# DATA STRUCTURE LAB RECORD

**NAME-ADITI AKARSH**

**USN-1BM19CS007**

**SECTION -3A**

**DEPARTMENT-CSE**

**ACADEMIC YEAR-2020-2021**

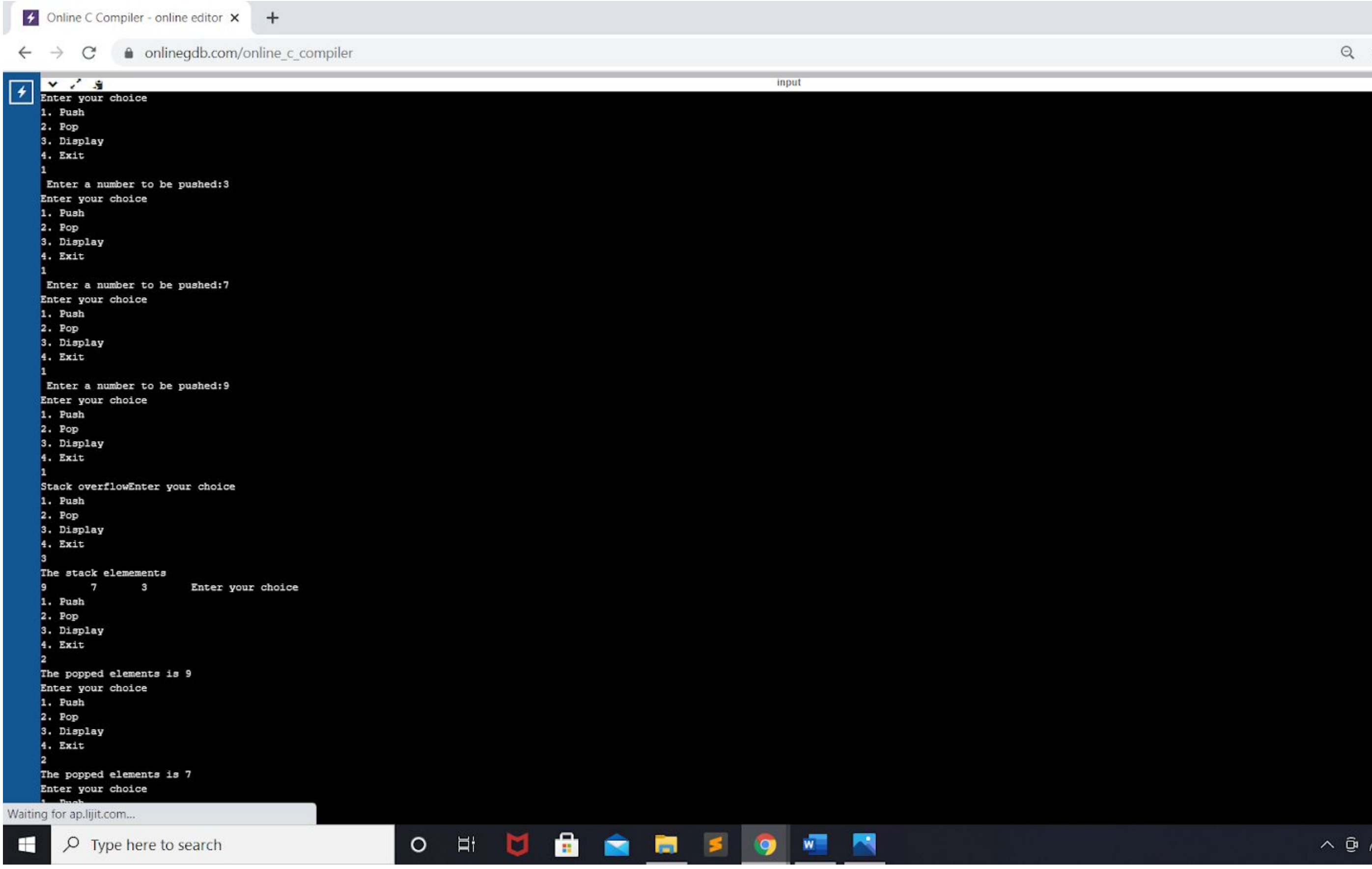
**LAB PROGRAM 1**:

```
#include
<stdio.h
>
           #define size 3
           int top=-1,choice,num,stack[size];
           void push();
           void pop();
           void display(int []);
           int main()
           {

               do
               {
               printf("Enter your choice\n");
               printf("1. Push\n");
               printf("2. Pop\n");
               printf("3. Display\n");
               printf("4. Exit\n");
               scanf("%d",&choice);
               switch(choice)
               {
                   case 1:
                           push();
                           break;

                       case 2:
```

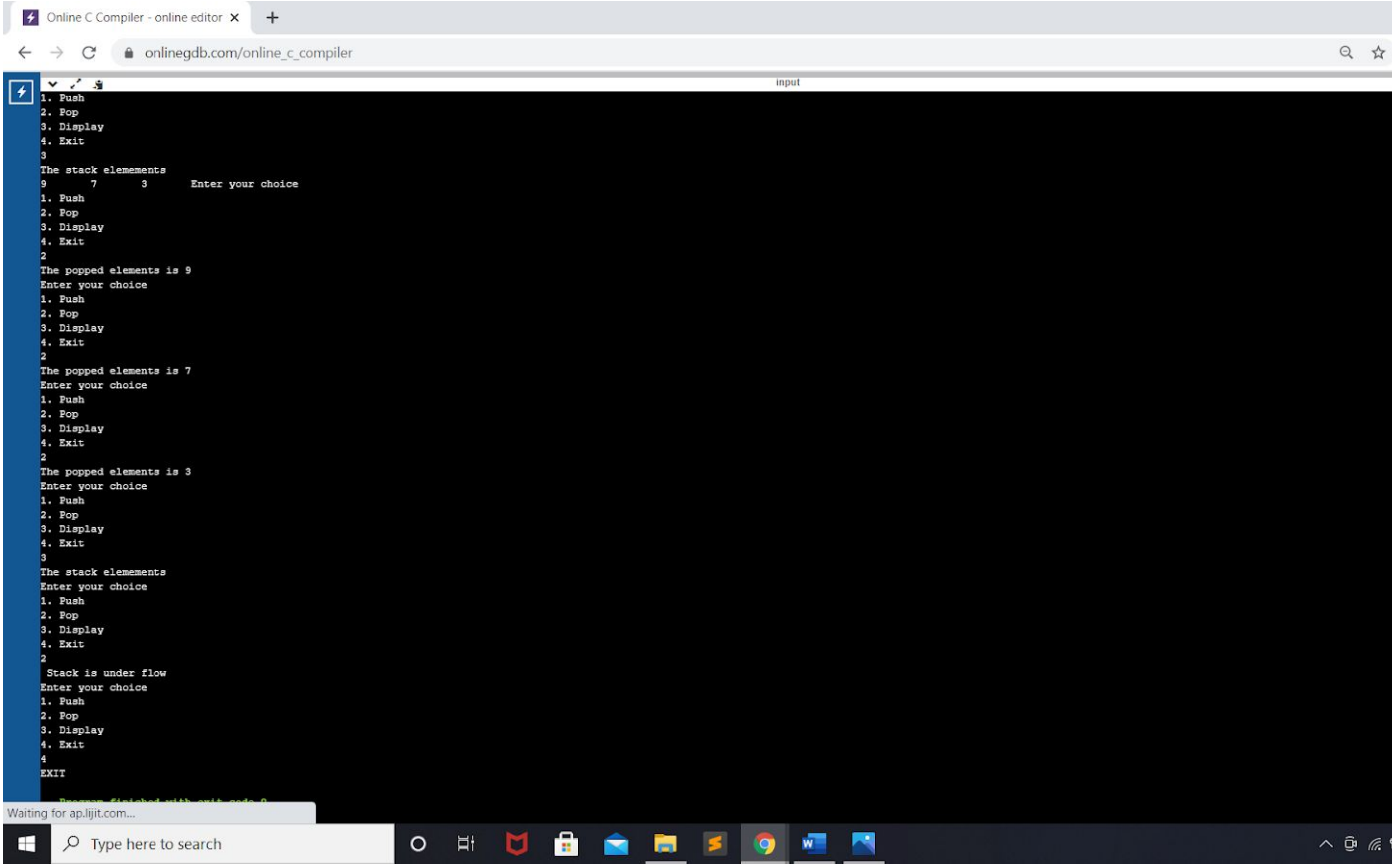
```
                pop();
                break;

            case 3:
                display(stack);
                break;

            case 4:
                printf("EXIT");
                break;

            default:

                printf ("\nINVALID
OPTION\n");
    }
    }
    while(choice!=4);
    return 0;
}




void push()
{
    if (top>=size-1)
    {
        printf("Stack overflow");
    }
    else
    {   printf(" Enter a number to be
pushed:");
        scanf("%d",&num);
        top++;
        stack[top]=num;
    }
}
```

```
void pop()
{

    if(top<=-1)
    {
        printf(" Stack is under flow\n");
    }
    else
    {
        printf("The popped elements is 
%d\n",stack[top]);
        top--;
    }


}


void display(int stack[])
{
    printf("The stack elemements\n");
    for(int i=top;i>=0;i--)
        printf("%d\t",stack[i]);

}
```

OUTPUT:


Online C Compiler - online editor
onlinegdb.com/online_c_compiler
input
Waiting for ap.lijit.com...
Type here to search

```
Enter your choice
1. Push
2. Pop
3. Display
4. Exit
1
 Enter a number to be pushed:3
Enter your choice
1. Push
2. Pop
3. Display
4. Exit
1
 Enter a number to be pushed:7
Enter your choice
1. Push
2. Pop
3. Display
4. Exit
1
 Enter a number to be pushed:9
Enter your choice
1. Push
2. Pop
3. Display
4. Exit
1
Stack overflowEnter your choice
1. Push
2. Pop
3. Display
4. Exit
3
The stack elemements
9       7       3       Enter your choice
1. Push
2. Pop
3. Display
4. Exit
2
The popped elements is 9
Enter your choice
1. Push
2. Pop
3. Display
4. Exit
2
The popped elements is 7
Enter your choice
1. Push
```


Online C Compiler - online editor
onlinegdb.com/online_c_compiler
input
Waiting for ap.lijit.com...
Type here to search

```
1. Push
2. Pop
3. Display
4. Exit
3
The stack elemements
9       7       3       Enter your choice
1. Push
2. Pop
3. Display
4. Exit
2
The popped elements is 9
Enter your choice
1. Push
2. Pop
3. Display
4. Exit
2
The popped elements is 7
Enter your choice
1. Push
2. Pop
3. Display
4. Exit
2
The popped elements is 3
Enter your choice
1. Push
2. Pop
3. Display
4. Exit
3
The stack elemements
Enter your choice
1. Push
2. Pop
3. Display
4. Exit
2
 Stack is under flow
Enter your choice
1. Push
2. Pop
3. Display
4. Exit
4
EXIT

   Program finished with exit code 0
```

*********************************************

*********************************************

*********************************************

**********************************************

************

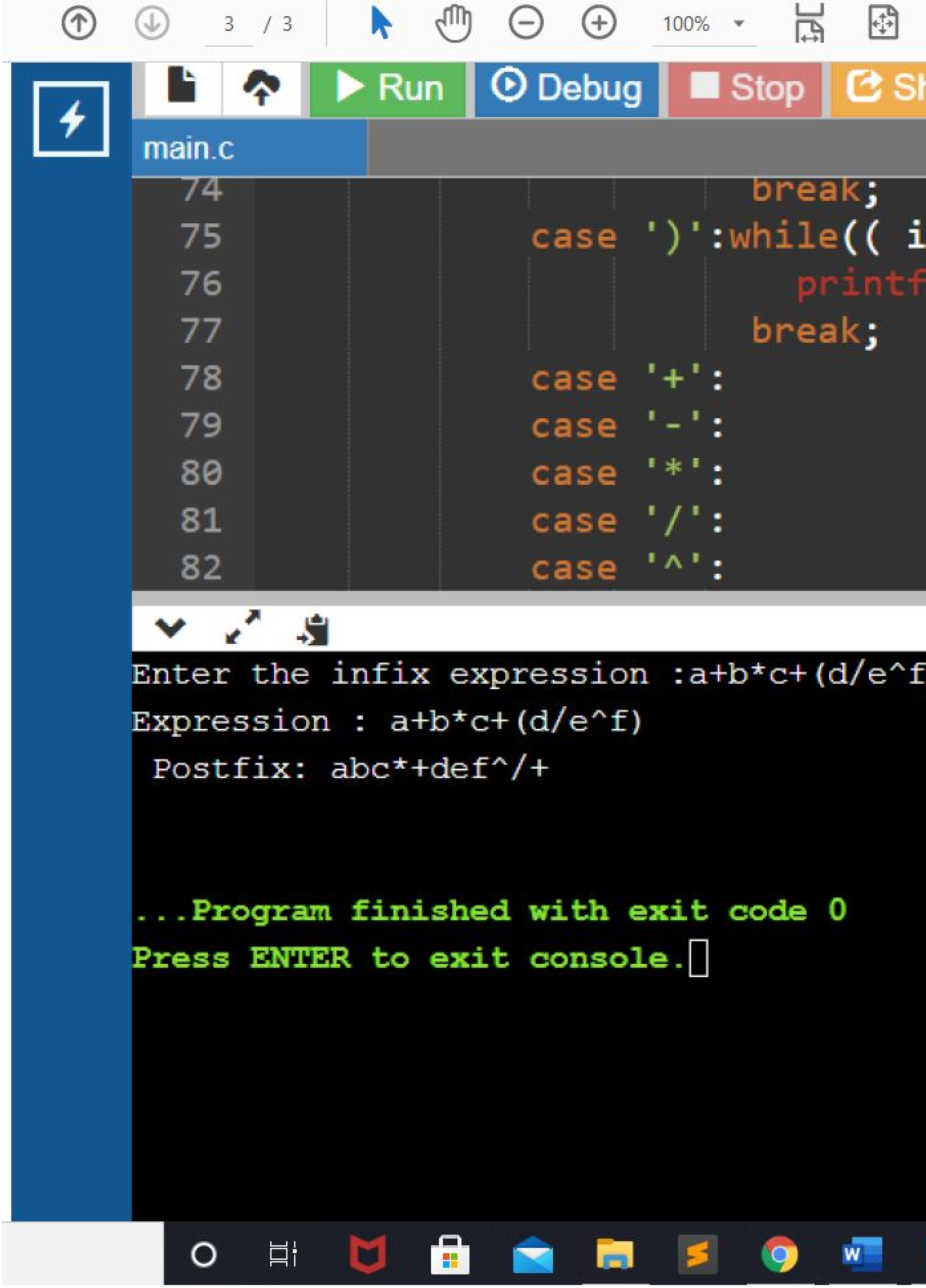
## LAB PROGRAM 2

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


                   #define SIZE 20


                   char stack[SIZE];
                   int top = -1;


                   void push(char ele)
                   {
                        if(top >= SIZE)
                        {
                             printf("\nStack Overflow.");
                        }
                        else
                        {
                             top = top+1;
                             stack[top] = ele;
                        }
                   }
                   char pop()
                   {
                        char ele ;
                        if(top==-1)
                        {
                             printf("stack under flow:
                   invalid infix expression");
                             getchar();
                             exit(1);
                        }
```

```
     else
     {
         ele = stack[top];
         top = top-1;
         return(ele);
     }
}
int is_operator(char symbol)
{
     if(symbol == '^' || symbol == '*'
|| symbol == '/' || symbol == '+' ||
symbol =='-')
     {
         return 1;
     }
     else
     {
        return 0;
     }
}
int higher(char symbol)
{
    switch(symbol)
    {
        case '^':
            return(3);
            break;
        case '*':
        case '/':
            return(2);
            break;
        case '+':
        case '-':
            return(1);
            break;
        default:
            return(0);
            break;
    }
```

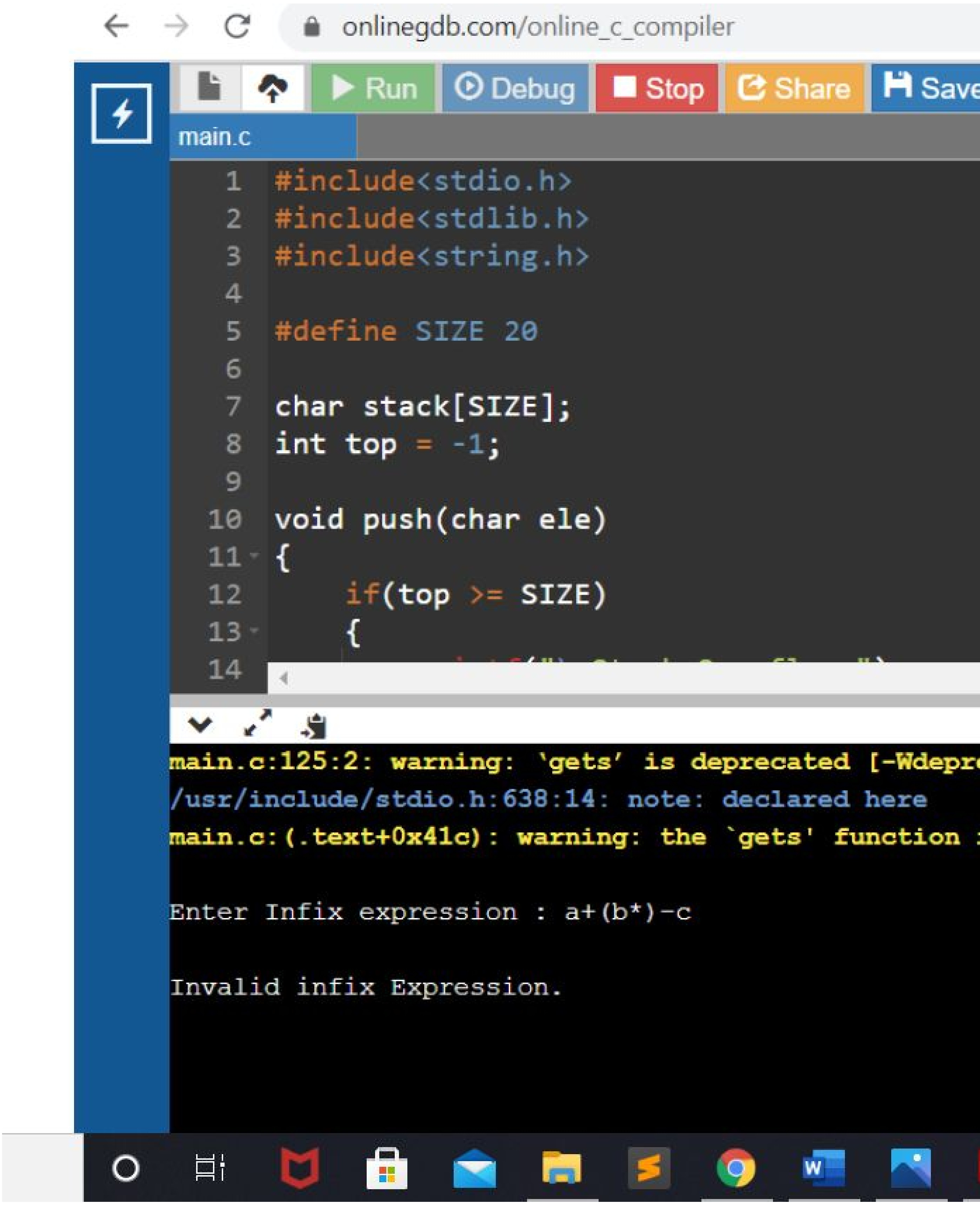
```
}
void InfixToPostfix(char infix_exp[],
char postfix_exp[])
{
     int i=0, j=0;
     char ele;
     char x;
     push('(');
     strcat(infix_exp,")");
     ele=infix_exp[i];
     while(ele != '\0')
     {
          if(ele == '(')
          {
               push(ele);
          }
          else if(ele=='A' || ele=='B'
|| ele=='C' || ele=='D' || ele=='E'
|| ele=='F' || ele=='G' || ele=='H'
|| ele=='I' || ele=='J' || ele=='K'
|| ele=='L' || ele=='M' || ele=='N'
                       || ele=='O' ||
ele=='P' || ele=='Q' || ele=='R' ||
ele=='S' || ele=='T' || ele=='U' ||
ele=='V' || ele=='W' || ele=='X' ||
ele=='Y' || ele=='Z' || ele=='0' ||
ele=='1'
                       || ele=='2' ||
ele=='3' || ele=='4' || ele=='5' ||
ele=='6' || ele=='7' || ele=='8' ||
ele=='9')
          {
               postfix_exp[j] = ele;
               j++;
          }
          else if(is_operator(ele) ==
1)
          {
               x=pop();
```

```
            while(is_operator(x) ==
1 && higher(x)>= higher(ele))
            {
                postfix_exp[j] = x;
                j++;
                x = pop();
            }
            push(x);
            push(ele);
        }
        else if(ele == ')')
        {
            x = pop();
            while(x != '(')
            {
                postfix_exp[j] = x;
                j++;
                x = pop();
            }
        }
        else
        {
            printf("\nInvalid infix
Expression.\n");
            getchar();
            exit(1);
        }
        i++;
       ele = infix_exp[i];
    }
    postfix_exp[j] = '\0';
}
int main()
{
    char infix[SIZE], postfix[SIZE];
    printf("\nEnter Infix expression
: ");
    gets(infix);
    InfixToPostfix(infix,postfix);
    printf("Postfix Expression is: 
");
    puts(postfix);
    return 0;
}
```



```
                        break;
            case ')':while(( i
                        printf
                        break;
            case '+':
            case '-':
            case '*':
            case '/':
            case '^':
```

```
Enter the infix expression :a+b*c+(d/e^f
Expression : a+b*c+(d/e^f)
 Postfix: abc*+def^/+



...Program finished with exit code 0
Press ENTER to exit console.
```

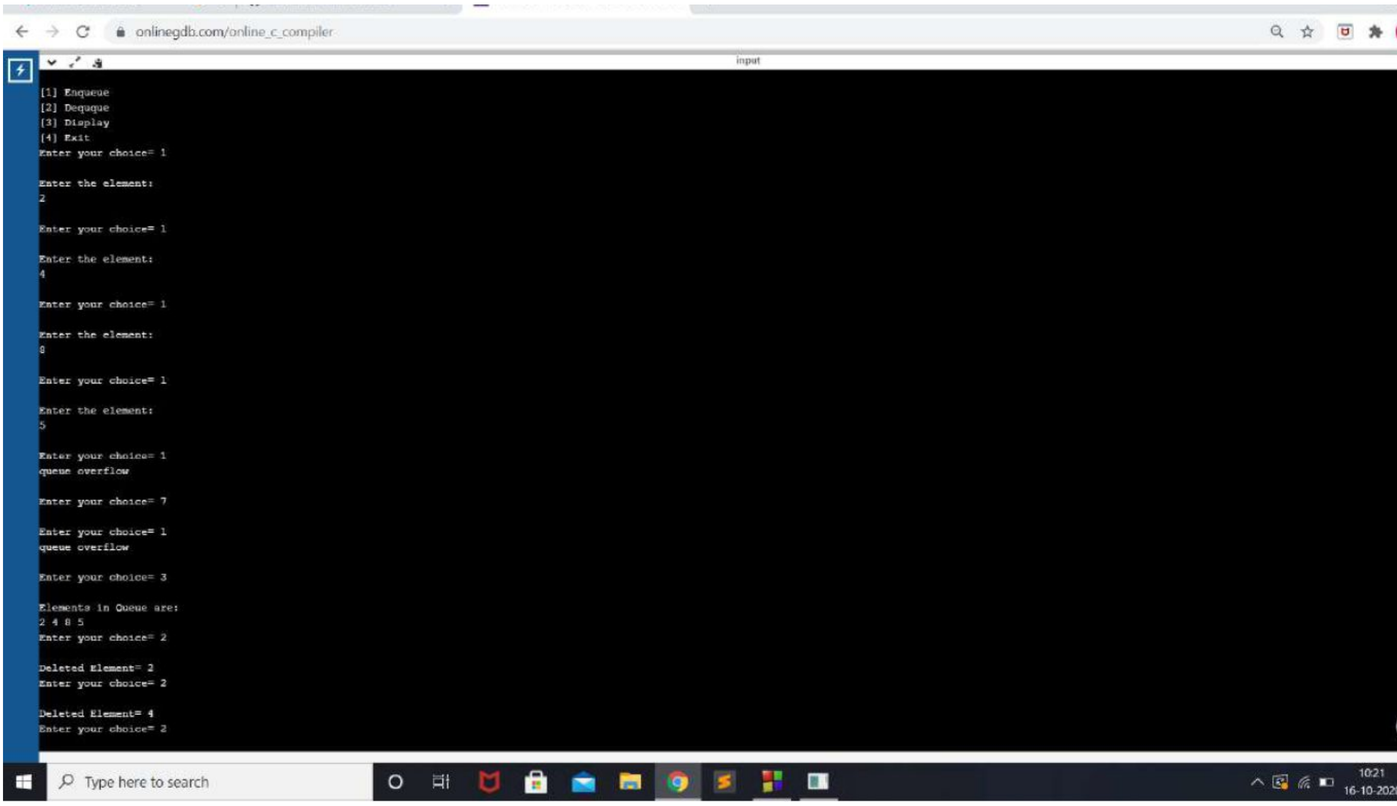


```c
#include<stdio.h>
#include<stdlib.h>
#include<string.h>

#define SIZE 20

char stack[SIZE];
int top = -1;

void push(char ele)
{
    if(top >= SIZE)
    {
```

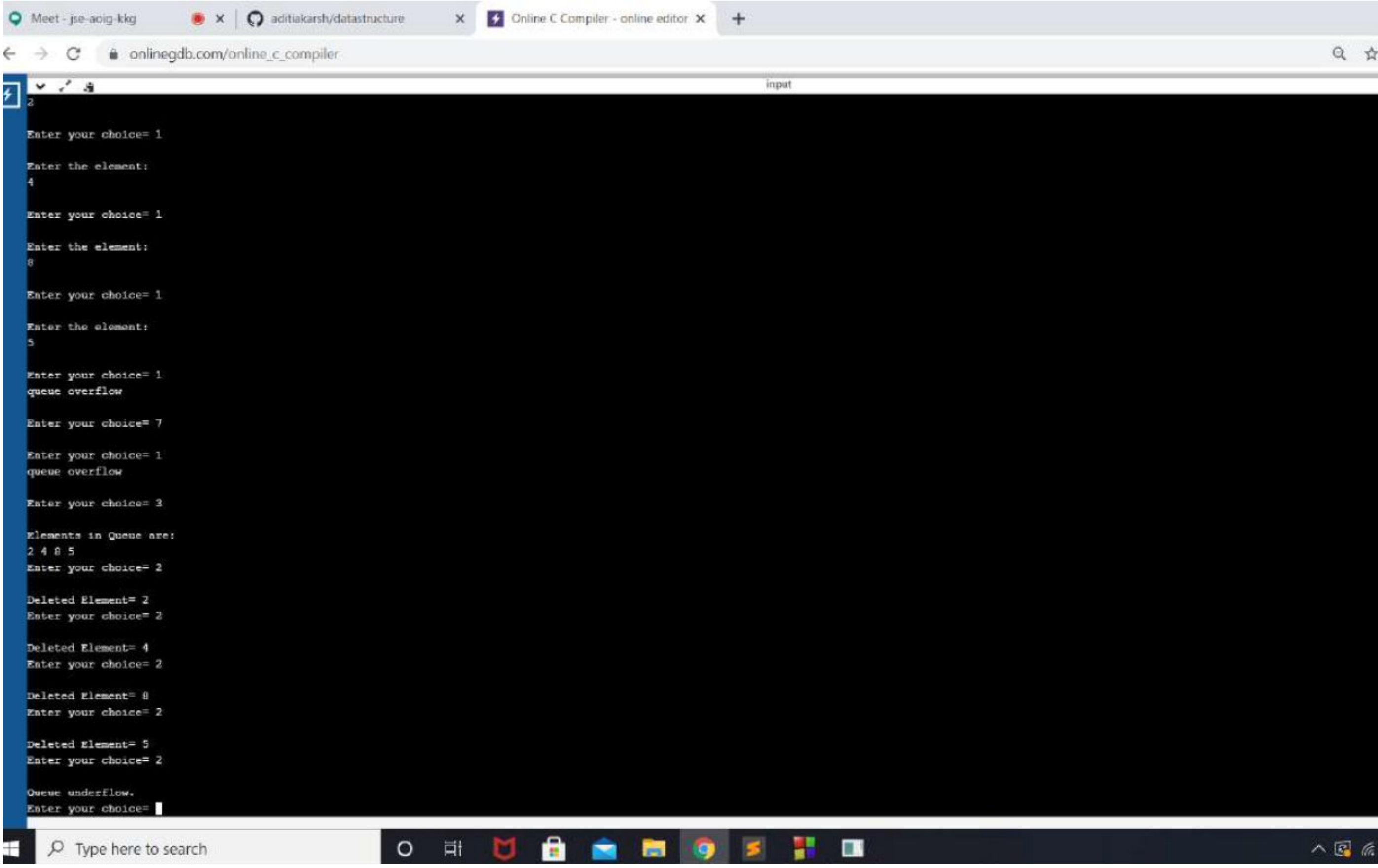
```
main.c:125:2: warning: ‘gets’ is deprecated [-Wdepre
/usr/include/stdio.h:638:14: note: declared here
main.c:(.text+0x41c): warning: the `gets' function 

Enter Infix expression : a+(b*)-c

Invalid infix Expression.
```

## LAB PROGRAM 3:

```
#include<stdio.h>
#define maxsize 5
void enqueue(int *queue,int *front, int *rear)
    {
    int ele;
    if(*rear>=maxsize-1)
        {
        printf("queue overflow\n");
        return;
        }
    if(*front==-1)
        {
        (*front)++;
        }
    (*rear)++;
    printf("\nEnter the element:\n");
    scanf("%d",&ele);
    *(queue+*rear)=ele;
    }


void display(int *queue,int front,int rear)
    {
    if(front==-1&&rear==-1)
        printf("\nQueue is empty");
    else
        {
        printf("\nElements in Queue are:\n");
        for(int i=front;i<=rear;i++)
            {
            printf("%d ",*(queue+i));
            }
        }
    }


void dequeue(int *queue,int *front, int *rear)
    {
    int ele;
    if(*front==-1&&*rear==-1)
        {
        printf("\nQueue underflow.");
        return;
        }
    else if(*front==*rear)
        {
        ele=*(queue+*front);
        *front=-1;
        *rear=-1;
        }
    else
        {
        ele=*(queue+*front);
        (*front)++;
        }
    printf("\nDeleted Element= %d",ele);
    }


void main()
{
int front1=-1,rear1=-1;
int queue1[maxsize];
int choice;
    printf("\n[1] Enqueue");
    printf("\n[2] Dequeue");
    printf("\n[3] Display");
    printf("\n[4] Exit");



do
    {
    printf("\nEnter your choice= ");
    scanf("%d",&choice);


    switch(choice)
        {
        case 1: enqueue(queue1,&front1,&rear1);
            break;
        case 2: dequeue(queue1,&front1,&rear1);
            break;
        case 3: display(queue1,front1,rear1);
        }
    }while(choice!=4);


}
```



```
[1] Enqueue
[2] Dequque
[3] Display
[4] Exit
Enter your choice= 1

Enter the element:
2

Enter your choice= 1

Enter the element:
4

Enter your choice= 1

Enter the element:
8

Enter your choice= 1

Enter the element:
5

Enter your choice= 1
queue overflow

Enter your choice= 7

Enter your choice= 1
queue overflow

Enter your choice= 3

Elements in Queue are:
2 4 8 5
Enter your choice= 2

Deleted Element= 2
Enter your choice= 2

Deleted Element= 4
Enter your choice= 2
```

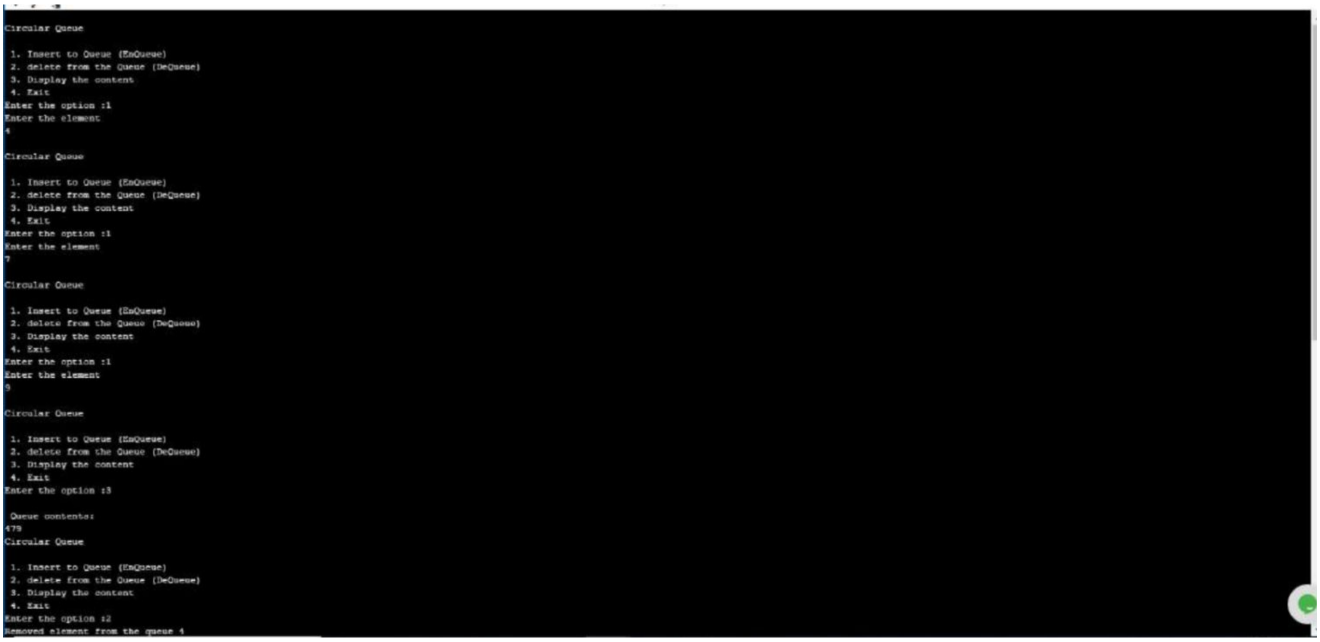

Meet - jse-aoig-kkg
aditiakarsh/datastructure
Online C Compiler - online editor
onlinegdb.com/online_c_compiler
input
Type here to search

```
2


Enter your choice= 1

Enter the element:
4


Enter your choice= 1

Enter the element:
8


Enter your choice= 1

Enter the element:
5


Enter your choice= 1
queue overflow

Enter your choice= 7

Enter your choice= 1
queue overflow

Enter your choice= 3

Elements in Queue are:
2 4 8 5
Enter your choice= 2

Deleted Element= 2
Enter your choice= 2

Deleted Element= 4
Enter your choice= 2

Deleted Element= 8
Enter your choice= 2

Deleted Element= 5
Enter your choice= 2

Queue underflow.
Enter your choice=
```

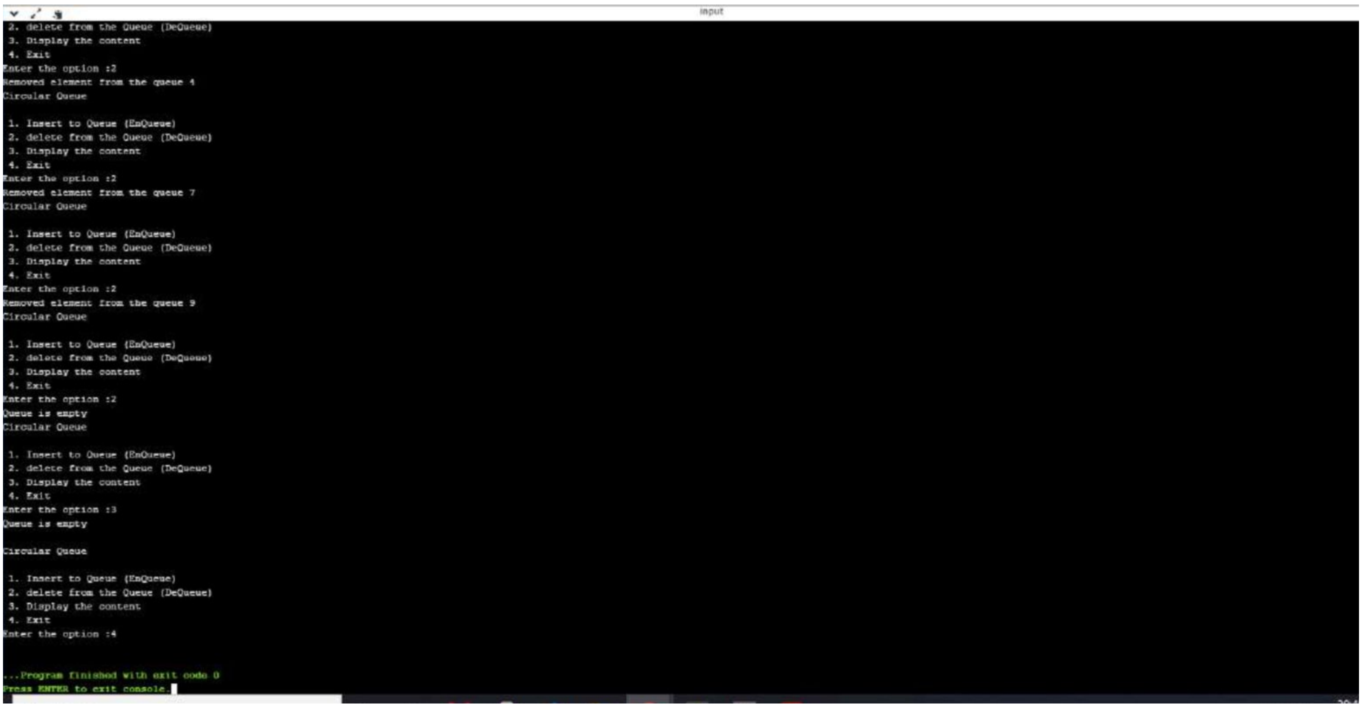
# LAB PROGRAM 4

```
#include<stdio.h
>
                    #include <stdlib.h>
                    #define MAX 3


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


                    int queue[MAX];


                    void Enque(int);
                    void Deque();
                    void display();
```

```
int main(int argc, char **argv)
{
       int option;
    int item;
    do{
        printf("\nCircular Queue\n");
        printf("\n 1. Insert to Queue (EnQueue)");
        printf("\n 2. delete from the Queue (DeQueue)");
        printf("\n 3. Display the content ");
        printf("\n 4. Exit\n");
        printf("Enter the option :");
        scanf("%d",&option);
        switch(option)
        {
            case 1:  printf("Enter the element\n");
                     scanf("%d",&item);
                     Enque(item);
                     break;
            case 2: Deque();

                    break;
            case 3: display();
                    break;
            case 4: exit(0);
        }
    } while (option!=4);
       return 0;
}


void Enque(int ele)
{
    if(((front == 0 && rear == MAX - 1))|| (front == rear + 1)
)
    {
       printf("Queue is full\n");return;

    }
    else
    {
      rear=(rear+1)%MAX;
      queue[rear]=ele;
      if(front ==-1)
          front=0;


    }
}
void Deque()
{
    int item;
    if((front == -1)&&(rear == -1))
    {

        printf("Queue is empty");
    }
    else
    {
        item=queue[front];
        printf("Removed element from the queue %d",item);
        if(front==rear)
        {
            front=-1;
            rear=-1;
        }
        else
        {
            front=(front+1)%MAX;
        }

    }

}


void display()
{
    int i;
    if((front==-1)&& (rear==-1))
    {

        printf("Queue is empty\n");return;

    }
    else
    {
        printf("\n Queue contents:\n");
        i=front;
        do
        {
           printf("%d",queue[i]);
           if(i==rear)
               break;
           i=(i+1)%MAX;
        }while (i!=front);

    }
}
```



```
Circular Queue

 1. Insert to Queue (EnQueue)
 2. delete from the Queue (DeQueue)
 3. Display the content
 4. Exit
Enter the option :1
Enter the element
4

Circular Queue

 1. Insert to Queue (EnQueue)
 2. delete from the Queue (DeQueue)
 3. Display the content
 4. Exit
Enter the option :1
Enter the element
7

Circular Queue

 1. Insert to Queue (EnQueue)
 2. delete from the Queue (DeQueue)
 3. Display the content
 4. Exit
Enter the option :1
Enter the element
9

Circular Queue

 1. Insert to Queue (EnQueue)
 2. delete from the Queue (DeQueue)
 3. Display the content
 4. Exit
Enter the option :3

 Queue contents:
479
Circular Queue

 1. Insert to Queue (EnQueue)
 2. delete from the Queue (DeQueue)
 3. Display the content
 4. Exit
Enter the option :2
Removed element from the queue 4
```



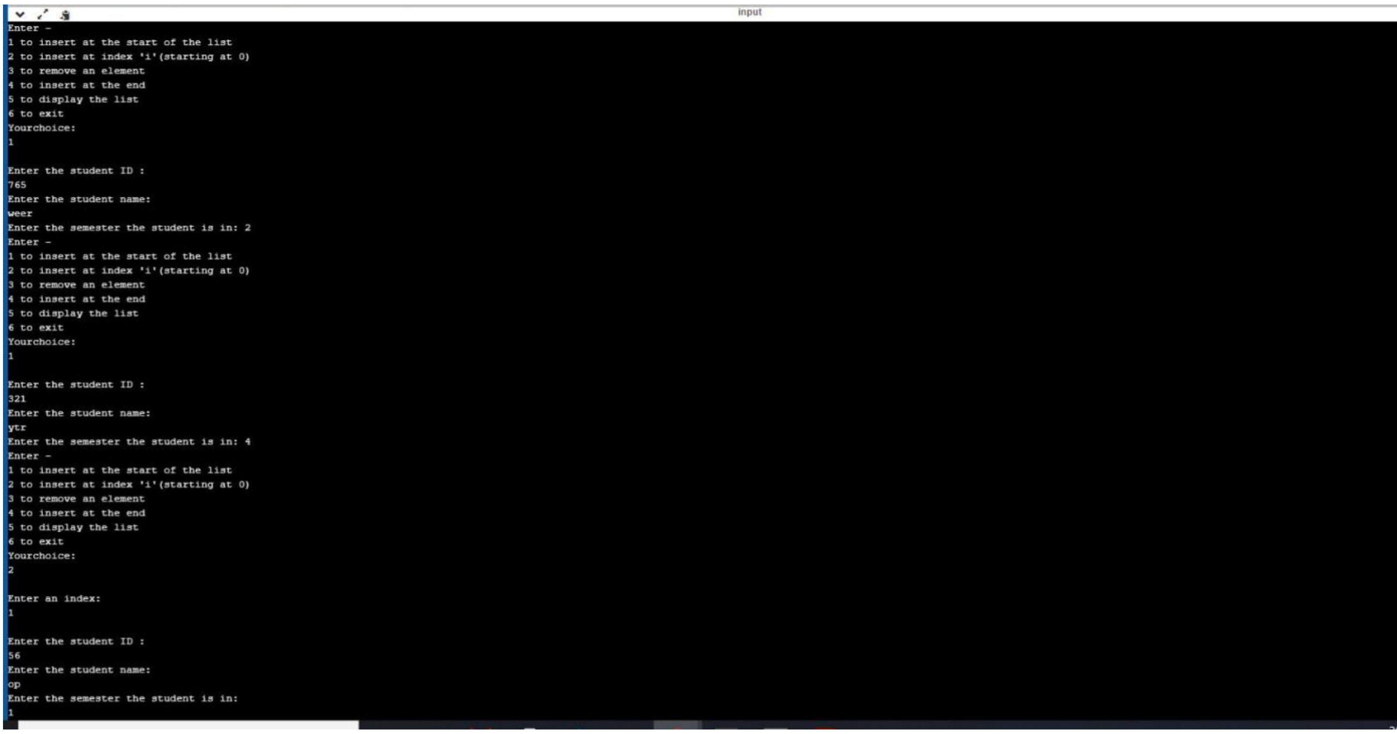
```
 2. delete from the Queue (DeQueue)
 3. Display the content
 4. Exit
Enter the option :2
Removed element from the queue 4
Circular Queue

 1. Insert to Queue (EnQueue)
 2. delete from the Queue (DeQueue)
 3. Display the content
 4. Exit
Enter the option :2
Removed element from the queue 7
Circular Queue

 1. Insert to Queue (EnQueue)
 2. delete from the Queue (DeQueue)
 3. Display the content
 4. Exit
Enter the option :2
Removed element from the queue 9
Circular Queue

 1. Insert to Queue (EnQueue)
 2. delete from the Queue (DeQueue)
 3. Display the content
 4. Exit
Enter the option :2
Queue is empty
Circular Queue

 1. Insert to Queue (EnQueue)
 2. delete from the Queue (DeQueue)
 3. Display the content
 4. Exit
Enter the option :3
Queue is empty

Circular Queue

 1. Insert to Queue (EnQueue)
 2. delete from the Queue (DeQueue)
 3. Display the content
 4. Exit
Enter the option :4


...Program finished with exit code 0
Press ENTER to exit console.
```

# LAB PROGRAM 5

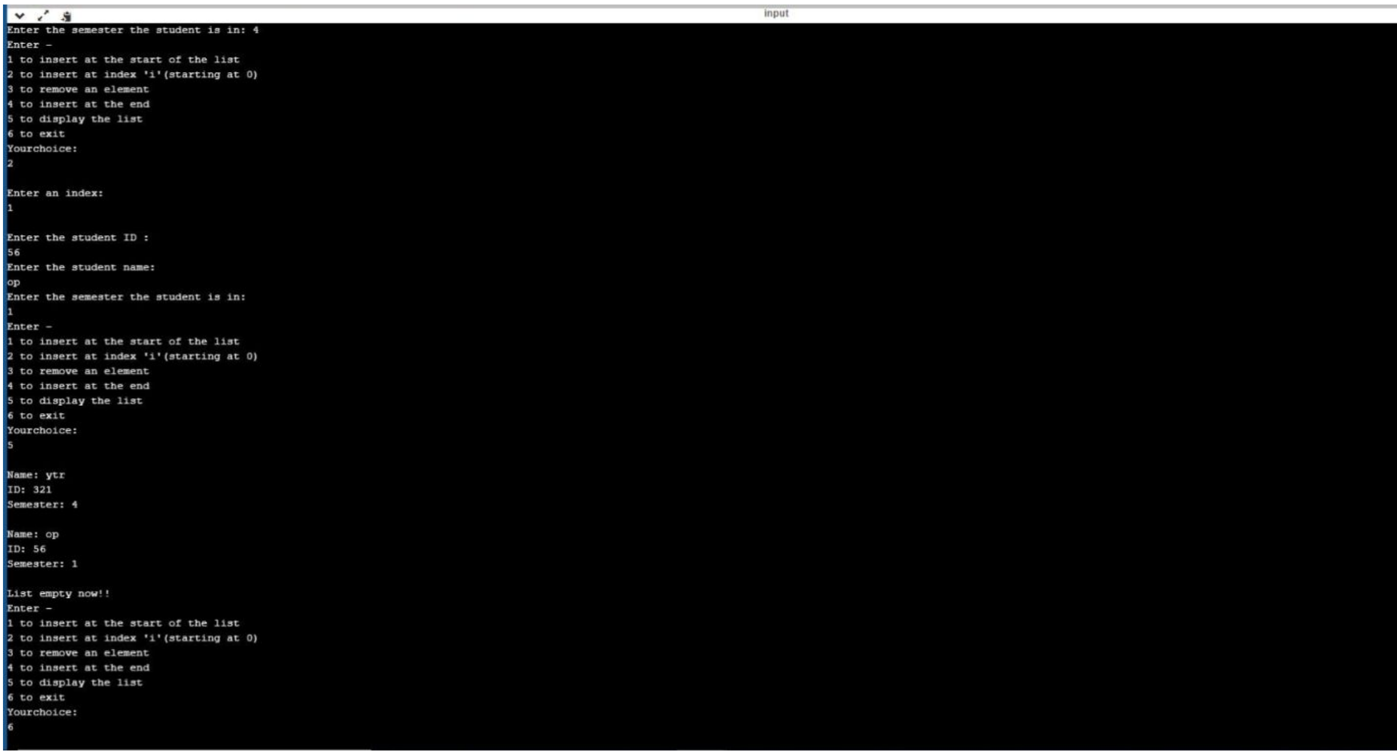
```
#include
<stdio.h
>
            #include <stdlib.h>
            #include <string.h>


            typedef struct node {
                    int sem;
                    char name[30];
                    char ID[30];
                    struct node *next;
            }node;


            node* removed = NULL;
            node* start = NULL;
            node* end = NULL;




            void insert()
            {
                    fflush(stdin);
                    printf("\nEnter the student ID :\n");
                    char I[30];
                    scanf("%s", I);
                    printf("Enter the student name:\n");
                    char n[30];
                    scanf("%s", n);
                    printf("Enter the semester the student is in: ");
                    int s =0;
                    scanf("%d", &s);
                    fflush(stdin);
                    node* temp = (node*)malloc(sizeof(node));
                    memcpy(temp->name, n, 20);
                    memcpy(temp->ID, I, 20);
                    temp->sem = s;
                    temp->next = start;
                    start = temp;
                    node* forEnd;
                                  forEnd = start;
                                  while(forEnd != NULL && forEnd->next != NULL)
                                  {
                                          forEnd = forEnd->next;

                                  }
                                  end = forEnd;
            }
            void insertAtI(int i)
            {
                    node* current = (node*)malloc(sizeof(node));
                    node* previous = (node*)malloc(sizeof(node));
                    current = start;
                    previous = start;
                    int j = 0;
                    while(current != NULL)
                    {
                            if(j == i)
```

```
                {
                        fflush(stdin);
                        printf("\nEnter the student ID :\n");
                        char I[30];
                        scanf("%s", I);
                        printf("Enter the student name:\n");
                        char n[30];
                        scanf("%s", n);
                        printf("Enter the semester the student is in:\n");
                        int s =0;
                        scanf("%d", &s);
                        fflush(stdin);
                        node* temp = (node*)malloc(sizeof(node));
                        memcpy(temp->name, n, 20);
                        memcpy(temp->ID, I, 20);
                        temp->sem = s;
                        temp->next = current->next;
                        if(i != 0)
                                previous->next = temp;
                        else
                                start = temp;
                        node* forEnd;
                        forEnd = start;
                        while(forEnd != NULL && forEnd->next != NULL)
                        {
                                forEnd = forEnd->next;
                        }
                        end = forEnd;
                        return;
                }
                previous = current;
                current = current->next;
                j++;
        }
        if(start == NULL && i == 0)
        {
                        fflush(stdin);
                        printf("\nEnter the student ID :\n");
                        char I[30];
                        scanf("%s", I);
                        printf("Enter the student name:\n");
                        char n[30];
                        scanf("%s", n);
                        printf("Enter the semester the student is in:\n");
                        int s =0;
                        scanf("%d", &s);
                        fflush(stdin);
                        node* temp = (node*)malloc(sizeof(node));
                        memcpy(temp->name, n, 20);
                        memcpy(temp->ID, I, 20);
                        temp->sem = s;
                        temp->next = NULL;
                        start = temp;
                        end = temp;
                        node* forEnd;
                        forEnd = start;
                        while(forEnd != NULL && forEnd->next != NULL)
                        {
                                forEnd = forEnd->next;
                        }
                        end = forEnd;
        }
        else
                printf("\n\t\tIndex is out of bounds!!!\n");
}
void display()
{
        node* temp = (node*)malloc(sizeof(node));
        temp = start;
        while(temp != NULL)
        {
                printf("\nName: %s\nID: %s\nSemester: %d\n", temp->name,
temp->ID, temp->sem);
                temp = temp->next;
        }
        printf("\nList empty now!!\n");
}


void displayN(node* toP)
{
        printf("\nName: %s\nID: %s\nSemester: %d\n", toP->name, toP->ID,
toP->sem);
}


void removeE()
{
        node* temp;
        temp = start;

        if(temp != NULL)
        {
                removed = start;
                start = start->next;
                printf("\n\t\tRemoved the first element\n");
                displayN(removed);
        }
        else
        {
                removed = NULL;
                printf("\n\t\tNo element is in the list\n");
        }

}
void insertEnd()
{
        node* temp;
        int i = 0;
        while(temp != NULL && temp->next != NULL)
        {
                i++;
                temp = temp->next;
        }
        insertAtI(i);
}
int main()
{
        int choice = 0;
        do{
```

```
		printf("Enter - \n1 to insert at the start of the list\n2 to
insert at index 'i'(starting at 0)\n3 to remove an element\n4 to insert at
the end\n5 to display the list\n6 to exit\nYourchoice:\n");
		scanf("%d", &choice);
		fflush(stdin);
		switch(choice)
		{
			case 1:
				insert();
				break;
			case 2:
				printf("\nEnter an index:\n");
				int k = 0;
				scanf("%d", &k);
				fflush(stdin);
				insertAtI(k);
				break;
			case 3:
				removeE();
				break;
			case 4:
				insertEnd();
				break;
			case 5:
				display();
				break;
			case 6:
				printf("\n\t\texiting......\n");
				break;
			default:
				printf("\n\t\tInvalid choice.\n\t\tTry
again.....\n");
		}
	}while(choice != 6);
	return 0;
}
```



```
Enter -
1 to insert at the start of the list
2 to insert at index 'i'(starting at 0)
3 to remove an element
4 to insert at the end
5 to display the list
6 to exit
Yourchoice:
1

Enter the student ID :
765
Enter the student name:
weer
Enter the semester the student is in: 2
Enter -
1 to insert at the start of the list
2 to insert at index 'i'(starting at 0)
3 to remove an element
4 to insert at the end
5 to display the list
6 to exit
Yourchoice:
1

Enter the student ID :
321
Enter the student name:
ytr
Enter the semester the student is in: 4
Enter -
1 to insert at the start of the list
2 to insert at index 'i'(starting at 0)
3 to remove an element
4 to insert at the end
5 to display the list
6 to exit
Yourchoice:
2

Enter an index:
1

Enter the student ID :
56
Enter the student name:
op
Enter the semester the student is in:
1
```



```
Enter the semester the student is in: 4
Enter -
1 to insert at the start of the list
2 to insert at index 'i'(starting at 0)
3 to remove an element
4 to insert at the end
5 to display the list
6 to exit
Yourchoice:
2

Enter an index:
1

Enter the student ID :
56
Enter the student name:
op
Enter the semester the student is in:
1
Enter -
1 to insert at the start of the list
2 to insert at index 'i'(starting at 0)
3 to remove an element
4 to insert at the end
5 to display the list
6 to exit
Yourchoice:
5

Name: ytr
ID: 321
Semester: 4

Name: op
ID: 56
Semester: 1

List empty now!!
Enter -
1 to insert at the start of the list
2 to insert at index 'i'(starting at 0)
3 to remove an element
4 to insert at the end
5 to display the list
6 to exit
Yourchoice:
6
```

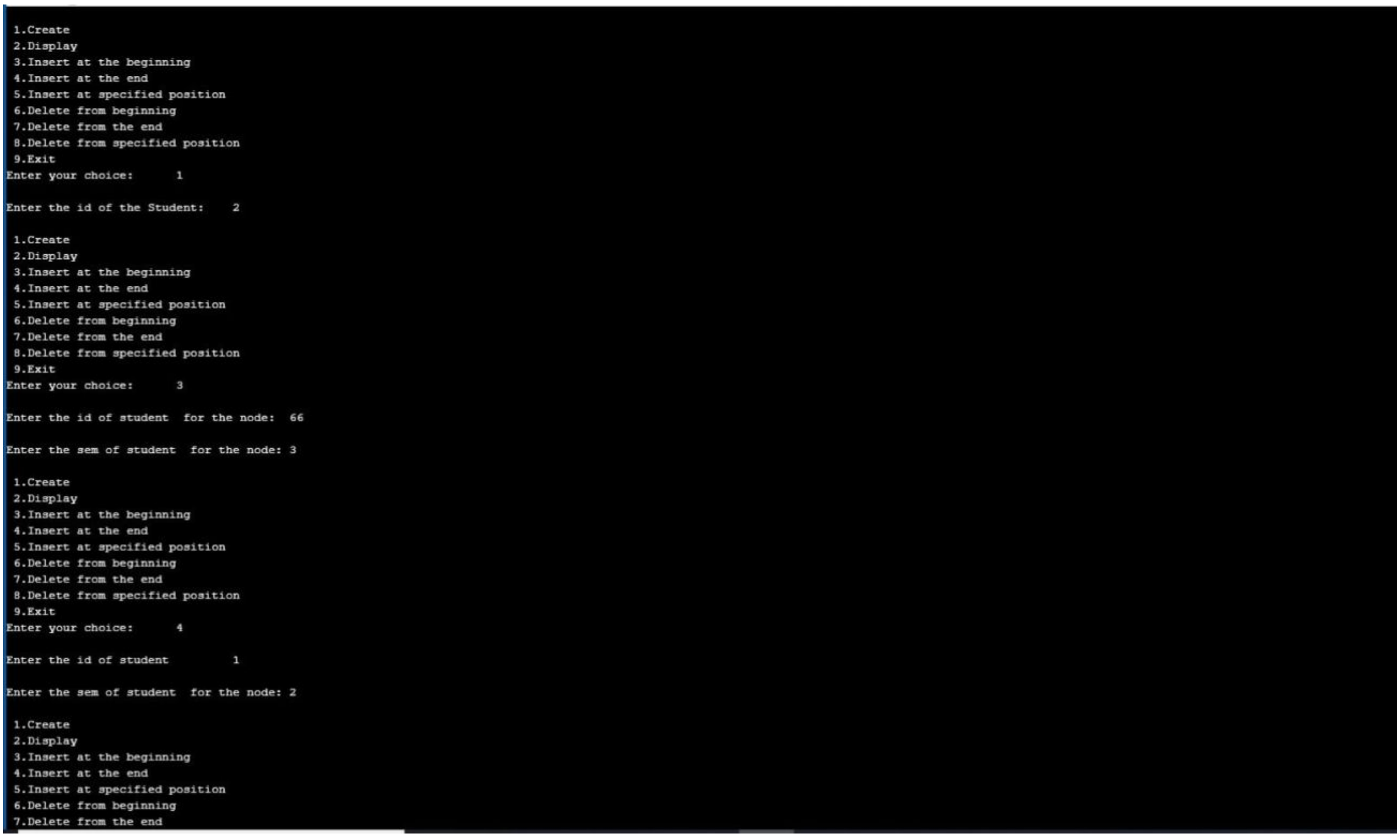
# LAB PROGRAM 6

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

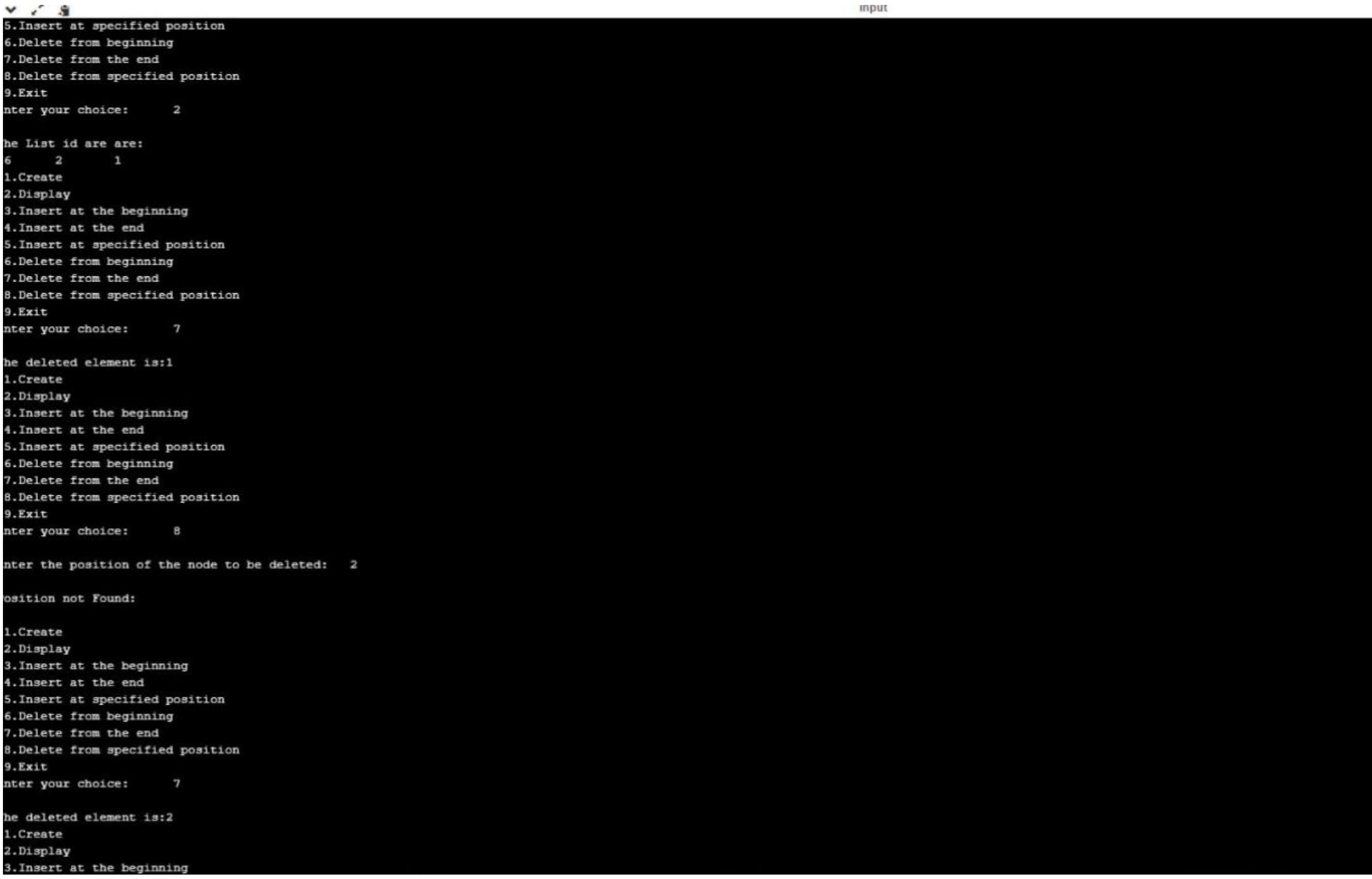
```
void create();
void display();
void insert_begin();
void insert_end();
void insert_pos();
void delete_begin();
void delete_end();
void delete_pos();




struct node
{
         int id,sem;
         //char name[20];
        //int info;
        struct node *next;
};
struct node *start=NULL;
int main()
{
        int choice;
        while(1){
                printf("\n 1.Create     \n");
                printf(" 2.Display    \n");
                printf(" 3.Insert at the beginning    \n");
                printf(" 4.Insert at the end  \n");
                printf(" 5.Insert at specified position       \n");
                printf(" 6.Delete from beginning      \n");
                printf(" 7.Delete from the end        \n");
                printf(" 8.Delete from specified position     \n");
                printf(" 9.Exit  \n");
                printf("Enter your choice:\t");
                scanf("%d",&choice);
                switch(choice)
                {
                        case 1:
                                        create();
                                        break;
                        case 2:
                                        display();
                                        break;
                        case 3:
                                        insert_begin();
                                        break;
                        case 4:
                                        insert_end();
                                        break;
                        case 5:
                                        insert_pos();
                                        break;
                        case 6:
                                        delete_begin();
                                        break;
                        case 7:
                                        delete_end();
                                        break;
                        case 8:
```

```
                                        delete_pos();
                                        break;


                        case 9:
                                        exit(0);
                                        break;


                        default:
                                        printf("\n Wrong
Choice:\n");
                                        break;
                }
        }
        return 0;
}
void create()
{
        struct node *temp,*ptr;
        temp=(struct node *)malloc(sizeof(struct node));
        if(temp==NULL)
        {
                printf("\nOut of Memory Space:\n");
                exit(0);
        }
        printf("\nEnter the id of the Student:\t");
        scanf("%d",&temp->id);
        temp->next=NULL;
        if(start==NULL)
        {
                start=temp;
        }
        else
        {
                ptr=start;
                while(ptr->next!=NULL)
                {
                        ptr=ptr->next;
                }
                ptr->next=temp;
        }
}
void display()
{
        struct node *ptr;
        if(start==NULL)
        {
                printf("\nList is empty:\n");
                return;
        }
        else
        {
                ptr=start;
                printf("\nThe List id are are:\n");
                while(ptr!=NULL)
                {
                        printf("%d\t",ptr->id );
                        ptr=ptr->next ;
                }
        }
```

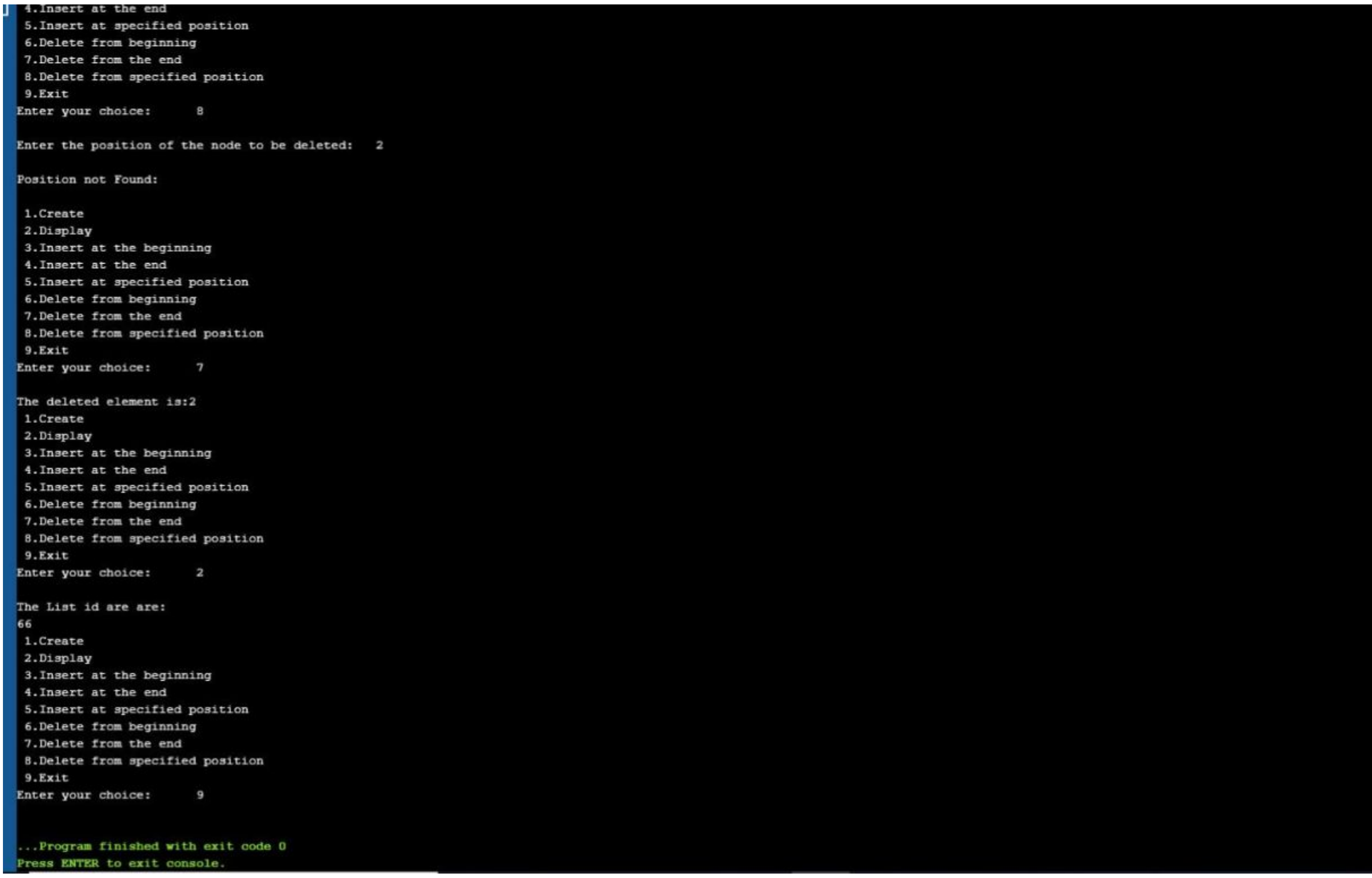
```
}
void insert_begin()
{
        struct node *temp;
        temp=(struct node *)malloc(sizeof(struct node));
        if(temp==NULL)
        {
                printf("\nOut of Memory Space:\n");
                return;
        }
        printf("\nEnter the id of student  for the node:\t" );
        scanf("%d",&temp->id);
        printf("\nEnter the sem of student  for the node:\t" );
        scanf("%d",&temp->sem);
        temp->next =NULL;
        if(start==NULL)
        {
                start=temp;
        }
        else
        {
                temp->next=start;
                start=temp;
        }
}
void insert_end()
{
        struct node *temp,*ptr;
        temp=(struct node *)malloc(sizeof(struct node));
        if(temp==NULL)
        {
                printf("\nOut of Memory Space:\n");
                return;
        }
        printf("\nEnter the id of student \t" );
        scanf("%d",&temp->id );
        printf("\nEnter the sem of student  for the node:\t" );
        scanf("%d",&temp->sem);
        temp->next =NULL;
        if(start==NULL)
        {
                start=temp;
        }
        else
        {
                ptr=start;
                while(ptr->next !=NULL)
                {
                        ptr=ptr->next ;
                }
                ptr->next =temp;
        }
}
void insert_pos()
{
        struct node *ptr,*temp;
        int i,pos;
        temp=(struct node *)malloc(sizeof(struct node));
        if(temp==NULL)
        {
                printf("\nOut of Memory Space:\n");
                return;
        }
        printf("\nEnter the position for the new id  to be 
inserted:\t");
        scanf("%d",&pos);
        printf("\nEnter the data value of the node:\t");
        scanf("%d",&temp->id) ;


        temp->next=NULL;
        if(pos==0)
        {
                temp->next=start;
                start=temp;
        }
        else
        {
                for(i=0,ptr=start;i<pos-1;i++) { ptr=ptr->next;
                        if(ptr==NULL)
                        {
                                printf("\nPosition not found\n");
                                return;
                        }
                }
                temp->next =ptr->next ;
                ptr->next=temp;
        }
}
void delete_begin()
{
        struct node *ptr;
        if(ptr==NULL)
        {
                printf("\nList is Empty:\n");
                return;
        }
        else
        {
                ptr=start;
                start=start->next ;
                printf("\nThe deleted element is :%d\t",ptr->id);
                free(ptr);
        }
}
void delete_end()
{
        struct node *temp,*ptr;
        if(start==NULL)
        {
                printf("\nList is Empty:");
                exit(0);
        }
        else if(start->next ==NULL)
        {
                ptr=start;
                start=NULL;
                printf("\nThe deleted id  is:%d\t",ptr->id);
                free(ptr);
        }
        else
        {
                ptr=start;
                while(ptr->next!=NULL)
                {
                        temp=ptr;
                        ptr=ptr->next;
                }
                temp->next=NULL;
                printf("\nThe deleted element is:%d\t",ptr->id);
                free(ptr);
        }
}
void delete_pos()
{
        int i,pos;
        struct node *temp,*ptr;
        if(start==NULL)
        {
                printf("\nThe List is Empty:\n");
                exit(0);
        }
        else
        {
                printf("\nEnter the position of the node to be
deleted:\t");
                scanf("%d",&pos);
                if(pos==0)
                {
                        ptr=start;
                        start=start->next ;
                        printf("\nThe deleted element
is:%d\t",ptr->id  );
                        free(ptr);
                }
                else
                {
                        ptr=start;
                        for(i=0;i<pos;i++) { temp=ptr;
ptr=ptr->next ;
                                if(ptr==NULL)
                                {
                                        printf("\nPosition not
Found:\n");
                                        return;
                                }
                        }
                        temp->next =ptr->next ;
                        printf("\nThe deleted element
is:%d\t",ptr->id);
                        free(ptr);
                }
        }
}
```



```
 1.Create
 2.Display
 3.Insert at the beginning
 4.Insert at the end
 5.Insert at specified position
 6.Delete from beginning
 7.Delete from the end
 8.Delete from specified position
 9.Exit
Enter your choice:      1

Enter the id of the Student:    2

 1.Create
 2.Display
 3.Insert at the beginning
 4.Insert at the end
 5.Insert at specified position
 6.Delete from beginning
 7.Delete from the end
 8.Delete from specified position
 9.Exit
Enter your choice:      3

Enter the id of student  for the node:  66

Enter the sem of student  for the node: 3

 1.Create
 2.Display
 3.Insert at the beginning
 4.Insert at the end
 5.Insert at specified position
 6.Delete from beginning
 7.Delete from the end
 8.Delete from specified position
 9.Exit
Enter your choice:      4

Enter the id of student         1

Enter the sem of student  for the node: 2

 1.Create
 2.Display
 3.Insert at the beginning
 4.Insert at the end
 5.Insert at specified position
 6.Delete from beginning
 7.Delete from the end
```



```
5.Insert at specified position
6.Delete from beginning
7.Delete from the end
8.Delete from specified position
9.Exit
nter your choice:      2

he List id are are:
6      2       1
1.Create
2.Display
3.Insert at the beginning
4.Insert at the end
5.Insert at specified position
6.Delete from beginning
7.Delete from the end
8.Delete from specified position
9.Exit
nter your choice:      7

he deleted element is:1
1.Create
2.Display
3.Insert at the beginning
4.Insert at the end
5.Insert at specified position
6.Delete from beginning
7.Delete from the end
8.Delete from specified position
9.Exit
nter your choice:      8

nter the position of the node to be deleted:   2

osition not Found:

1.Create
2.Display
3.Insert at the beginning
4.Insert at the end
5.Insert at specified position
6.Delete from beginning
7.Delete from the end
8.Delete from specified position
9.Exit
nter your choice:      7

he deleted element is:2
1.Create
2.Display
3.Insert at the beginning
```

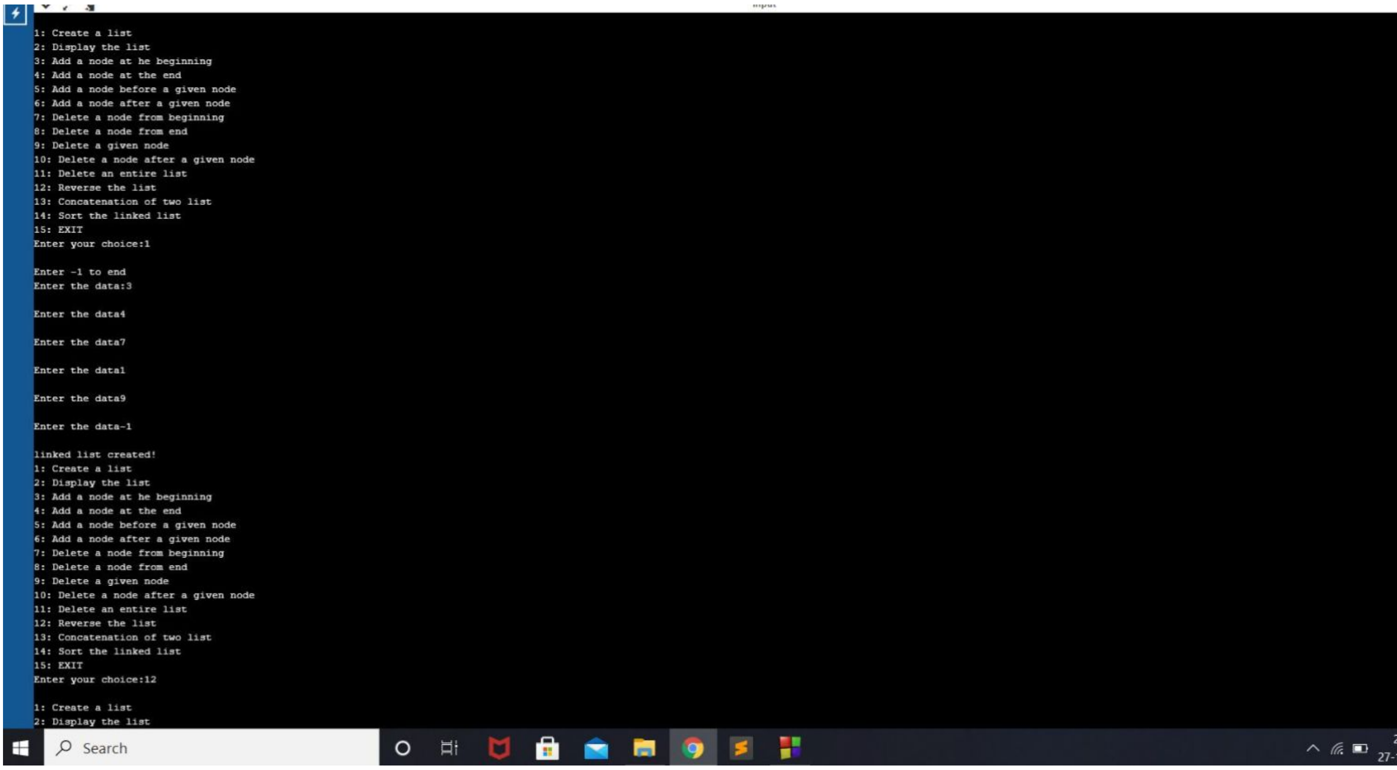


```
 4.Insert at the end
 5.Insert at specified position
 6.Delete from beginning
 7.Delete from the end
 8.Delete from specified position
 9.Exit
Enter your choice:      8

Enter the position of the node to be deleted:   2

Position not Found:

 1.Create
 2.Display
 3.Insert at the beginning
 4.Insert at the end
 5.Insert at specified position
 6.Delete from beginning
 7.Delete from the end
 8.Delete from specified position
 9.Exit
Enter your choice:      7

The deleted element is:2
 1.Create
 2.Display
 3.Insert at the beginning
 4.Insert at the end
 5.Insert at specified position
 6.Delete from beginning
 7.Delete from the end
 8.Delete from specified position
 9.Exit
Enter your choice:      2

The List id are are:
66
 1.Create
 2.Display
 3.Insert at the beginning
 4.Insert at the end
 5.Insert at specified position
 6.Delete from beginning
 7.Delete from the end
 8.Delete from specified position
 9.Exit
Enter your choice:      9


...Program finished with exit code 0
Press ENTER to exit console.
```

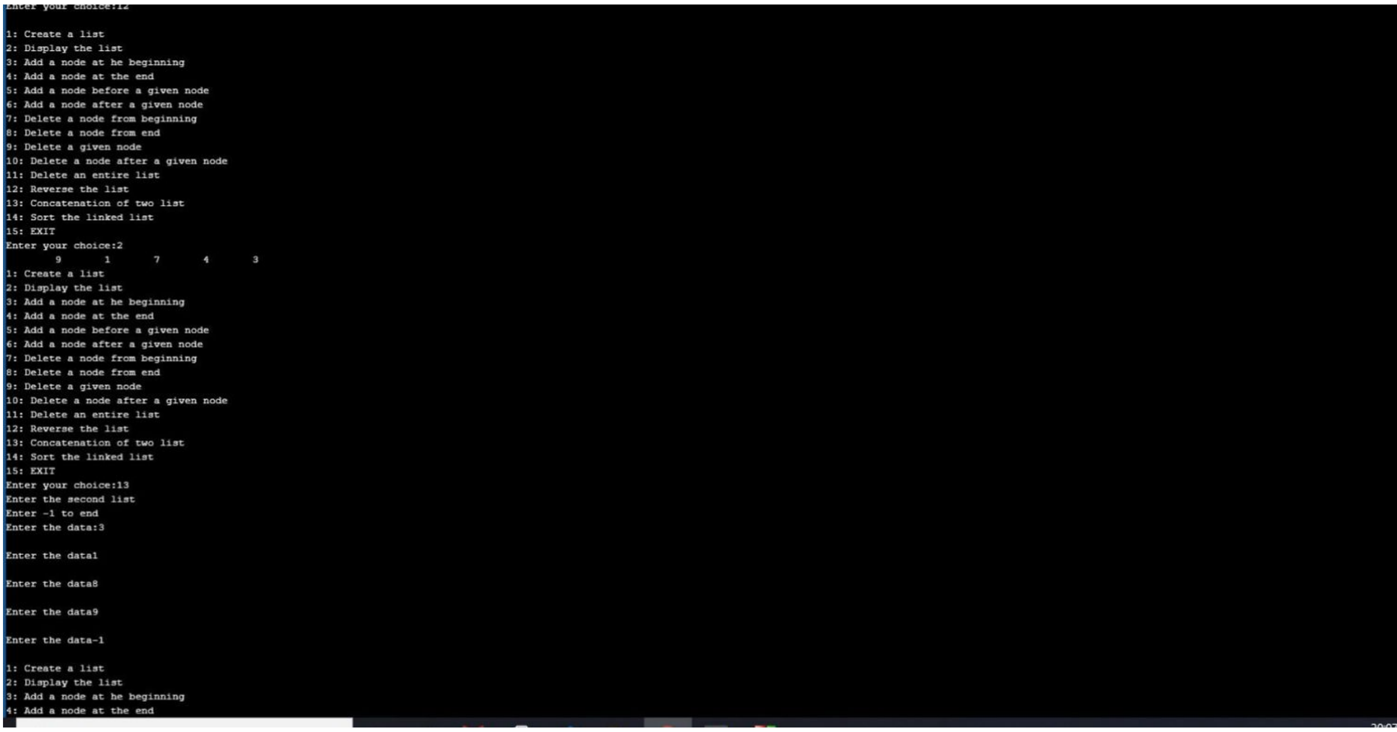
# LAB PROGRAM 7

```
#include<stdlib.h>
                    #include<stdio.h>
                    #include<malloc.h>
                    struct node{
                        int data;
                        struct node *next;
                    };
                    struct node *start=NULL,*start1=NULL;
                    struct node *create_l1(struct node *);
                    struct node *create_l2(struct node *);
                    struct node *display(struct node *);
                    struct node *insert_beg(struct node *);
                    struct node *insert_end(struct node *);
                    struct node *insert_bef(struct node *);
                    struct node *insert_aft(struct node *);
                    struct node *delete_end(struct node *);
                    struct node *delete_beg(struct node *);
                    struct node *delete_node(struct node *);
                    struct node *delete_aft(struct node *);
                    struct node *delete_list(struct node *);
                    struct node *reverse(struct node *);
                    struct node *concat(struct node *,struct node *);
                    struct node *sort_list(struct node *);
                    int main(){
                        int option;
                        do{
                            printf("\n1: Create a list");
                            printf("\n2: Display the list");
                            printf("\n3: Add a node at he beginning");
                            printf("\n4: Add a node at the end");
                            printf("\n5: Add a node before a given node");
                            printf("\n6: Add a node after a given node");
                            printf("\n7: Delete a node from beginning");
                            printf("\n8: Delete a node from end");
                            printf("\n9: Delete a given node");
                            printf("\n10: Delete a node after a given node");
                            printf("\n11: Delete an entire list");
                            printf("\n12: Reverse the list");
                            printf("\n13: Concatenation of two list");
                            printf("\n14: Sort the linked list");
                            printf("\n15: EXIT");
                            printf("\nEnter your choice:");
                            scanf("%d",&option);
                            switch(option){
                                case 1:start=create_l1(start);
                                printf("\nlinked list created!");
                                break;
                                case 2:start=display(start);
                                break;
                                case 3:start=insert_beg(start);
                                break;
                                case 4:start=insert_end(start);
                                break;
                                case 5:start=insert_bef(start);
                                break;
                                case 6:start=insert_aft(start);
                                break;
                                case 7:start=delete_beg(start);
                                break;
                                case 8:start=delete_end(start);
                                break;
                                case 9:start=delete_node(start);
                                break;
                                case 10:start=delete_aft(start);
                                break;
                                case 11:start=delete_list(start);
                                printf("\nLinked list deleted!");
                                break;
                                case 12:start=reverse(start);
                                break;
                                case 13:start=concat(start,start1);
                                break;
                                case 14:start=sort_list(start);
                                break;
                            }
                        }while(option!=15);
                        return 0;
                    }
                    struct node *create_l1(struct node *start){
                        struct node *ptr,*new_node;
                        int num;
                        printf("\nEnter -1 to end");
                        printf("\nEnter the data:");
                        scanf("%d",&num);
                        while(num!=-1){
                            new_node=(struct node *)malloc(sizeof(struct node));
                            new_node->data=num;
                            if(start==NULL){
                                new_node->next=NULL;
                                start=new_node;
                            }
                            else{
                                ptr=start;
                                while(ptr->next!=NULL)
                                    ptr=ptr->next;
                                    ptr->next=new_node;
                                    new_node->next=NULL;

                            }
                            printf("\nEnter the data");
                            scanf("%d",&num);
                        }
                        return start;
                    }
                    struct node *create_l2(struct node *start1){
                        struct node *ptr,*new_node;
                        int num;
                        printf("\nEnter -1 to end");
                        printf("\nEnter the data:");
                        scanf("%d",&num);
                        while(num!=-1){
                            new_node=(struct node *)malloc(sizeof(struct node));
                            new_node->data=num;
                            if(start1==NULL){
                                new_node->next=NULL;
                                start1=new_node;
                            }
                            else{
                                ptr=start1;
                                while(ptr->next!=NULL)
                                    ptr=ptr->next;
                                    ptr->next=new_node;
                                    new_node->next=NULL;

                            }
                            printf("\nEnter the data");
                            scanf("%d",&num);
                        }
                        return start1;
                    }
                    struct node *display(struct node *start){
                        struct node *ptr;
                        ptr=start;
                        while(ptr!=NULL){
                            printf("\t%d",ptr->data);
                            ptr=ptr->next;
                        }
                        return start;
                    }
                    struct node *insert_beg(struct node *start){
                        struct node *new_node;
                        int num;
                        printf("Enter the data:");
                        scanf("%d",&num);
                        new_node=(struct node *)malloc(sizeof(struct node));
                        new_node->data=num;
                        new_node->next=start;
                        start=new_node;
                        return start;
                    }
                    struct node *insert_end(struct node *start){
                        struct node *ptr,*new_node;
                        int num;
                        printf("Enter the data:");
                        scanf("%d",&num);
                        new_node=(struct node *)malloc(sizeof(struct node));
                        new_node->data=num;
                        new_node->next=NULL;
                        ptr=start;
                        while(ptr->next!=NULL)
                            ptr=ptr->next;
                        ptr->next=new_node;
                        return start;
                    }
                    struct node *insert_bef(struct node *start){
                        struct node *ptr,*preptr,*new_node;
                        int num,val;
                        printf("Enter the data:");
                        scanf("%d",&num);
                        printf("Enter the value before which the data has to be 
                    inserted:");
                        scanf("%d",&val);
                        new_node=(struct node *)malloc(sizeof(struct node));
                        new_node->data=num;
                        ptr=start;
                        while(ptr->data!=val){
                            preptr=ptr;
                            ptr=ptr->next;
                        }
                        preptr->next=new_node;
                        new_node->next=ptr;
                        return start;
                    }
                    struct node *insert_aft(struct node *start){
                        struct node *ptr,*preptr,*new_node;
                        int num,val;
                        printf("Enter the data:");
                        scanf("%d",&num);
                        printf("Enter the value after which the data has to be 
                    inserted:");
                        scanf("%d",&val);
                        new_node=(struct node *)malloc(sizeof(struct node));
                        new_node->data=num;
                        ptr=start;
                        preptr=ptr;
                        while(preptr->data!=val){
                            preptr=ptr;
                            ptr=ptr->next;
                        }
                        preptr->next=new_node;
                        new_node->next=ptr;
                        return start;
                    }
```

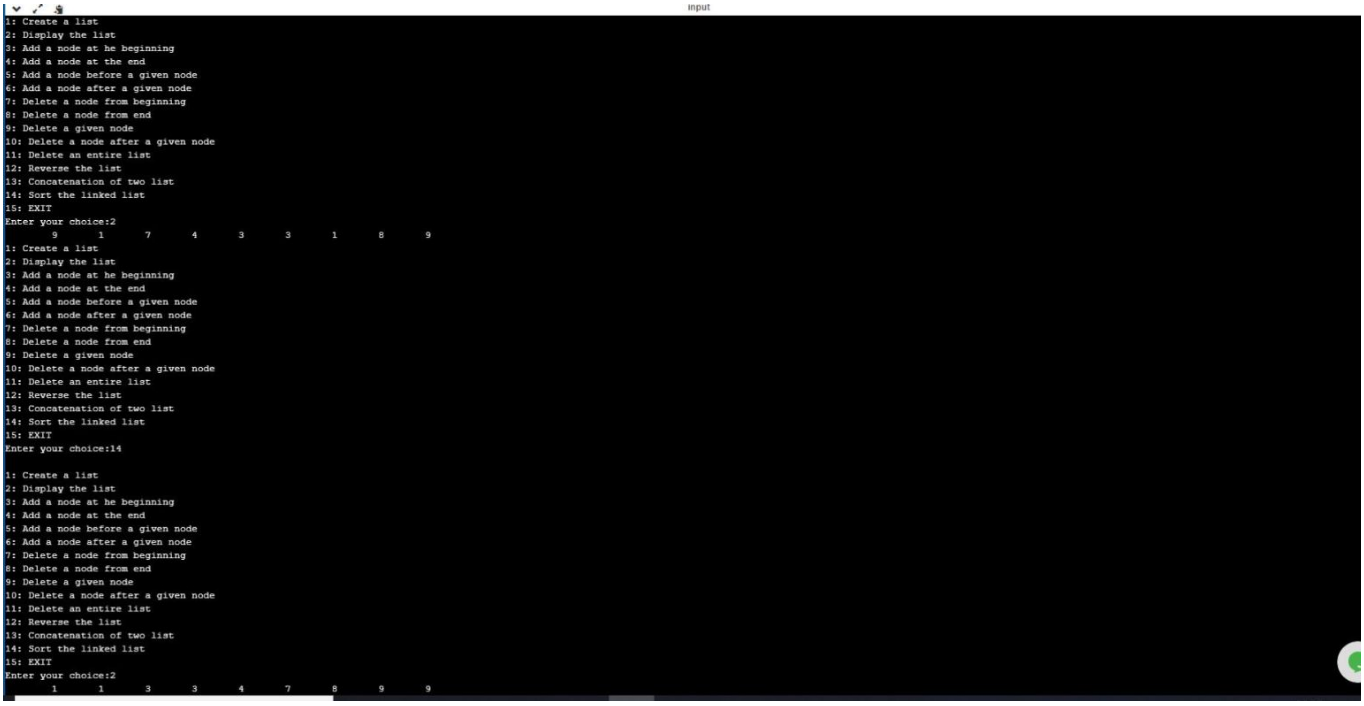
```
struct node *delete_end(struct node *start){
    struct node *ptr,*preptr;
    ptr=start;
    while(ptr->next!=NULL){
        preptr=ptr;
        ptr=ptr->next;
    }
    preptr->next=NULL;
    free(ptr);
    return start;
}
struct node *delete_beg(struct node *start){
    struct node *ptr;
    ptr=start;
    start=start->next;
    free(ptr);
    return start;
}
struct node *delete_node(struct node *start){
    struct node*ptr,*preptr;
    int val;
    printf("Enter the value which has to be deleted");
    scanf("%d",&val);
    ptr=start;
    if(ptr->data==val){
        start=delete_beg(start);
        return start;
    }
    else{
        while(ptr->data!=val){
            preptr=ptr;
            ptr=ptr->next;
        }
        preptr->next=ptr->next;
        free(ptr);
        return start;
    }
}
struct node *delete_aft(struct node *start){
    struct node *ptr,*preptr;
    int val;
    printf("Enter the value after which the node has to be
deleted");
    scanf("%d",&val);
    ptr=start;
    preptr=ptr;
    while(preptr->data!=val){
        preptr=ptr;
        ptr=ptr->next;
    }
    preptr->next=ptr->next;
    free(ptr);
    return start;
}
struct node *delete_list(struct node *start){
    struct node *ptr;
    if(start!=NULL){
        ptr=start;
        while(ptr!=NULL){
            printf("\n%d is to be deleted next",ptr->data);
            start=delete_beg(ptr);
            ptr=start;
        }
    }
    return start;
}
struct node *reverse(struct node *start){
    struct node *prev=NULL,*current=start, *next=NULL;
    while(current!=NULL)
    {
        next=current->next;
        current->next=prev;
        prev=current;
        current=next;
    }
    start=prev;
    return start;
}
struct node *concat(struct node *start,struct node *start1){
    struct node *ptr;
    ptr=start;
    printf("Enter the second list");
    start1=create_l2(start1);
    while(ptr->next!=NULL)
         ptr=ptr->next;

      ptr->next=start1;
      return start;
}
struct node *sort_list(struct node *start){
    struct node *ptr1,*ptr2;
    int temp;
    ptr1=start;
    while(ptr1->next!=NULL){
        ptr2=ptr1->next;
        while(ptr2!=NULL){
            if(ptr1->data>ptr2->data){
                temp=ptr1->data;
                ptr1->data=ptr2->data;
                ptr2->data=temp;
            }
            ptr2=ptr2->next;
        }
        ptr1=ptr1->next;
    }
    return start;
}
```



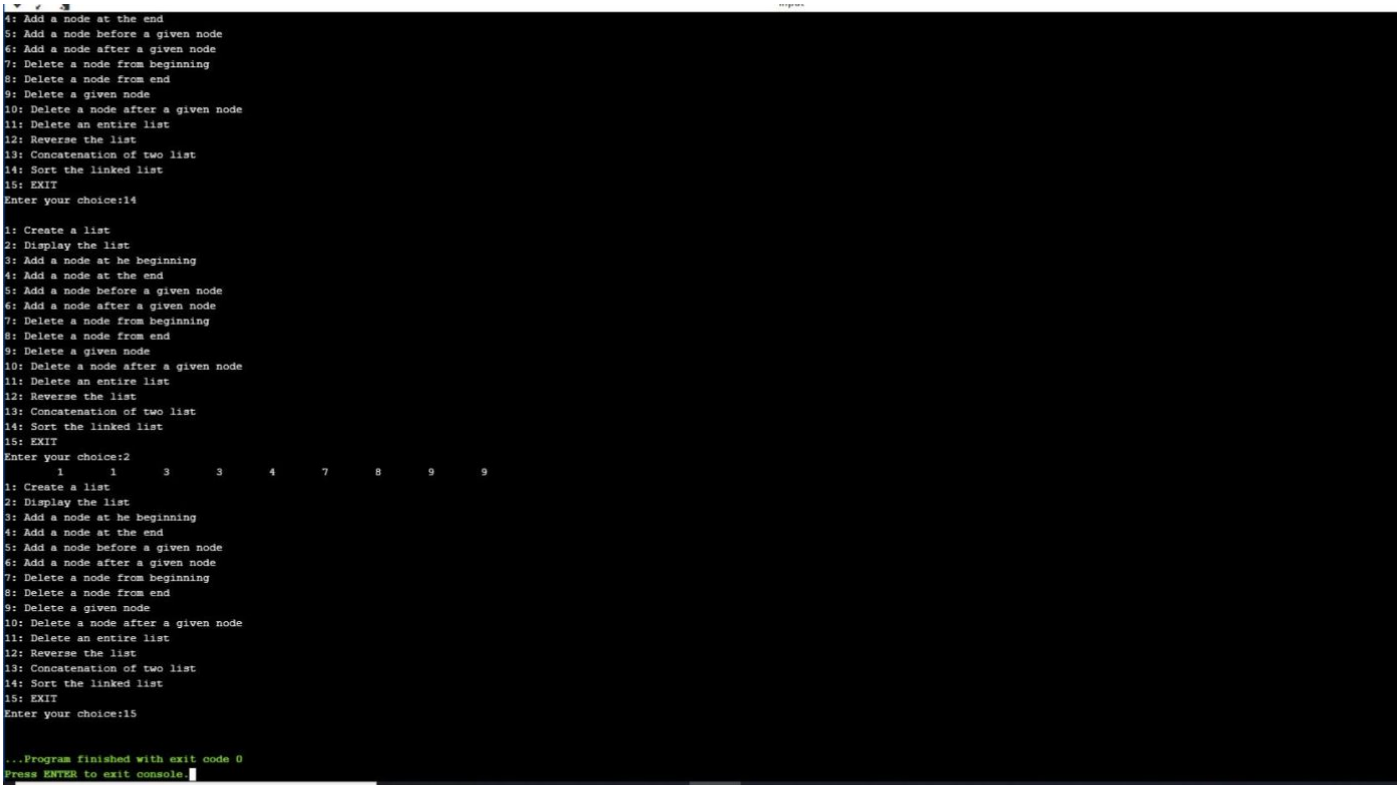
```
1: Create a list
2: Display the list
3: Add a node at he beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from beginning
8: Delete a node from end
9: Delete a given node
10: Delete a node after a given node
11: Delete an entire list
12: Reverse the list
13: Concatenation of two list
14: Sort the linked list
15: EXIT
Enter your choice:1

Enter -1 to end
Enter the data:3

Enter the data4

Enter the data7

Enter the data1

Enter the data9

Enter the data-1

linked list created!
1: Create a list
2: Display the list
3: Add a node at he beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from beginning
8: Delete a node from end
9: Delete a given node
10: Delete a node after a given node
11: Delete an entire list
12: Reverse the list
13: Concatenation of two list
14: Sort the linked list
15: EXIT
Enter your choice:12

1: Create a list
2: Display the list
```



```
Enter your choice:12

1: Create a list
2: Display the list
3: Add a node at he beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from beginning
8: Delete a node from end
9: Delete a given node
10: Delete a node after a given node
11: Delete an entire list
12: Reverse the list
13: Concatenation of two list
14: Sort the linked list
15: EXIT
Enter your choice:2
        9       1       7       4       3
1: Create a list
2: Display the list
3: Add a node at he beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from beginning
8: Delete a node from end
9: Delete a given node
10: Delete a node after a given node
11: Delete an entire list
12: Reverse the list
13: Concatenation of two list
14: Sort the linked list
15: EXIT
Enter your choice:13
Enter the second list
Enter -1 to end
Enter the data:3

Enter the data1

Enter the data8

Enter the data9

Enter the data-1

1: Create a list
2: Display the list
3: Add a node at he beginning
4: Add a node at the end
```



```
1: Create a list
2: Display the list
3: Add a node at he beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from beginning
8: Delete a node from end
9: Delete a given node
10: Delete a node after a given node
11: Delete an entire list
12: Reverse the list
13: Concatenation of two list
14: Sort the linked list
15: EXIT
Enter your choice:2
        9       1       7       4       3       3       1       8       9
1: Create a list
2: Display the list
3: Add a node at he beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from beginning
8: Delete a node from end
9: Delete a given node
10: Delete a node after a given node
11: Delete an entire list
12: Reverse the list
13: Concatenation of two list
14: Sort the linked list
15: EXIT
Enter your choice:14

1: Create a list
2: Display the list
3: Add a node at he beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from beginning
8: Delete a node from end
9: Delete a given node
10: Delete a node after a given node
11: Delete an entire list
12: Reverse the list
13: Concatenation of two list
14: Sort the linked list
15: EXIT
Enter your choice:2
        1       1       3       3       4       7       8       9       9
```



```
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from beginning
8: Delete a node from end
9: Delete a given node
10: Delete a node after a given node
11: Delete an entire list
12: Reverse the list
13: Concatenation of two list
14: Sort the linked list
15: EXIT
Enter your choice:14

1: Create a list
2: Display the list
3: Add a node at he beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from beginning
8: Delete a node from end
9: Delete a given node
10: Delete a node after a given node
11: Delete an entire list
12: Reverse the list
13: Concatenation of two list
14: Sort the linked list
15: EXIT
Enter your choice:2
        1       1       3       3       4       7       8       9       9
1: Create a list
2: Display the list
3: Add a node at he beginning
4: Add a node at the end
5: Add a node before a given node
6: Add a node after a given node
7: Delete a node from beginning
8: Delete a node from end
9: Delete a given node
10: Delete a node after a given node
11: Delete an entire list
12: Reverse the list
13: Concatenation of two list
14: Sort the linked list
15: EXIT
Enter your choice:15


...Program finished with exit code 0
Press ENTER to exit console.
```

# LAB PROGRAM 8

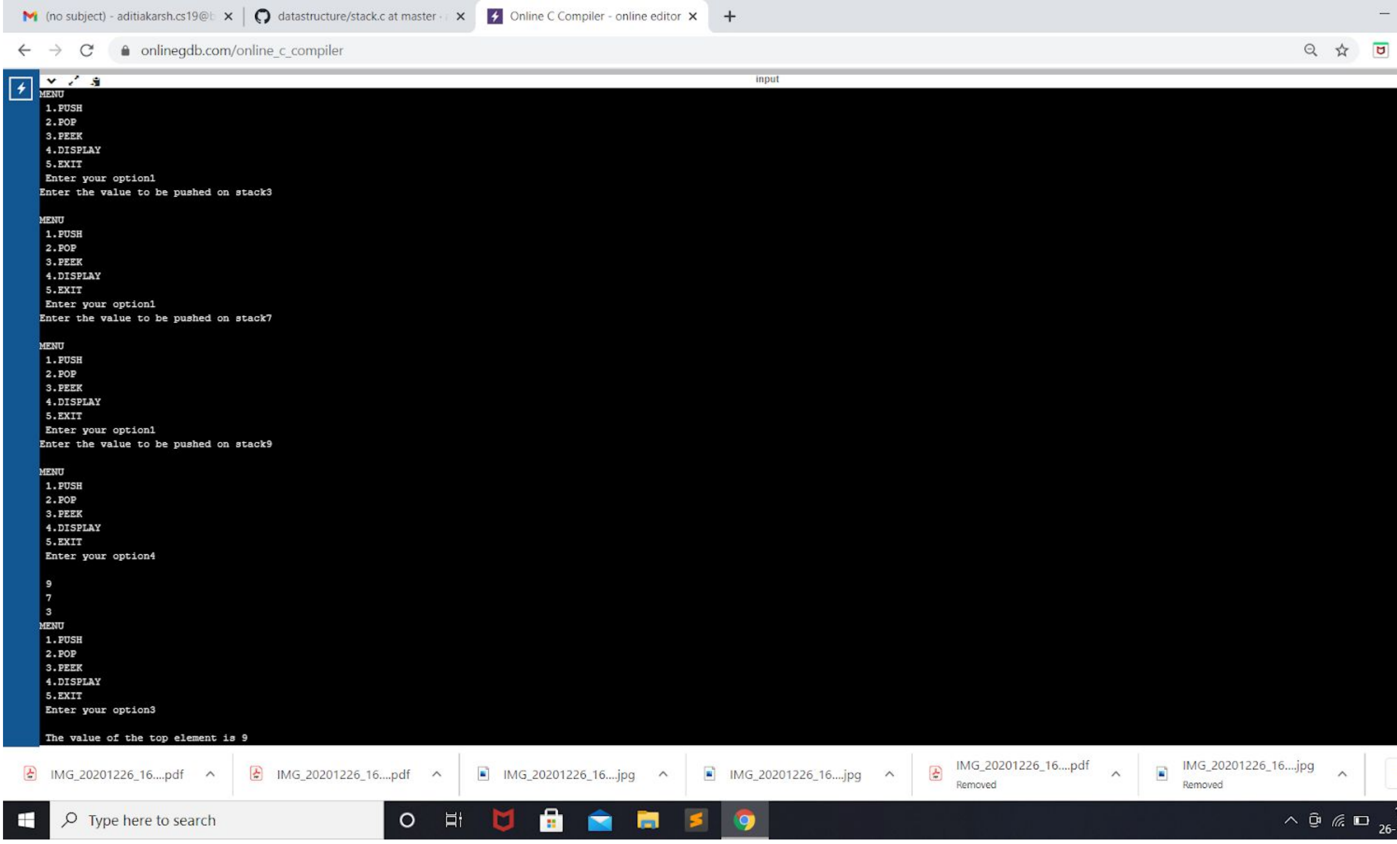
```
#include<stdio.h
>
                    #include<stdlib.h>
                    struct node
                    {
                        int data;
```

```
    struct node *next;
};
struct node *front;
struct node *rear;
void insert();
void delete();
void display();
void main ()
{
    int choice;
    while(choice != 4)
    {
        printf("\n*Menu*");
        printf("\n 1.insert an element\n 2.Delete an element\n
3.Display the queue\n   4.Exit\n  ");
        printf("\nEnter your choice");
        scanf("%d",& choice);
        switch(choice)
        {
            case 1:
            insert();
            break;
            case 2:
            delete();
            break;
            case 3:
            display();
            break;
            case 4:
            exit(0);
            break;
            default:
            printf("\nEnter valid choice\n");
        }
    }
}
void insert()
{
    struct node *ptr;
    int item;


    ptr = (struct node *) malloc (sizeof(struct node));
    if(ptr == NULL)
    {
        printf("\nOVERFLOW\n");
        return;
    }
    else
    {
        printf("\nEnter value\n");
        scanf("%d",&item);
        ptr -> data = item;
        if(front == NULL)
        {
            front = ptr;
            rear = ptr;
            front -> next = NULL;
            rear -> next = NULL;
        }
        else
```

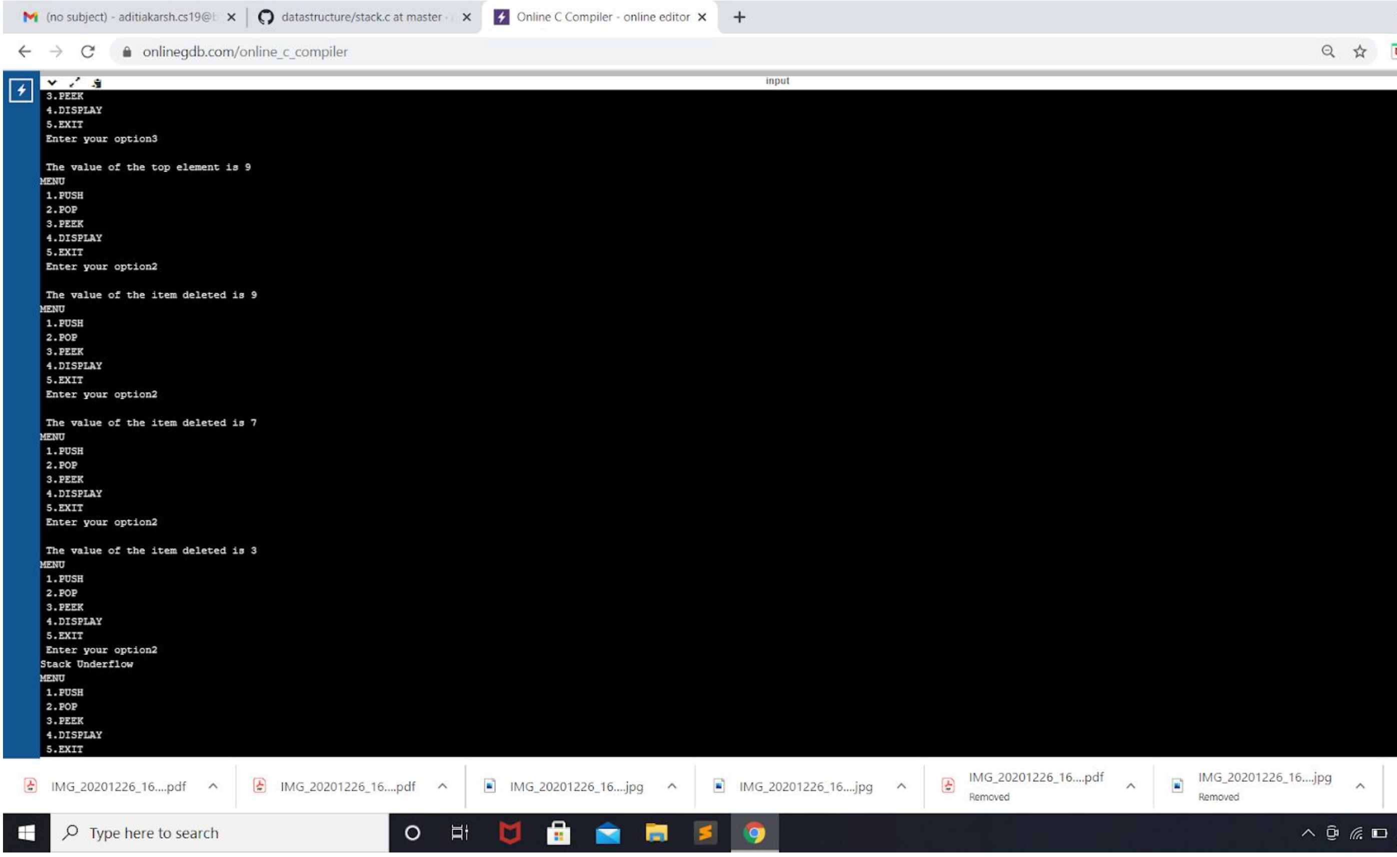
```
                    {
                        rear -> next = ptr;
                        rear = ptr;
                        rear->next = NULL;
                    }
                }
            }
            void delete ()
            {
                struct node *ptr;
                if(front == NULL)
                {
                    printf("\nUNDERFLOW\n");
                    return;
                }
                else
                {
                    ptr = front;
                    front = front -> next;
                    free(ptr);
                }
            }
            void display()
            {
                struct node *ptr;
                ptr = front;
                if(front == NULL)
                {
                    printf("\nEmpty queue\n");
                }
                else
                {   printf("\n values are .....\n");
                    while(ptr != NULL)
                    {
                        printf("\n%d\n",ptr -> data);
                        ptr = ptr -> next;
                    }
                }
            }
```

---

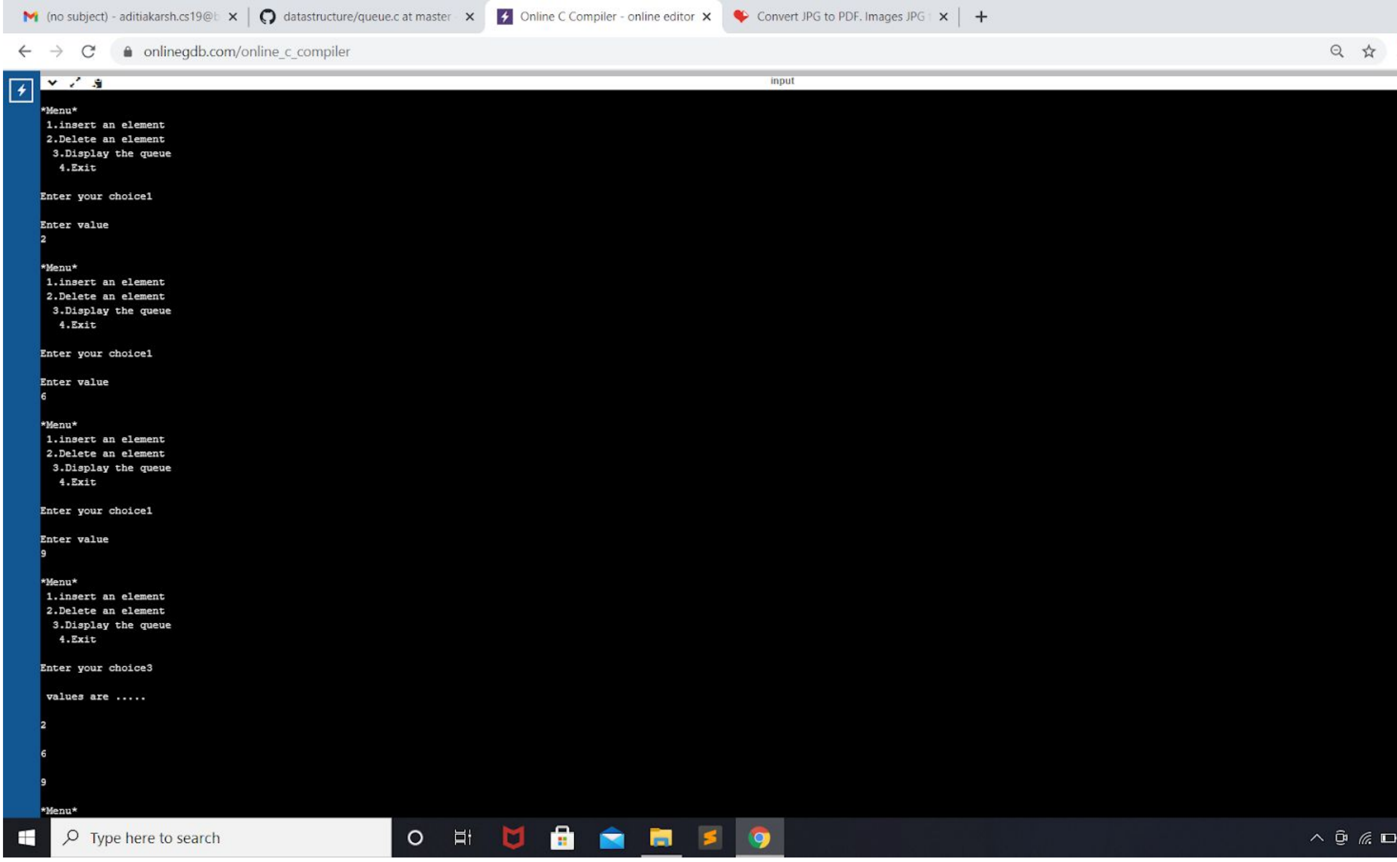
```
#include
<stdio.h
>
            #include <stdlib.h>
            #include <malloc.h>


            struct stack{
            int data;
            struct node *next;
            };


            struct stack *top=NULL;


            struct stack *push(struct stack *,int );
            struct stack *display(struct stack * );
            struct stack *pop(struct stack * );
            int peek(struct stack *);


            int main(){
            int val,option;
            do{printf("\nMENU");
             printf("\n 1.PUSH");
            printf("\n 2.POP");
            printf("\n 3.PEEK");
            printf("\n 4.DISPLAY");
            printf("\n 5.EXIT");
            printf("\n Enter your option");
            scanf("%d",& option);
            switch(option){
            case 1:
              printf("Enter the value to be pushed on stack");
              scanf("%d",&val);
              top=push(top,val);
              break;
            case 2:
              top=pop(top);
              break;
            case 3:
              val=peek(top);
              if(val!=-1)
                printf("\n The value of the top element is %d ",val);
              else
                printf("\n Stack is EMPTY");
              break;
            case 4:
              top=display(top);
              break;
            }
            }while(option!=5);
            return 0;
            }


            struct stack *push (struct stack *top,int val){
            struct stack *p;
            p=(struct stack *)malloc (sizeof(struct stack));
            p->data=val;
            if(top==0){
              p->next=0;
              top=p;
            }
            else {
              p->next=top;
              top=p;
            }
            return top;
            }


            struct stack *display (struct stack *top){
            struct stack *p;
            p=top;
            if(top==NULL)
              printf("\n Stack is Empty");
            else {
              while(p!=NULL)
              {
                printf("\n %d",p->data);
                p=p->next;
              }
            }
            return top;
            }


            struct stack *pop(struct stack *top){
            struct stack *p;
            p=top;
            if(top==0)
              printf("Stack Underflow");
            else {
              top=top->next;
              printf("\n The value of the item deleted is
            %d",p->data);
              free(p);
            }
            return top;
            }


            int peek (struct stack *top ){
            if(top==NULL)
              return -1;
            else
              return top->data;
            }
```



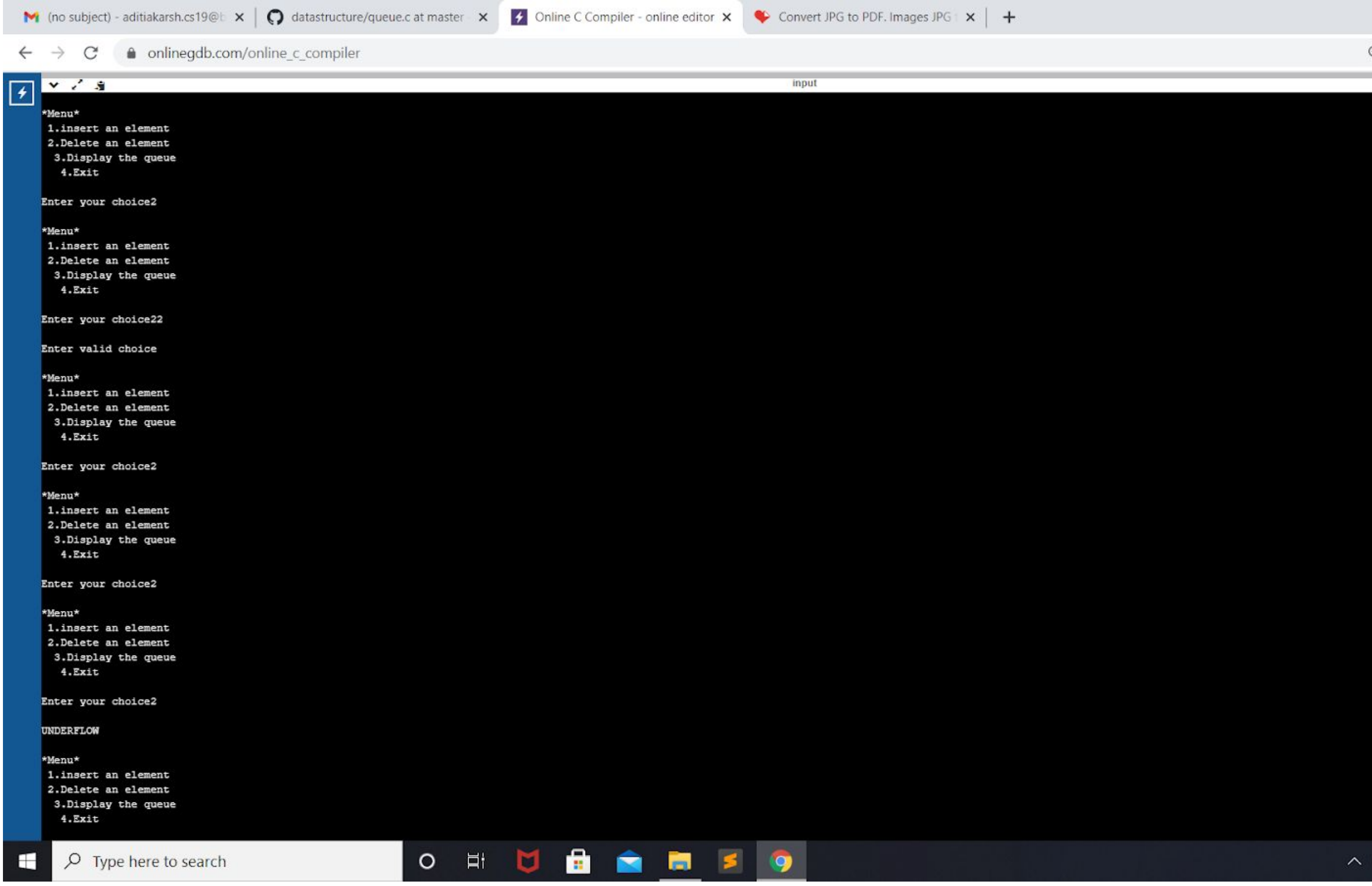
```
MENU
 1.PUSH
 2.POP
 3.PEEK
 4.DISPLAY
 5.EXIT
 Enter your option1
Enter the value to be pushed on stack3

MENU
 1.PUSH
 2.POP
 3.PEEK
 4.DISPLAY
 5.EXIT
 Enter your option1
Enter the value to be pushed on stack7

MENU
 1.PUSH
 2.POP
 3.PEEK
 4.DISPLAY
 5.EXIT
 Enter your option1
Enter the value to be pushed on stack9

MENU
 1.PUSH
 2.POP
 3.PEEK
 4.DISPLAY
 5.EXIT
 Enter your option4

 9
 7
 3
MENU
 1.PUSH
 2.POP
 3.PEEK
 4.DISPLAY
 5.EXIT
 Enter your option3

 The value of the top element is 9
```



```
 3.PEEK
 4.DISPLAY
 5.EXIT
 Enter your option3

 The value of the top element is 9
MENU
 1.PUSH
 2.POP
 3.PEEK
 4.DISPLAY
 5.EXIT
 Enter your option2

 The value of the item deleted is 9
MENU
 1.PUSH
 2.POP
 3.PEEK
 4.DISPLAY
 5.EXIT
 Enter your option2

 The value of the item deleted is 7
MENU
 1.PUSH
 2.POP
 3.PEEK
 4.DISPLAY
 5.EXIT
 Enter your option2

 The value of the item deleted is 3
MENU
 1.PUSH
 2.POP
 3.PEEK
 4.DISPLAY
 5.EXIT
 Enter your option2
Stack Underflow
MENU
 1.PUSH
 2.POP
 3.PEEK
 4.DISPLAY
 5.EXIT
```


(no subject) - aditiakarsh.cs19@
datastructure/queue.c at master
Online C Compiler - online editor
Convert JPG to PDF. Images JPG
onlinegdb.com/online_c_compiler
input
Type here to search

```
*Menu*
 1.insert an element
 2.Delete an element
  3.Display the queue
   4.Exit

Enter your choice1

Enter value
2

*Menu*
 1.insert an element
 2.Delete an element
  3.Display the queue
   4.Exit

Enter your choice1

Enter value
6

*Menu*
 1.insert an element
 2.Delete an element
  3.Display the queue
   4.Exit

Enter your choice1

Enter value
9

*Menu*
 1.insert an element
 2.Delete an element
  3.Display the queue
   4.Exit

Enter your choice3

 values are .....

2

6

9

*Menu*
```


(no subject) - aditiakarsh.cs19@
datastructure/queue.c at master
Online C Compiler - online editor
Convert JPG to PDF. Images JPG
onlinegdb.com/online_c_compiler
input
Type here to search

```
*Menu*
 1.insert an element
 2.Delete an element
  3.Display the queue
   4.Exit

Enter your choice2

*Menu*
 1.insert an element
 2.Delete an element
  3.Display the queue
   4.Exit

Enter your choice22

Enter valid choice

*Menu*
 1.insert an element
 2.Delete an element
  3.Display the queue
   4.Exit

Enter your choice2

*Menu*
 1.insert an element
 2.Delete an element
  3.Display the queue
   4.Exit

Enter your choice2

*Menu*
 1.insert an element
 2.Delete an element
  3.Display the queue
   4.Exit

Enter your choice2

UNDERFLOW

*Menu*
 1.insert an element
 2.Delete an element
  3.Display the queue
   4.Exit
```

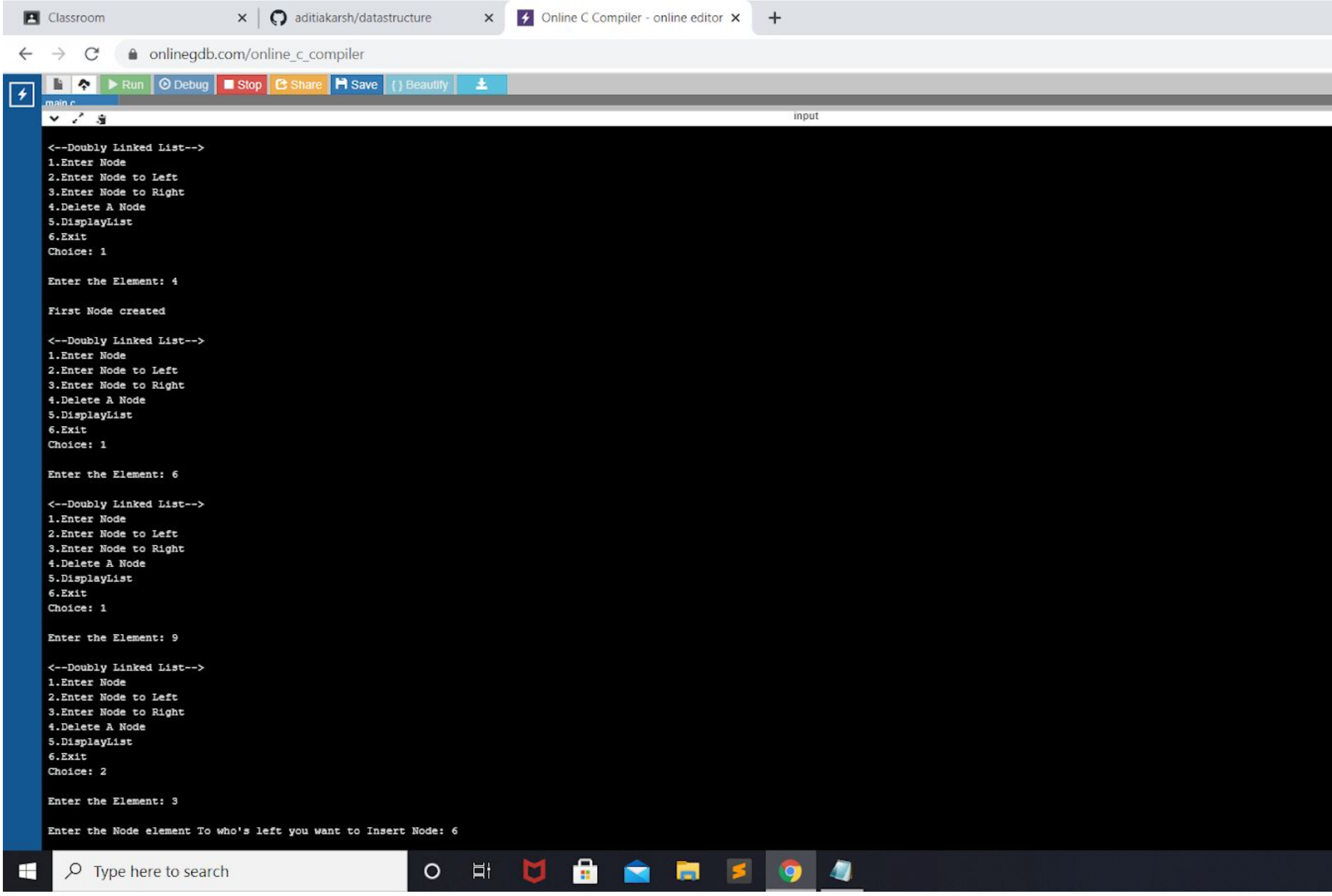
# LAB PROGRAM 9

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


typedef struct node
{
    int data;
    struct node* prev;
    struct node *next;
}Node;


Node *head=NULL;


void doublyLinkedList();
void insertNode(int);
void insertNodeToLeft();
void insertNodeToRight();
void deleteSpecifiedValue();
void displayList();


int main()
{
    doublyLinkedList();
    return 0;
}


void doublyLinkedList()
{
    int choice=0;
    printf("\n <--Doubly Linked List-->");
    printf("\n 1.Enter Node\n 2.Enter Node to Left\n 3.Enter Node to
Right\n 4.Delete A Node\n 5.DisplayList\n 6.Exit\n Choice: ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1: insertNode(0);
                break;
        case 2: insertNode(1);
                break;
        case 3: insertNode(2);
                break;
        case 4: deleteSpecifiedValue();
                break;
        case 5: displayList();
                break;
        case 6: exit(0);


        default: printf("\n Error choice, Try Again! ");
                    doublyLinkedList();
    }
    doublyLinkedList();
```

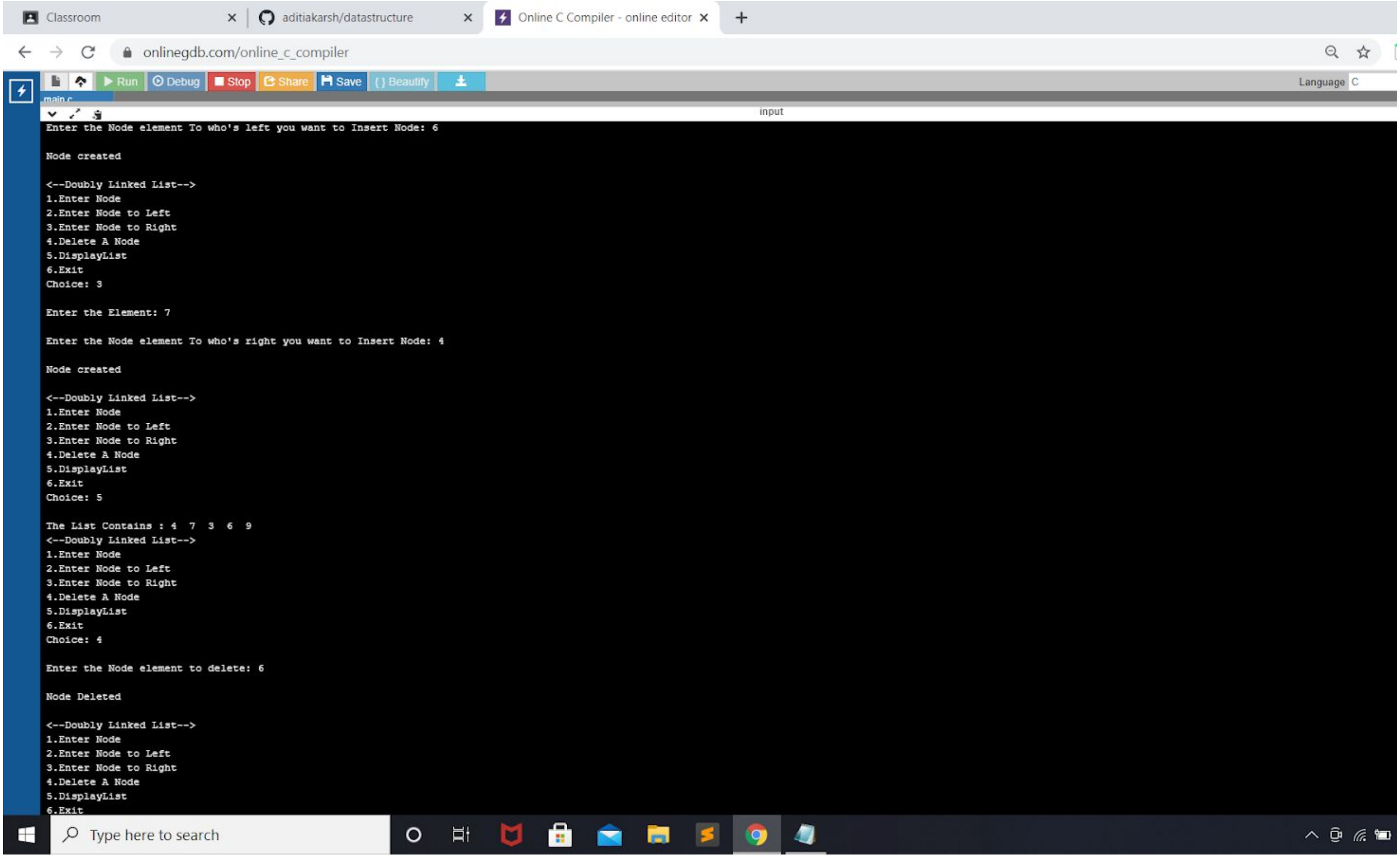
```
}


void insertNode(int flag)
{
    Node *newnode;
    newnode=(Node*)malloc(sizeof(Node));
    printf("\n Enter the Element: ");
    scanf("%d",&newnode->data);
    if(head==NULL)
    {
        head=newnode;
        newnode->next=NULL;
        newnode->prev=NULL;
        printf("\n First Node created \n");
        doublyLinkedList();
    }
    if(flag==0)
    {
         Node *temp=head;
         for(temp;(temp->next)!=NULL;temp=temp->next);
         temp->next=newnode;
         newnode->prev=temp;
         newnode->next=NULL;
    }
    else
        if(flag==1)
        insertNodeToLeft(newnode);
    else
        insertNodeToRight(newnode);
}
void insertNodeToRight(Node *tempNew)
{
    int ele;
    char choice;
    printf("\n Enter the Node element To who's right you want to
Insert Node: ");
    scanf("%d",&ele);
    Node *temp=head;
    for(temp;temp!=NULL;temp=temp->next)
        {
            if(temp->data==ele)
            {
                  if(temp->next!=NULL)
                  {
                        tempNew->next=temp->next;
                        tempNew->prev=temp;
                        (temp->next)->prev=tempNew;
                        temp->next=tempNew;
                        printf("\n Node created \n");
                        doublyLinkedList();
                    }
                    else
                    {
                        tempNew->next=NULL;
                        tempNew->prev=temp;
                        temp->next=tempNew;
                        printf("\n Node created \n");
                        doublyLinkedList();
                    }
            }
        }
    printf("\n The given Element was not found! ,press Y to Try
again! or press anything to exit: ");
    fflush(stdin);
    scanf("%c",&choice);
    if(choice=='Y' || choice =='y')
    insertNodeToRight(tempNew);
    else
    {
        free(tempNew);
        printf("\n Node creation Failed \n");
        doublyLinkedList();
    }


}
void insertNodeToLeft(Node *tempNew)
{
    int ele;
    char choice;
    printf("\n Enter the Node element To who's left you want to
Insert Node: ");
    scanf("%d",&ele);
    Node *temp=head;
    if(head->data==ele)
    {
        tempNew->next=head;
        tempNew->prev=NULL;
        head=tempNew;
        printf("\n Node created \n");
        doublyLinkedList();
    }
    for(temp;temp!=NULL;temp=temp->next)
        {
            if(temp->data==ele)
            {
                    tempNew->next=temp;
                    tempNew->prev=temp->prev;
                    (temp->prev)->next=tempNew;
                    temp->prev=tempNew;
                    printf("\n Node created \n");
                    doublyLinkedList();


            }
        }
    printf("\n The given Element was not found! ,press Y to Try
again! or press anything to exit: ");
    fflush(stdin);
    scanf("%c",&choice);
    if(choice=='Y' || choice =='y')
    insertNodeToLeft(tempNew);
    else
    {
        free(tempNew);
        printf("\n Node creation Failed \n");
        doublyLinkedList();
    }
```

```
}
void deleteSpecifiedValue()
{
    if(head==NULL)
    {
        printf("\n Empty List!\n");
        doublyLinkedList();
    }
    int ele;
    printf("\n Enter the Node element to delete: ");
    scanf("%d",&ele);
    Node *temp=head;
    if(head->next==NULL)
    {
        if(head->data==ele)
        {
            free(temp);
            head=NULL;
            printf("\n Node Deleted. \n Now List Is empty! ");
            doublyLinkedList();
        }
    }
     else
     {
         for(temp;temp!=NULL;temp=temp->next)
        {
            if(temp->data==ele)
            {
                    if(temp->next==NULL)
                    {
                        (temp->prev)->next=NULL;
                        free(temp);
                        printf("\n Node Deleted \n");
                        doublyLinkedList();
                    }
                    if(temp->prev==NULL)
                    {
                        (temp->next)->prev=NULL;
                        head=head->next;
                        printf("\n Node Deleted \n");
                        doublyLinkedList();
                    }
                    else
                    {
                        (temp->prev)->next=temp->next;
                        (temp->next)->prev=temp->prev;
                        free(temp);
                        printf("\n Node Deleted \n");
                        doublyLinkedList();
                    }
            }
        }
     }
     printf("\n The given element %d is not present in list!\n
",ele);
}


void displayList()
```

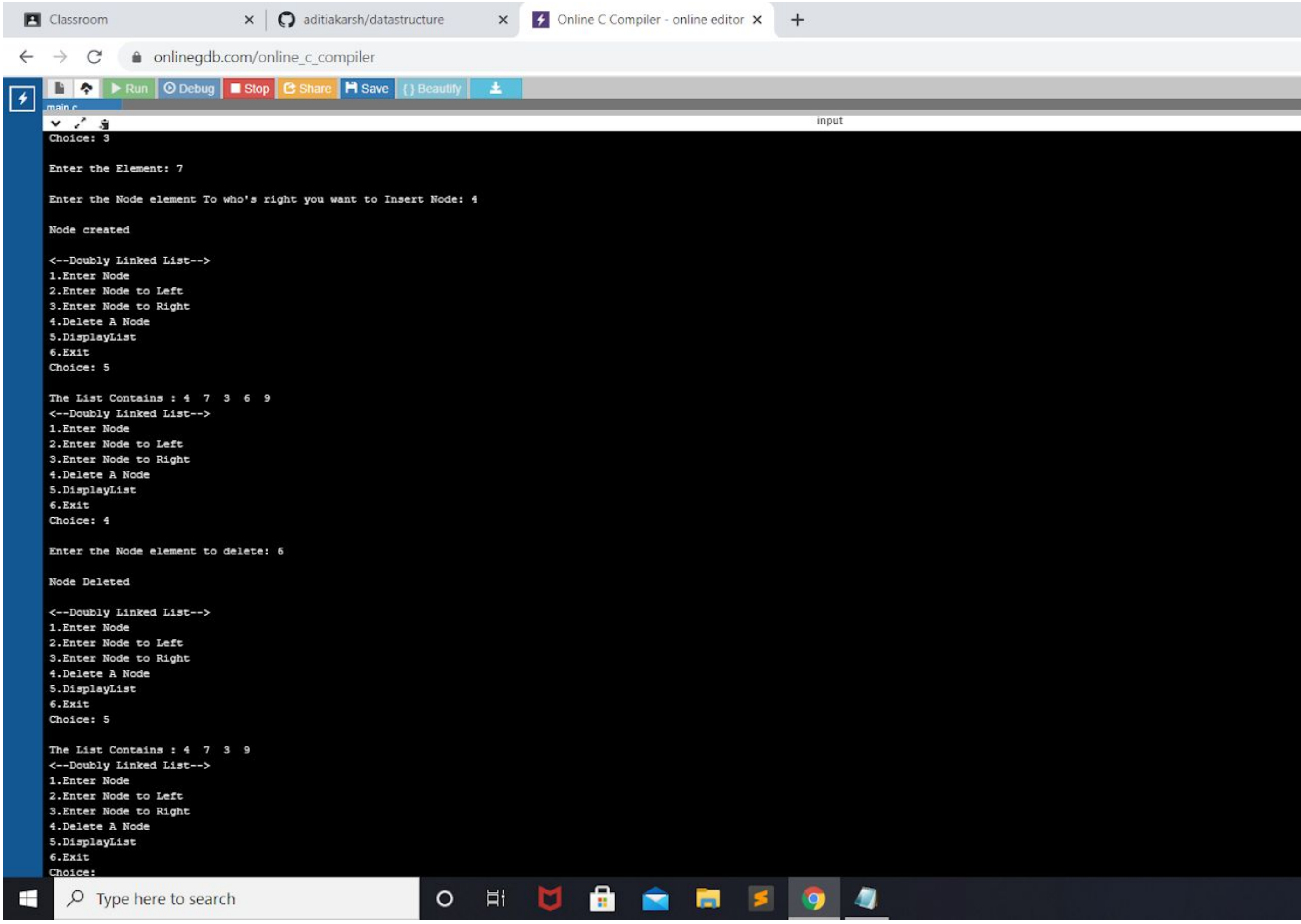
```
{
    if(head==NULL)
    {
        printf("\n Empty List!\n");
        doublyLinkedList();
    }
    Node *temp=head;
    printf("\n The List Contains :");
    for(temp;temp!=NULL;temp=temp->next)
      {
        printf(" %d ",temp->data);
      }
    doublyLinkedList();
}
```


Classroom
aditiakarsh/datastructure
Online C Compiler - online editor
onlinegdb.com/online_c_compiler
Run
Debug
Stop
Share
Save
{ } Beautify
main.c
input
Type here to search

```
<--Doubly Linked List-->
1.Enter Node
2.Enter Node to Left
3.Enter Node to Right
4.Delete A Node
5.DisplayList
6.Exit
Choice: 1

Enter the Element: 4

First Node created

<--Doubly Linked List-->
1.Enter Node
2.Enter Node to Left
3.Enter Node to Right
4.Delete A Node
5.DisplayList
6.Exit
Choice: 1

Enter the Element: 6

<--Doubly Linked List-->
1.Enter Node
2.Enter Node to Left
3.Enter Node to Right
4.Delete A Node
5.DisplayList
6.Exit
Choice: 1

Enter the Element: 9

<--Doubly Linked List-->
1.Enter Node
2.Enter Node to Left
3.Enter Node to Right
4.Delete A Node
5.DisplayList
6.Exit
Choice: 2

Enter the Element: 3

Enter the Node element To who's left you want to Insert Node: 6
```


Classroom
aditiakarsh/datastructure
Online C Compiler - online editor
onlinegdb.com/online_c_compiler
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
input
Type here to search

```
Enter the Node element To who's left you want to Insert Node: 6

Node created

<--Doubly Linked List-->
1.Enter Node
2.Enter Node to Left
3.Enter Node to Right
4.Delete A Node
5.DisplayList
6.Exit
Choice: 3

Enter the Element: 7

Enter the Node element To who's right you want to Insert Node: 4

Node created

<--Doubly Linked List-->
1.Enter Node
2.Enter Node to Left
3.Enter Node to Right
4.Delete A Node
5.DisplayList
6.Exit
Choice: 5

The List Contains : 4  7  3  6  9
<--Doubly Linked List-->
1.Enter Node
2.Enter Node to Left
3.Enter Node to Right
4.Delete A Node
5.DisplayList
6.Exit
Choice: 4

Enter the Node element to delete: 6

Node Deleted

<--Doubly Linked List-->
1.Enter Node
2.Enter Node to Left
3.Enter Node to Right
4.Delete A Node
5.DisplayList
6.Exit
```


Classroom
aditiakarsh/datastructure
Online C Compiler - online editor
onlinegdb.com/online_c_compiler
Run
Debug
Stop
Share
Save
{} Beautify
main.c
input
Type here to search

```
Choice: 3

Enter the Element: 7

Enter the Node element To who's right you want to Insert Node: 4

Node created

<--Doubly Linked List-->
1.Enter Node
2.Enter Node to Left
3.Enter Node to Right
4.Delete A Node
5.DisplayList
6.Exit
Choice: 5

The List Contains : 4  7  3  6  9
<--Doubly Linked List-->
1.Enter Node
2.Enter Node to Left
3.Enter Node to Right
4.Delete A Node
5.DisplayList
6.Exit
Choice: 4

Enter the Node element to delete: 6

Node Deleted

<--Doubly Linked List-->
1.Enter Node
2.Enter Node to Left
3.Enter Node to Right
4.Delete A Node
5.DisplayList
6.Exit
Choice: 5

The List Contains : 4  7  3  9
<--Doubly Linked List-->
1.Enter Node
2.Enter Node to Left
3.Enter Node to Right
4.Delete A Node
5.DisplayList
6.Exit
Choice:
```

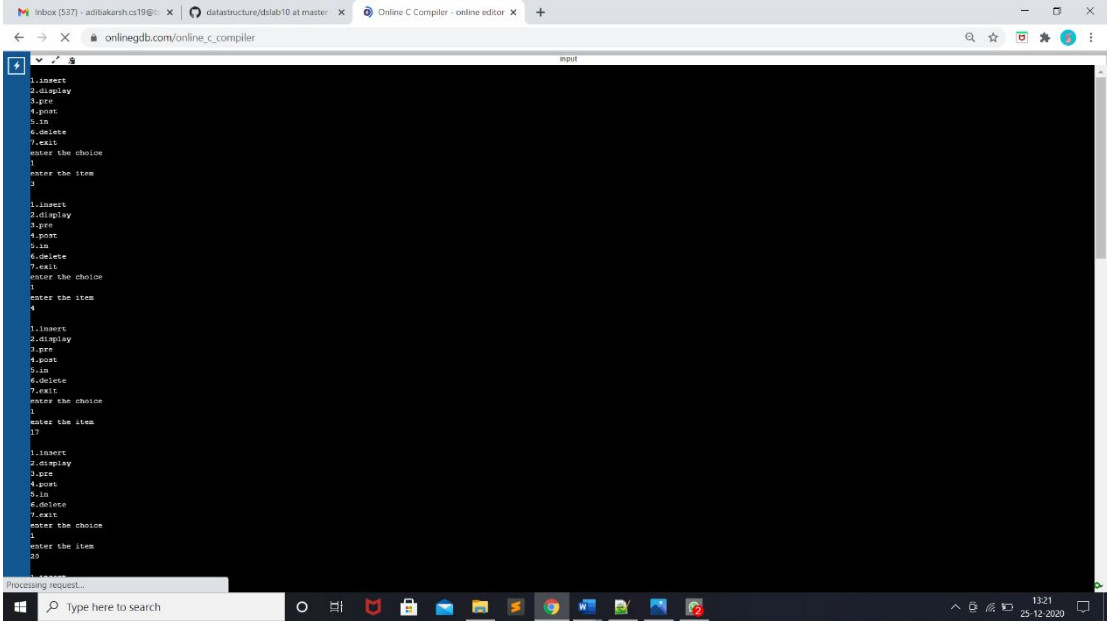
# LAB PROGRAM 10

```
#include<stdio
.h>
                         #include<stdlib.h>
                         #include<conio.h>
                         struct node
                          {
                           int info;
                           struct node *rlink;
                           struct node *llink;
                          };
                         typedef struct node *NODE;
                         NODE getnode()
                         {
                         NODE x;
                         x=(NODE)malloc(sizeof(struct node));
                         if(x==NULL)
                          {
                           printf("mem full\n");
                           exit(0);
                          }
```

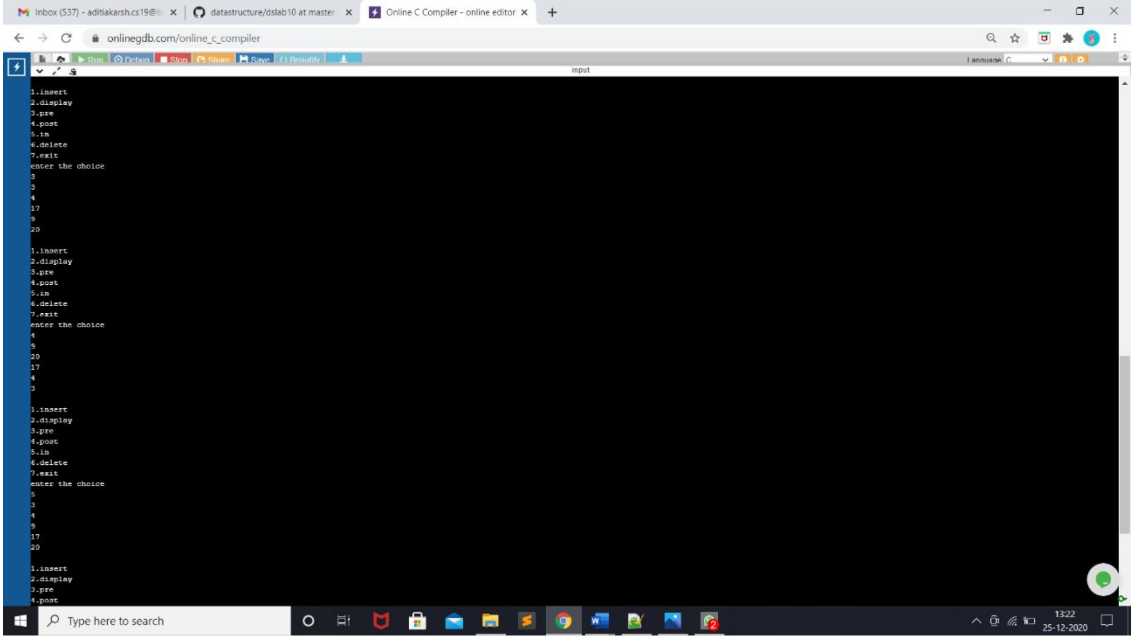
```
 return x;
}
void freenode(NODE x)
{
free(x);
}
NODE insert(NODE root,int item)
{
NODE temp,cur,prev;
temp=getnode();
temp->rlink=NULL;
temp->llink=NULL;
temp->info=item;
if(root==NULL)
 return temp;
prev=NULL;
cur=root;
while(cur!=NULL)
{
prev=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(item<prev->info)
 prev->llink=temp;
else
 prev->rlink=temp;
return root;
}
void display(NODE root,int i)
{
int j;
if(root!=NULL)
 {
  display(root->rlink,i+1);
  for(j=0;j<i;j++)
         printf("  ");
   printf("%d\n",root->info);
        display(root->llink,i+1);
 }
}
NODE delete(NODE root,int item)
{
NODE cur,parent,q,suc;
if(root==NULL)
{
printf("empty\n");
return root;
}
parent=NULL;
cur=root;
while(cur!=NULL&&item!=cur->info)
{
parent=cur;
cur=(item<cur->info)?cur->llink:cur->rlink;
}
if(cur==NULL)
{
 printf("not found\n");
 return root;
}
if(cur->llink==NULL)
```

```
 q=cur->rlink;
else if(cur->rlink==NULL)
 q=cur->llink;
else
 {
 suc=cur->rlink;
 while(suc->llink!=NULL)
  suc=suc->llink;
 suc->llink=cur->llink;
 q=cur->rlink;
 }
 if(parent==NULL)
  return q;
 if(cur==parent->llink)
  parent->llink=q;
 else
  parent->rlink=q;
 freenode(cur);
 return root;
 }



void preorder(NODE root)
{
if(root!=NULL)
 {
  printf("%d\n",root->info);
  preorder(root->llink);
  preorder(root->rlink);
  }
 }
void postorder(NODE root)
{
if(root!=NULL)
 {



  postorder(root->llink);
  postorder(root->rlink);
  printf("%d\n",root->info);
  }
 }
void inorder(NODE root)
{
if(root!=NULL)
 {



  inorder(root->llink);
  printf("%d\n",root->info);
  inorder(root->rlink);
  }
 }
void main()
{
int item,choice;
NODE root=NULL;
```

```
        for(;;)
        {

printf("\n1.insert\n2.display\n3.pre\n4.post\n5.in\n6.delete\n7.exit\n
");
        printf("enter the choice\n");
        scanf("%d",&choice);
        switch(choice)
         {
          case 1:printf("enter the item\n");
                        scanf("%d",&item);
                        root=insert(root,item);
                        break;
          case 2:display(root,0);
                        break;
          case 3:preorder(root);
                        break;
          case 4:postorder(root);
                        break;
          case 5:inorder(root);
                        break;
          case 6:printf("enter the item\n");
                        scanf("%d",&item);
                        root=delete(root,item);
                        break;
          default:exit(0);
                         break;
                 }
               }
         }
```

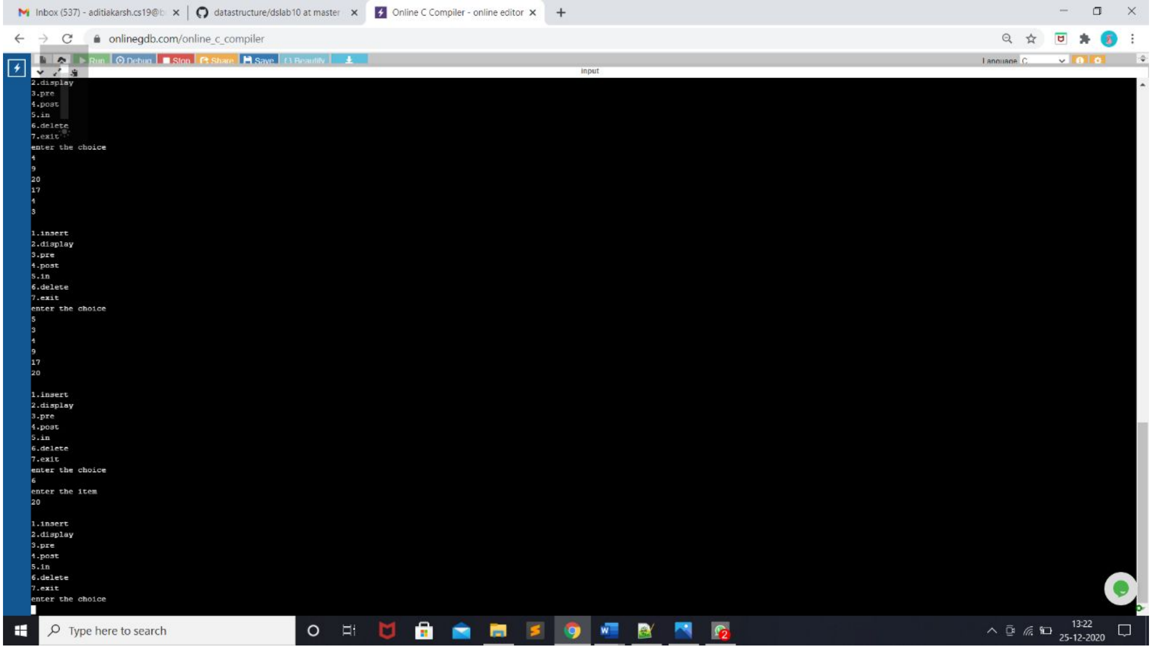

Inbox (537) - aditiakarsh.cs19@
datastructure/dslab10 at master
Online C Compiler - online editor
onlinegdb.com/online_c_compiler
input
Processing request...
Type here to search
13:21
25-12-2020

```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
3

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
4

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
17

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
1
enter the item
20

1.insert
```



Inbox (537) - aditiakarsh.cs19@b
datastructure/dslab10 at master
Online C Compiler - online editor
onlinegdb.com/online_c_compiler
Run
Debug
Stop
Share
Save
Beautify
Language
input
Type here to search
13:22
25-12-2020

```
1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
3
3
4
17
9
20

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
9
20
17
4
3

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
5
3
4
9
17
20

1.insert
2.display
3.pre
4.post
```



Inbox (537) - aditiakarsh.cs19@b
datastructure/dslab10 at master
Online C Compiler - online editor
onlinegdb.com/online_c_compiler
Run
Debug
Stop
Share
Save
Beautify
Language
input
Type here to search
13:22
25-12-2020

```
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
4
9
20
17
4
3

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
5
3
4
9
17
20

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
6
enter the item
20

1.insert
2.display
3.pre
4.post
5.in
6.delete
7.exit
enter the choice
```