
            {
                array[d+1]=array[d];//sorting
            flag=1;//flag=1 is true and flag=0 is false. 19

            }
            else
                break;
        }
        if(flag)
    array[d+1]=t; 25   }
    printf("sorted list in ascending order\n");
    for(c=0;c<=n-1;c++)
    {
        printf("%d\n",array[c]);//sorted array
    }
    return 0;
}

```

```c
#include <stdio.h>//circular queue of given values.
#define SIZE 5
int items[SIZE];
int front = -1, rear =-1;//declaring variables
int isFull()
{
    if( (front == rear + 1) || (front == 0 && rear == SIZE-1)) return 1;//applying condition of that front starting from 0 and rear of total-1.
return 0; 9
}
int isEmpty()
{
    if(front == -1) return 1;
return 0; 14
}
void enQueue(int element)//enqueing the elements
{
    if(isFull()) printf("\n Queue is full!! \n");
    else
    {
        if(front == -1) front = 0;//if front is of -1 and 0.
        rear = (rear + 1) % SIZE;//extracting rear
        items[rear] = element;
    printf("\n Inserted -> %d", element); 24
    }
}
int deQueue()//dequeing
{
    int element;
    if(isEmpty()) {
        printf("\n Queue is empty !! \n");
        return(-1);
    } else {
        element = items[front];
        if (front == rear){
            front = -1;
            rear = -1;
        } /* Q has only one element, so we reset the queue after dequeing it. */
        else {
            front = (front + 1) % SIZE;

        }
        printf("\n Deleted element -> %d \n", element);
    return(element); 44
    }
}
void display()
{
    int i;
    if(isEmpty()) printf(" \n Empty Queue\n");
    else
    {
        printf("\n Front -> %d ",front);//assigning using values.
        printf("\n Items -> ");
        for( i = front; i!=rear; i=(i+1)%SIZE) {
        printf("%d ",items[i]); 56
        }
        printf("%d ",items[i]);
    printf("\n Rear -> %d \n",rear); 59
    }
}
int main()
{
    // Fails because front = -1
    deQueue();

    enQueue(1);
    enQueue(2);
    enQueue(3);
    enQueue(4);
    enQueue(5);//printing the values

    // Fails to enqueue because front == 0 && rear == SIZE - 1
    enQueue(6);

    display();
    deQueue();

    display();

    enQueue(7);
    display();

    // Fails to enqueue because front == rear + 1
```

```
84          enQueue(8);
85
86          return 0;
87      }
88
```

```c
#include <stdio.h>//just another program to enter our values
#define size 5

void insertq(int[], int);
void deleteq(int[]);
void display(int[]);

int front =  - 1;
int rear =  - 1;

int main()
{
    int n, ch;
    int queue[size];
    do
    {
        printf("\n\n Circular Queue:\n1. Insert \n2. Delete\n3. Display\n0. Exit");
        printf("\nEnter Choice 0-3? : ");
        scanf("%d", &ch);
        switch (ch)//using switch cases to ask whether to insert,delete , or display.
        {
            case 1:
                printf("\nEnter number: ");
                scanf("%d", &n);
                insertq(queue, n);
                break;
            case 2:
                deleteq(queue);
                break;
            case 3:
                display(queue);
                break;
        }
    }while (ch != 0);
}


void insertq(int queue[], int item)//same thing what we did in last program.
{
    if ((front == 0 && rear == size - 1) || (front == rear + 1))//front and rear values
assigning
    {
        printf("queue is full");
        return;
    }
    else if (rear == - 1)//if rear=-1
    {
        rear++;
        front++;
    }
    else if (rear == size - 1 && front > 0)//in circular if rear is -1 and front >0
    {
        rear = 0;
    }
    else
    {
        rear++;
    }
    queue[rear] = item;
}

void display(int queue[])//displaying the queue
{
    int i;
    printf("\n");
    if (front > rear)
    {
        for (i = front; i < size;  i++)
        {
            printf("%d ", queue[i]);
        }
        for (i = 0; i <= rear;  i++)
    printf("%d ", queue[i]); 73
    }
    else
    {
        for (i = front; i <= rear;  i++)
    printf("%d ", queue[i]); 78
    }
}

void deleteq(int queue[])//for deleting the elements
{
    if (front == - 1)
```

```
84          {
85              printf("Queue is empty ");
86          }
87          else if (front == rear)
88          {
89              printf("\n %d deleted", queue[front]);
90              front =  - 1;
91          rear =      - 1;
92          }
93          else
94          {
95              printf("\n %d deleted", queue[front]);
96              front++;
97          }
98      }
99
```