Report on

"Data Structures"

Submitted in partial fulfillment of the requirements for the award of the degree of Bachelor of Engineering in Computer Science and Engineering in the course of ***Data Structures (19CS3PCDST)***

Submitted by

Ashutosh Upadhyay

(1BM19CS027)

Under the Guidance
of **Dr. Kayarvizhy N.**
Associate Professor

Department of CSE

Department of Computer Science and Engineering

BMS College of Engineering

P.O. Box No.: 1908, Bull Temple Road, Bangalore-560 019

2020-2021

B M S COLLEGE OF ENGINEERING

P.O. Box No: 1908 Bull Temple Road Bangalore-560019

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ASSESSMENT

Report on **Data Structures (19CS3PCDST),** "Advanced Algorithm assignment" has been successfully completed by **Ashutosh Upadhyay** at B.M.S College of Engineering in partial fulfilment of the requirements for the 3rd Semester, degree in Bachelor of Engineering in Computer Science and Engineering under Visvesvaraya Technological University, Belgaum during academic year 2020-2021.

Dr. Kayarvizhy N.

Final Marks Awarded

Associate Professor
Department of Computer science

Obtained	Total

CONTENT

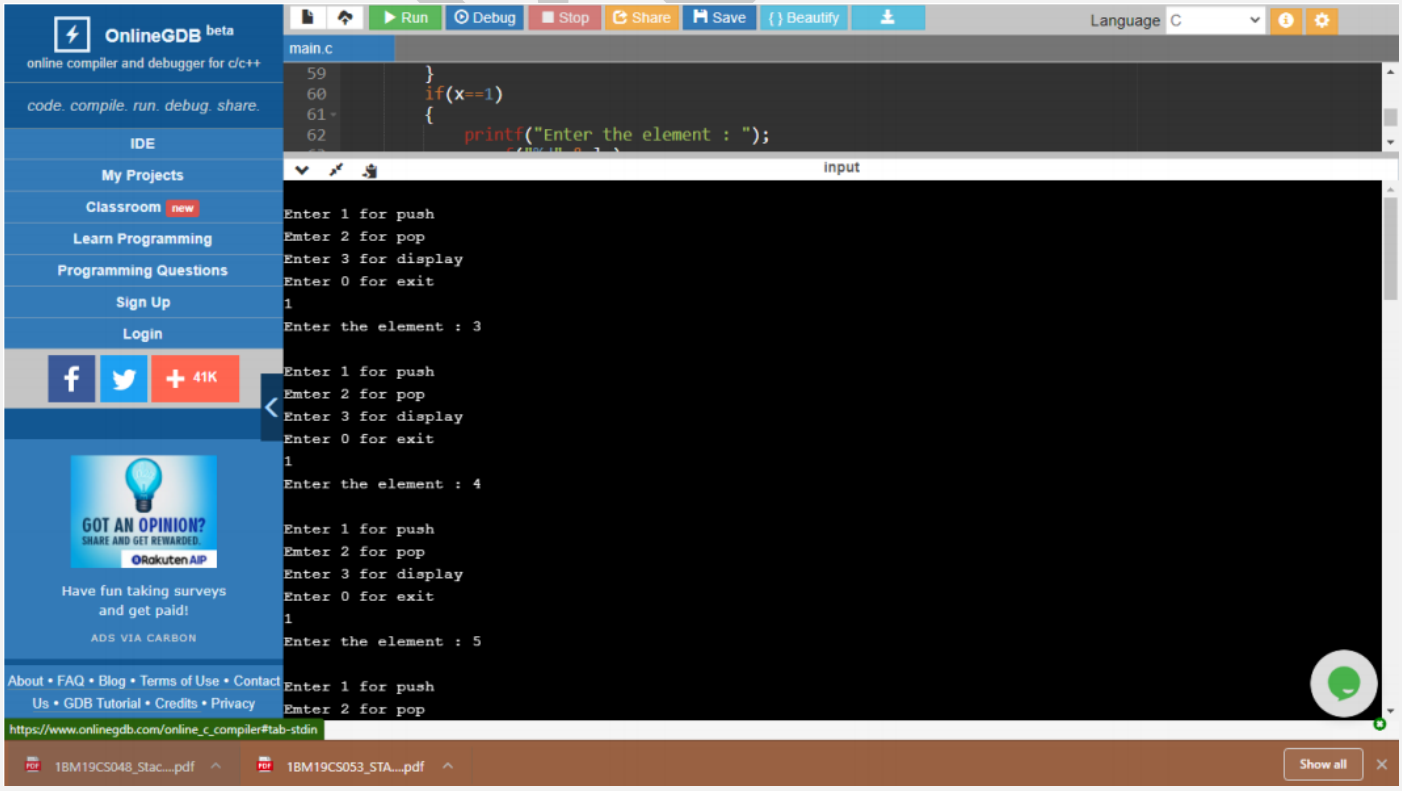
LAB PROGRAM 1: **Write a program to simulate the working of stack using an array with the following: a) Push b) Pop c) Display The program should print appropriate messages for stack overflow, stack underflow**

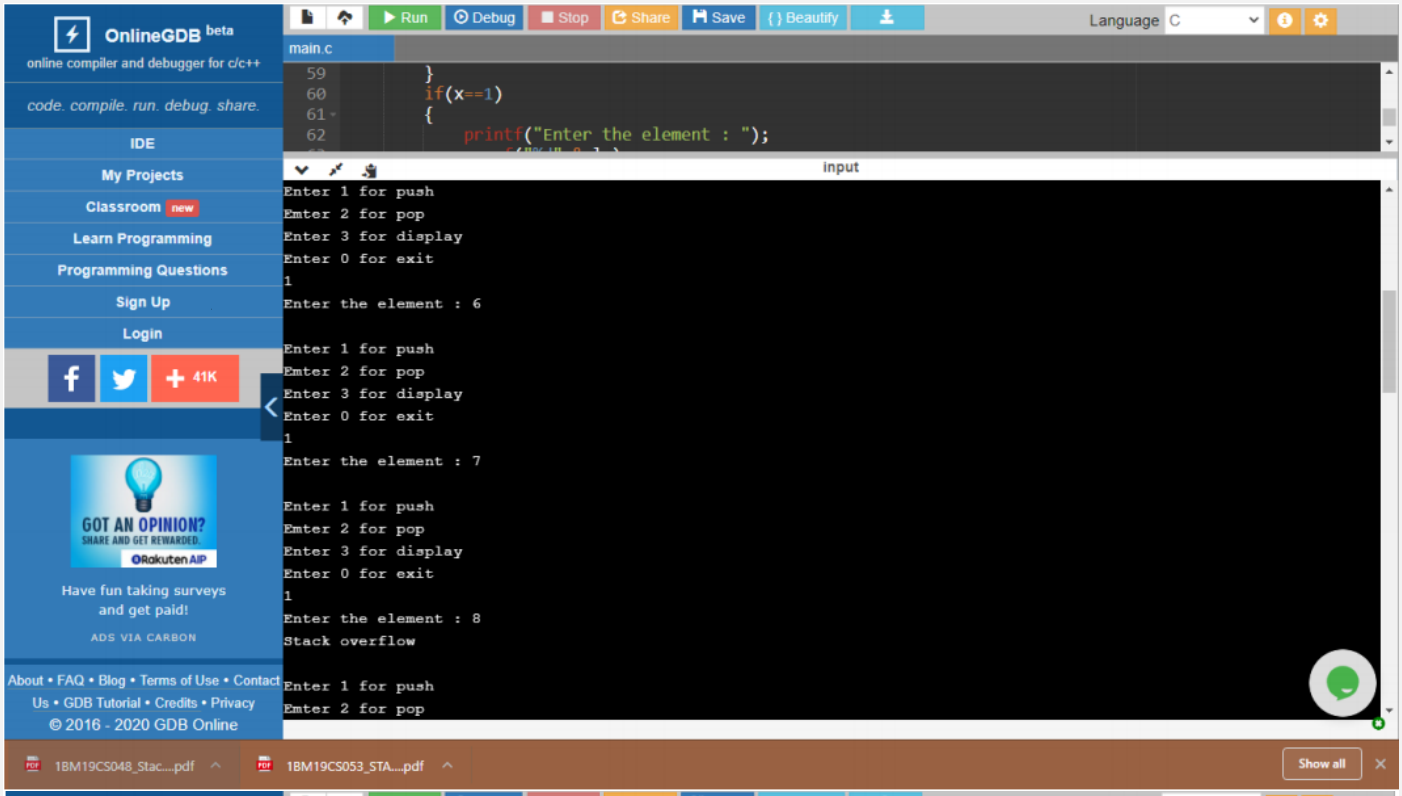
PROGRAM CODE

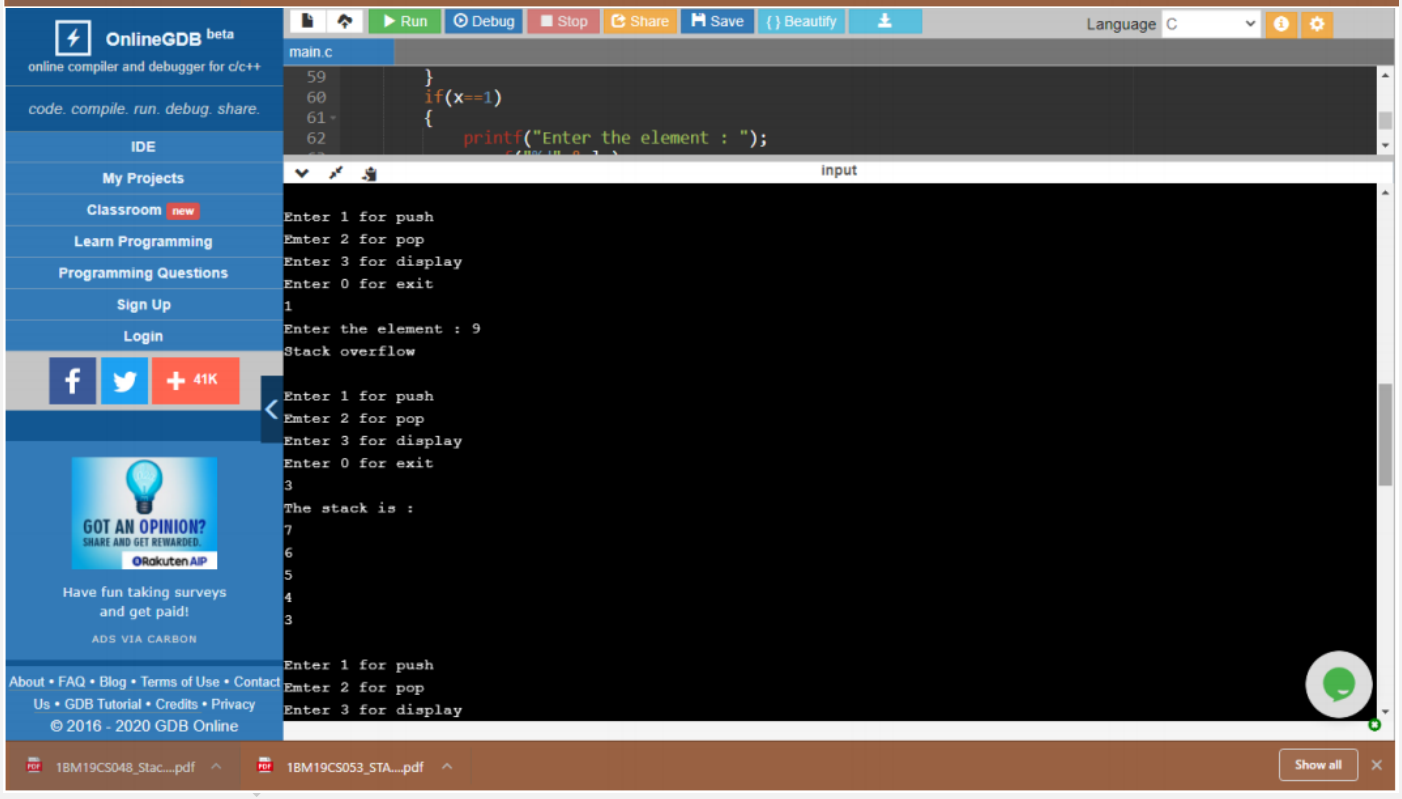
```
#include<stdio.h>
#define size 5
int top=-1;
void push(int stack[],int ele)
{
    if(top==size-1)
    {
        printf("Stack overflow\n");
    }
    else
    {
        top++;
        stack[top]=ele;
    }
}
int pop(int stack[])
{
    if(top==-1)
    {
        printf("Stack underflow\n");
        return -1;
    }
    else
    {
        int x=stack[top];
        top--;
        return x;
    }
}
void display(int stack[])
{
    int i;
    if(top==-1)
    {
        printf("Stack is empty\n");
    }
    else
    {
        for(i=top;i>=0;i--)
        {
            printf("%d\n",stack[i]);
        }
    }
}
int main()
{
    int stack[size];
    int x=1,ele;
    while(x!=0)
    {
        printf("\nEnter 1 for push\n");
```

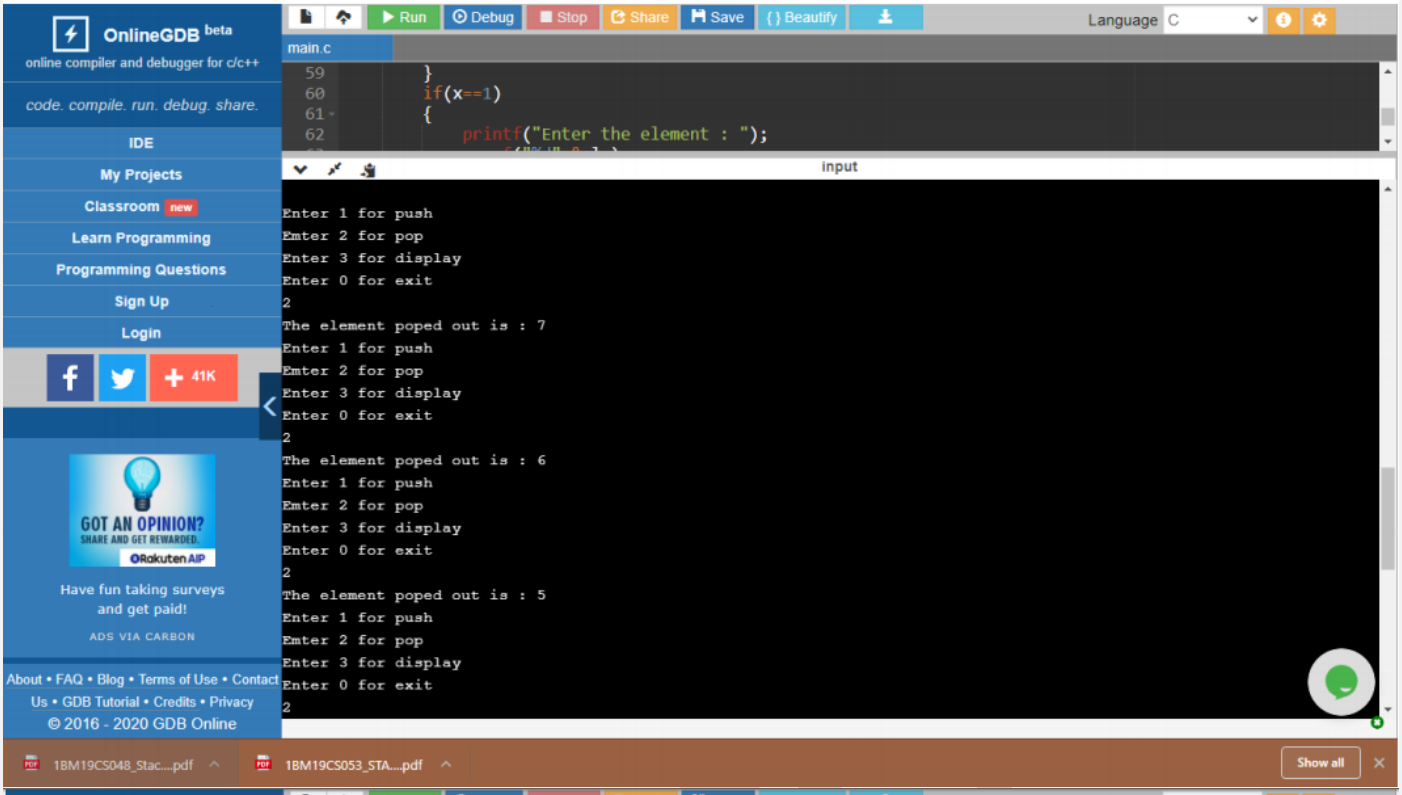
```
        printf("Emter 2 for pop\n");
        printf("Enter 3 for display\n");
        printf("Enter 0 for exit\n");
        scanf("%d",&x);
        if(x==0)
        {
            break;
        }
        if(x==1)
        {
            printf("Enter the element : ");
            scanf("%d",&ele);
            push(stack,ele);
        }
        else if(x==2)
        {
            int popele=pop(stack);
            printf("The element poped out is : %d",popele);
        }
        else if(x==3)
        {
            printf("The stack is : \n");
            display(stack);
        }
    }
    return 0;
}
```

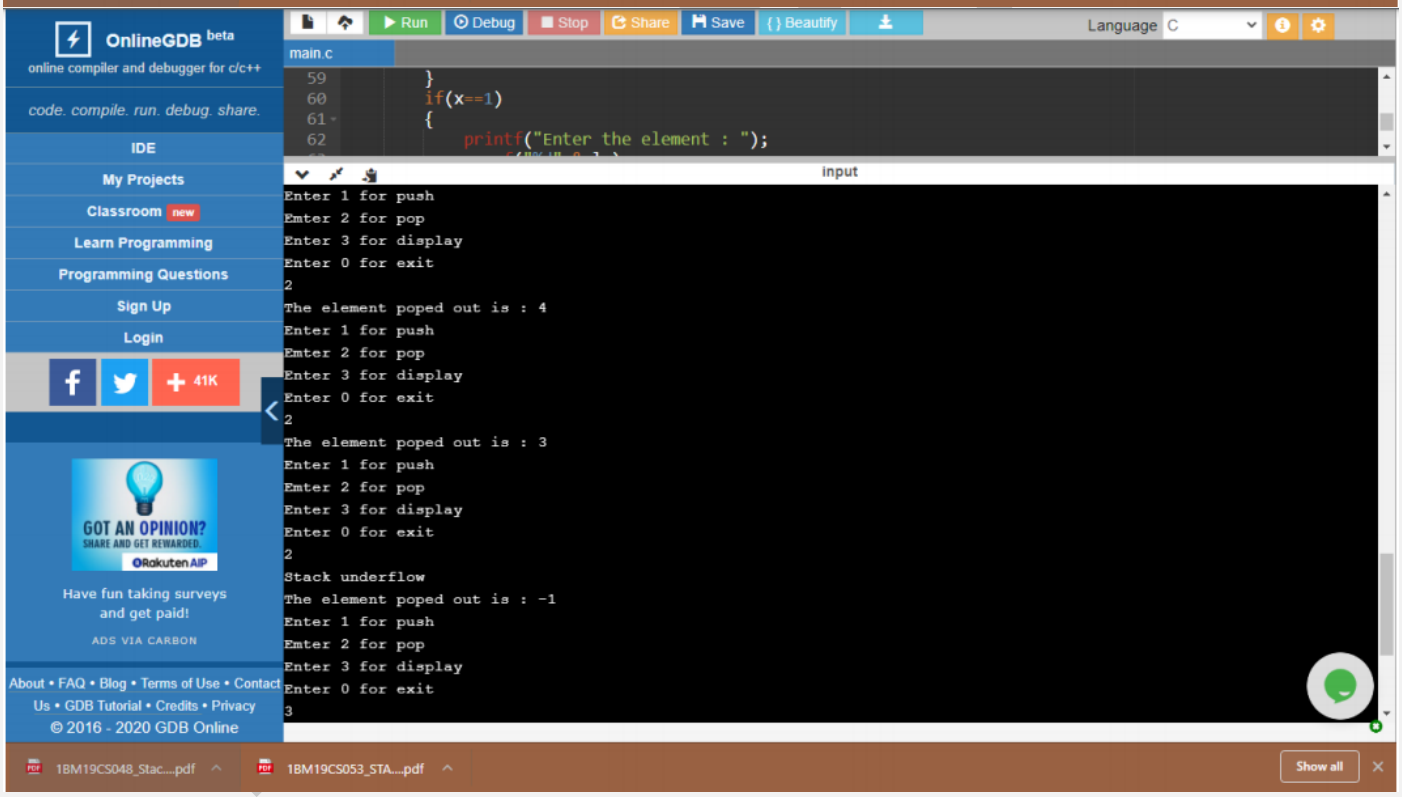
OUTPUT

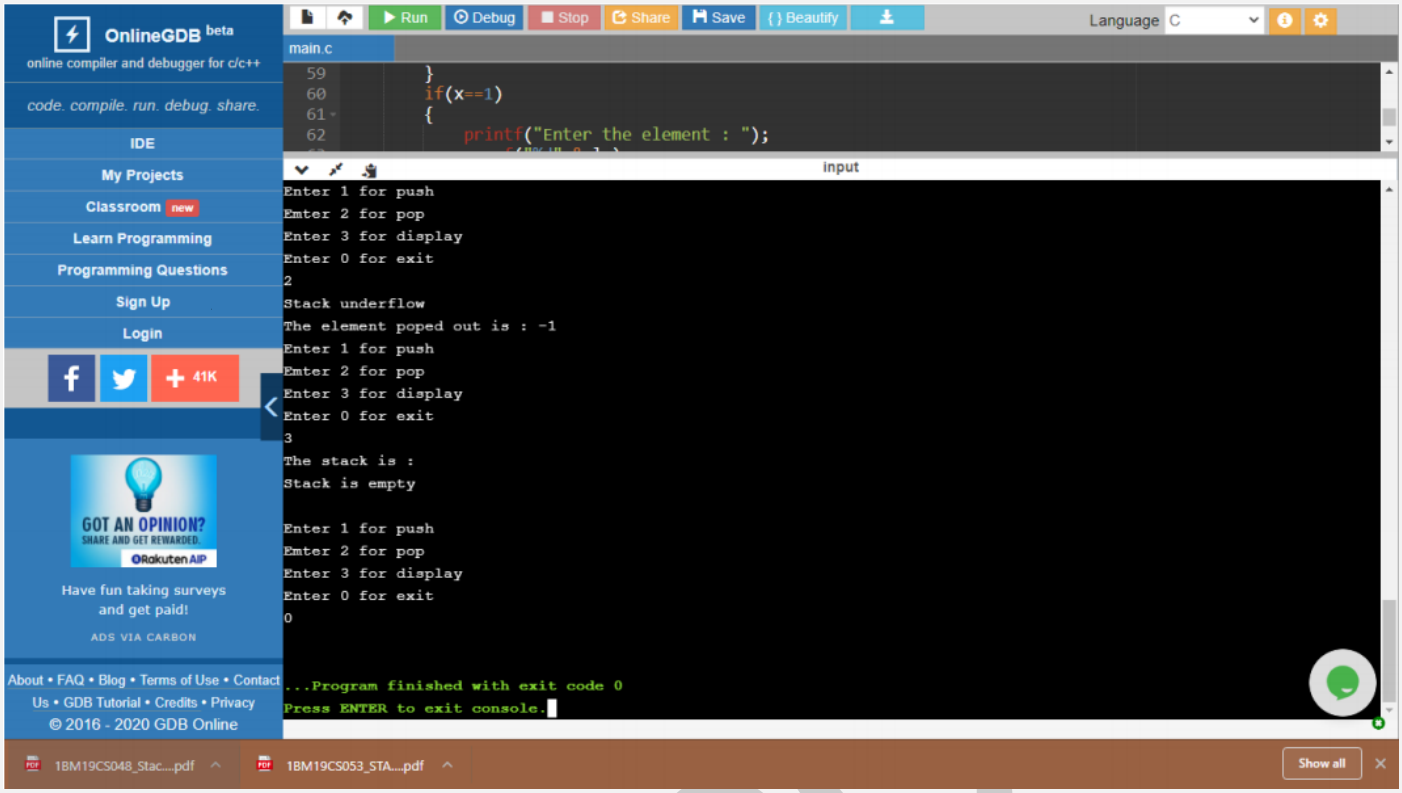



OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
41K
GOT AN OPINION?
SHARE AND GET REWARDED.
Rakuten AIP
Have fun taking surveys and get paid!
ADS VIA CARBON
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2020 GDB Online
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
59 }
60 if(x==1)
61 {
62 printf("Enter the element : ");
input
Enter 1 for push
Emter 2 for pop
Enter 3 for display
Enter 0 for exit
1
Enter the element : 6
Enter 1 for push
Emter 2 for pop
Enter 3 for display
Enter 0 for exit
1
Enter the element : 7
Enter 1 for push
Emter 2 for pop
Enter 3 for display
Enter 0 for exit
1
Enter the element : 8
Stack overflow
Enter 1 for push
Emter 2 for pop
1BM19CS048_Stac....pdf
1BM19CS053_STA....pdf
Show all


OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
41K
GOT AN OPINION?
SHARE AND GET REWARDED.
Rakuten AIP
Have fun taking surveys and get paid!
ADS VIA CARBON
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2020 GDB Online
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
59 }
60 if(x==1)
61 {
62 printf("Enter the element : ");
input
Enter 1 for push
Emter 2 for pop
Enter 3 for display
Enter 0 for exit
1
Enter the element : 9
Stack overflow
Enter 1 for push
Emter 2 for pop
Enter 3 for display
Enter 0 for exit
3
The stack is :
7
6
5
4
3
Enter 1 for push
Emter 2 for pop
Enter 3 for display
1BM19CS048_Stac....pdf
1BM19CS053_STA....pdf
Show all


OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
41K
GOT AN OPINION?
SHARE AND GET REWARDED.
Rakuten AIP
Have fun taking surveys and get paid!
ADS VIA CARBON
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2020 GDB Online
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
59 }
60 if(x==1)
61 {
62 printf("Enter the element : ");
input
Enter 1 for push
Emter 2 for pop
Enter 3 for display
Enter 0 for exit
2
The element poped out is : 7
Enter 1 for push
Emter 2 for pop
Enter 3 for display
Enter 0 for exit
2
The element poped out is : 6
Enter 1 for push
Emter 2 for pop
Enter 3 for display
Enter 0 for exit
2
The element poped out is : 5
Enter 1 for push
Emter 2 for pop
Enter 3 for display
Enter 0 for exit
2
1BM19CS048_Stac....pdf
1BM19CS053_STA....pdf
Show all


OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
41K
GOT AN OPINION?
SHARE AND GET REWARDED.
Rakuten AIP
Have fun taking surveys and get paid!
ADS VIA CARBON
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2020 GDB Online
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
59 }
60 if(x==1)
61 {
62 printf("Enter the element : ");
input
Enter 1 for push
Emter 2 for pop
Enter 3 for display
Enter 0 for exit
2
The element poped out is : 4
Enter 1 for push
Emter 2 for pop
Enter 3 for display
Enter 0 for exit
2
The element poped out is : 3
Enter 1 for push
Emter 2 for pop
Enter 3 for display
Enter 0 for exit
2
Stack underflow
The element poped out is : -1
Enter 1 for push
Emter 2 for pop
Enter 3 for display
Enter 0 for exit
3
1BM19CS048_Stac....pdf
1BM19CS053_STA....pdf
Show all


OnlineGDB beta
online compiler and debugger for c/c++
code. compile. run. debug. share.
IDE
My Projects
Classroom new
Learn Programming
Programming Questions
Sign Up
Login
41K
GOT AN OPINION?
SHARE AND GET REWARDED.
Rakuten AIP
Have fun taking surveys and get paid!
ADS VIA CARBON
About • FAQ • Blog • Terms of Use • Contact Us • GDB Tutorial • Credits • Privacy
© 2016 - 2020 GDB Online
Run
Debug
Stop
Share
Save
{ } Beautify
Language C
main.c
59 }
60 if(x==1)
61 {
62 printf("Enter the element : ");
input
Enter 1 for push
Emter 2 for pop
Enter 3 for display
Enter 0 for exit
2
Stack underflow
The element poped out is : -1
Enter 1 for push
Emter 2 for pop
Enter 3 for display
Enter 0 for exit
3
The stack is :
Stack is empty
Enter 1 for push
Emter 2 for pop
Enter 3 for display
Enter 0 for exit
0
...Program finished with exit code 0
Press ENTER to exit console.
1BM19CS048_Stac....pdf
1BM19CS053_STA....pdf
Show all

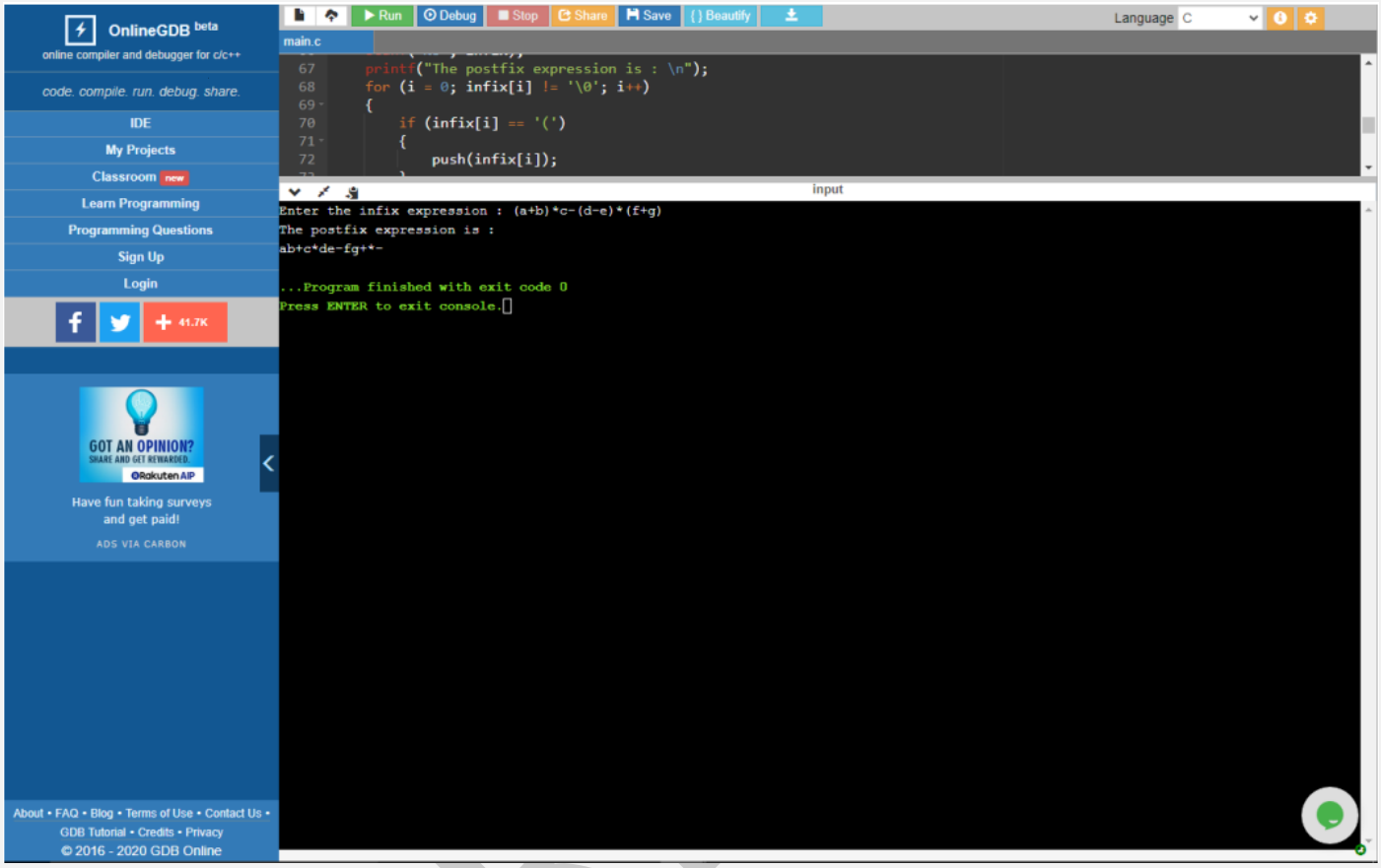
LAB PROGRAM 2: ***WAP to convert a given valid parenthesized infix arithmetic expression to postfix expression. The expression consists of single character operands and the binary operators + (plus), - (minus), * (multiply) and / (divide)***

PROGRAM CODE

```
#include<stdio.h>
#define max 100
char stack[max];
int top = -1;

void push(char ch)
{
      if (top == max - 1)
            printf("Stack is full\n");
      else
      {
            top++;
            stack[top] = ch;
      }
}
char pop()
{
      char ch=-1;
      if (top == -1)
            printf("stack is empty\n");

      else
      {
            ch = stack[top];
            top--;
      }
      return ch;
}
int stackempty()
{
      if (top == -1)
            return 1;
      return 0;
}
char stacktop()
{
      if (top == -1)
            return -1;
      else
      {
            char ch = stack[top];
            return ch;
      }
}
int priority(char ch)
{
      if (ch == '^')
            return 5;
      else if (ch == '/')
            return 4;
      else if (ch == '*')
```

```
            return 3;
      else if (ch == '-')
            return 2;
      else if (ch == '+')
            return 1;
      else
            return 0;

}
int main()
{
      int i,k=0;
      char infix[100], postfix[100];
      printf("Enter the infix expression : ");
      scanf("%s", infix);
      printf("The postfix expression is : \n");
      for (i = 0; infix[i] != '\0'; i++)
      {
            if (infix[i] == '(')
            {
                  push(infix[i]);
            }
            else if (infix[i] == ')')
            {
                  while (stacktop() != '(')
                  {
                        postfix[k] = pop();
                        k++;
                  }
                  pop();
            }
            else if (infix[i] == '^' || infix[i] == '/' || infix[i] == '*' ||
infix[i] == '-' || infix[i] == '+')
            {
                  while(stackempty() == 0 && priority(stacktop()) >=
priority(infix[i]))
                  {
                        postfix[k] = pop();
                        k++;
                  }
                  push(infix[i]);
            }
            else
            {
                  postfix[k] = infix[i];
                  k++;
            }

      }
      while(!stackempty())
    {
       postfix[k] = pop();
                        k++;
    }
      for(i=0;i<k;i++)
      {
          printf("%c",postfix[i]);
      }
}
```

OUTPUT



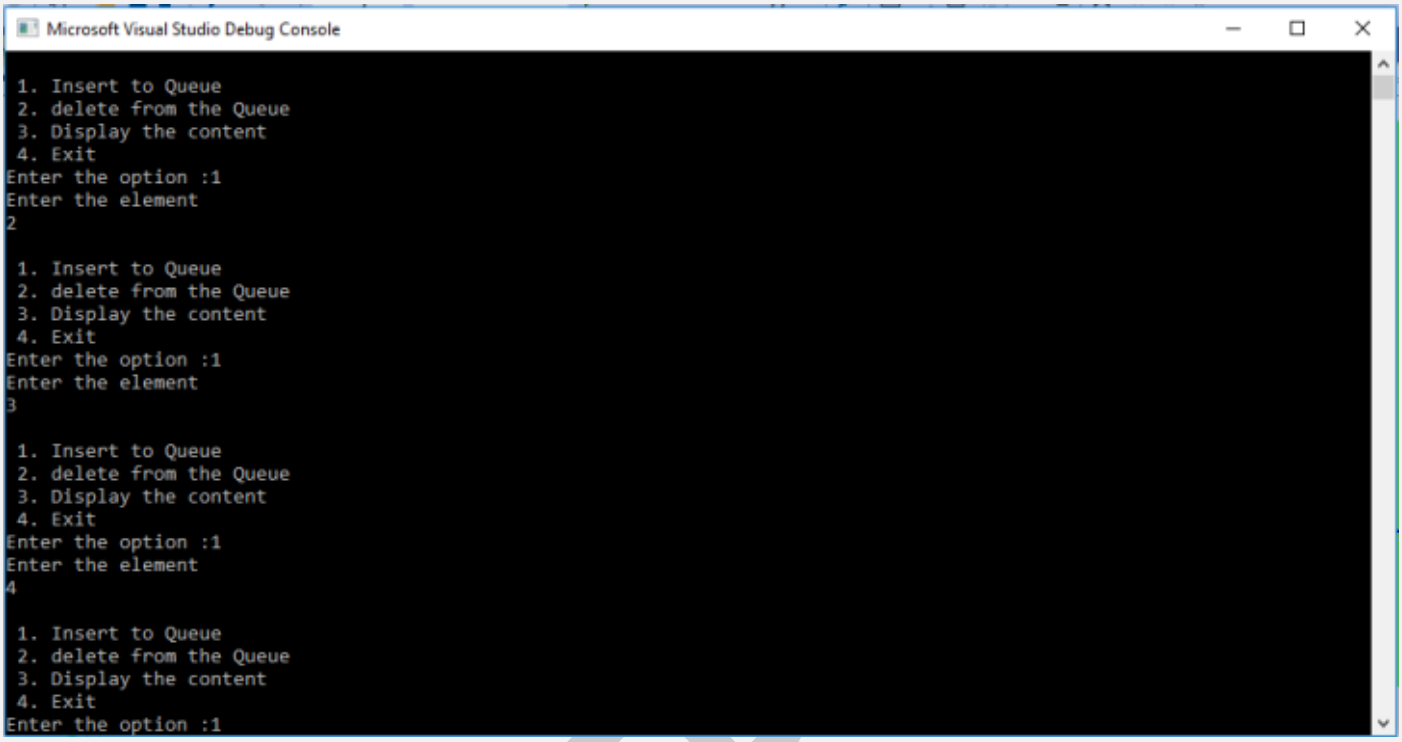
LAB PROGRAM 3: ***WAP to simulate the working of a queue of integers using an array. Provide the following operations a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions***

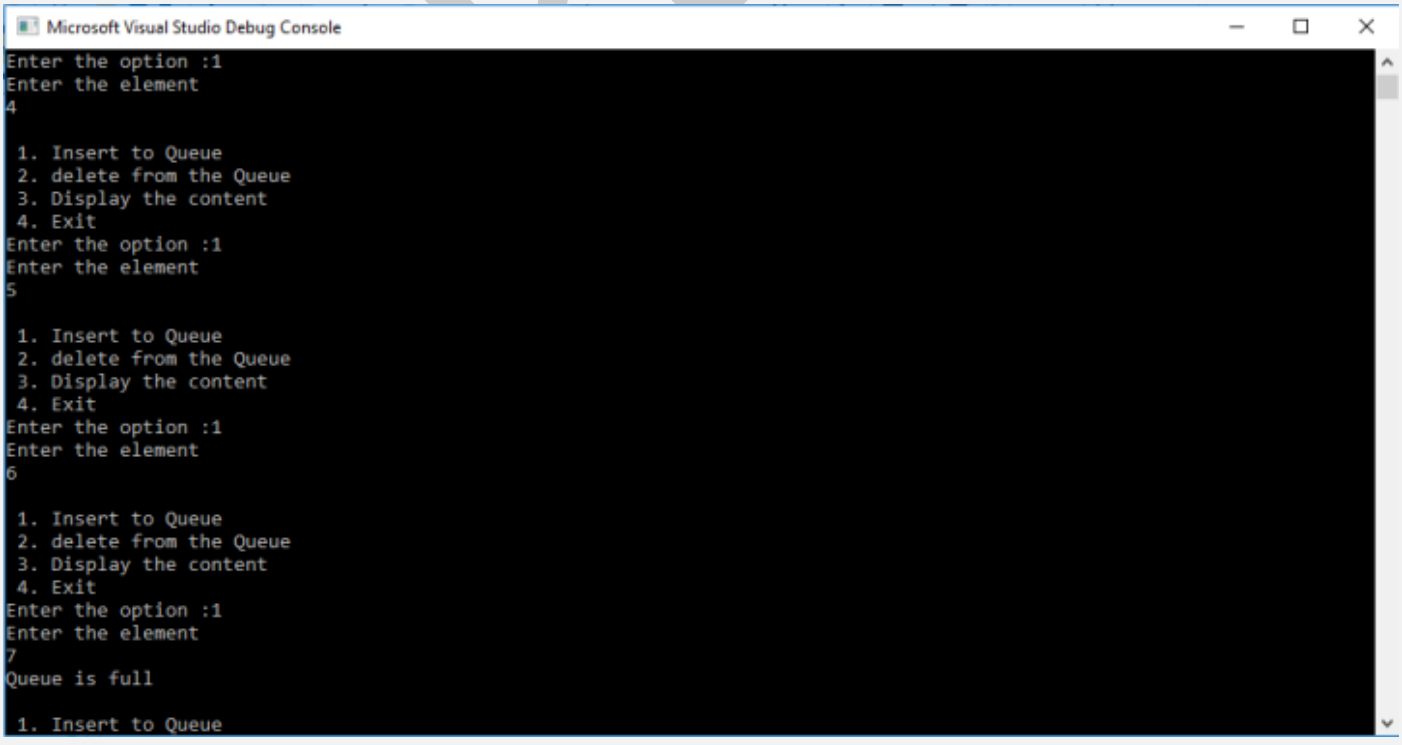
PROGRAM CODE

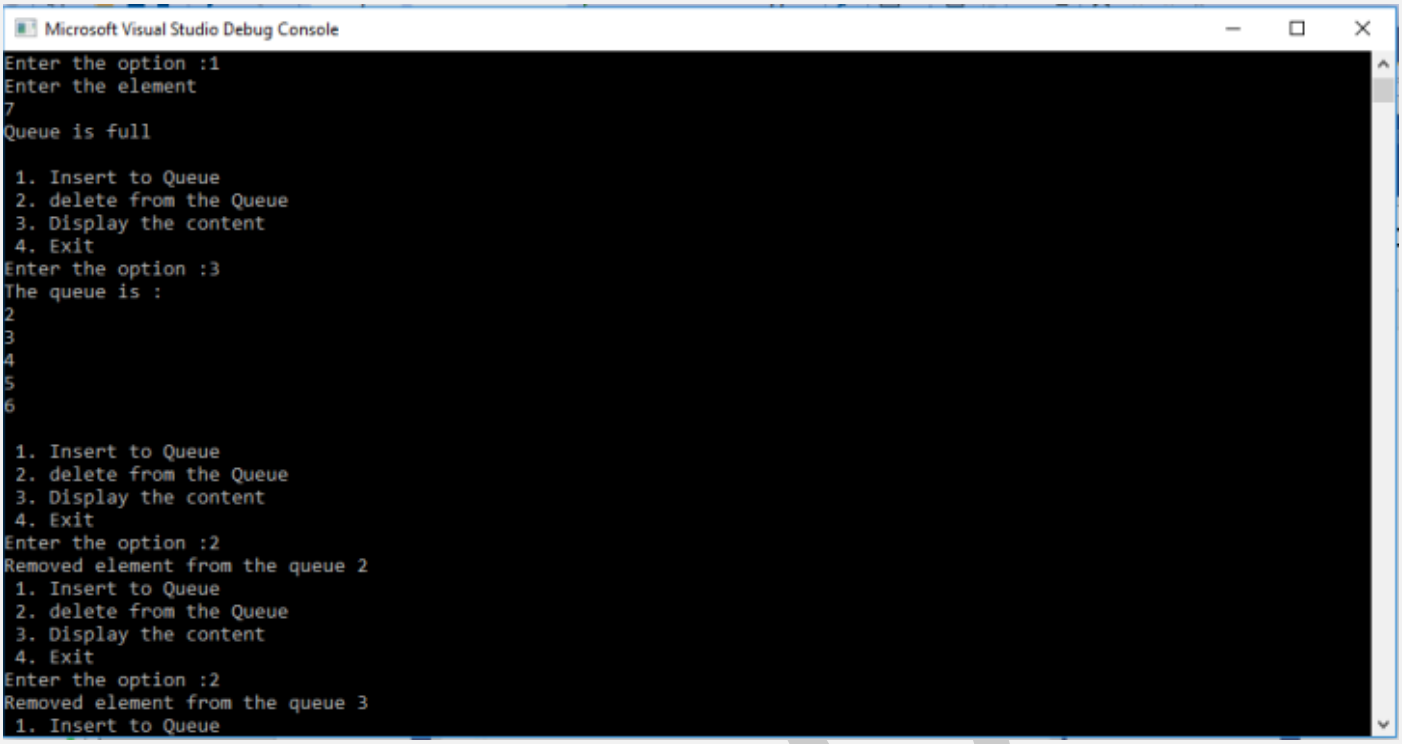
```
#include<stdio.h>


void enqueue(int x, int queue[], int* front, int* rear, int size)
{
	if (*rear == size - 1)
		printf("Queue is full\n");
	else if (*front == -1 && *rear == -1)
	{
		*front++;
		*rear++;
		queue[*rear] = x;
	}
	else
	{
		*rear++;
		queue[*rear] = x;
	}
}

int dequeue(int queue[], int* front, int* rear, int size)
{
	int x;
	if (*front == -1)
		return -1;
	else
	{
		x = queue[*front];
		*front++;
		if (*front > * rear)
		{
			*front = -1;
			*rear = -1;
		}
		return x;
	}
}

void display(int queue[], int* front, int* rear, int size)
{
	int i;
	if (*front == -1)
		printf("Queue is empty\n");
	else
	{
		printf("The queue is :\n");
		for (i = *front; i <= *rear; i++)
		{
			printf("%d\n", queue[i]);
		}
```

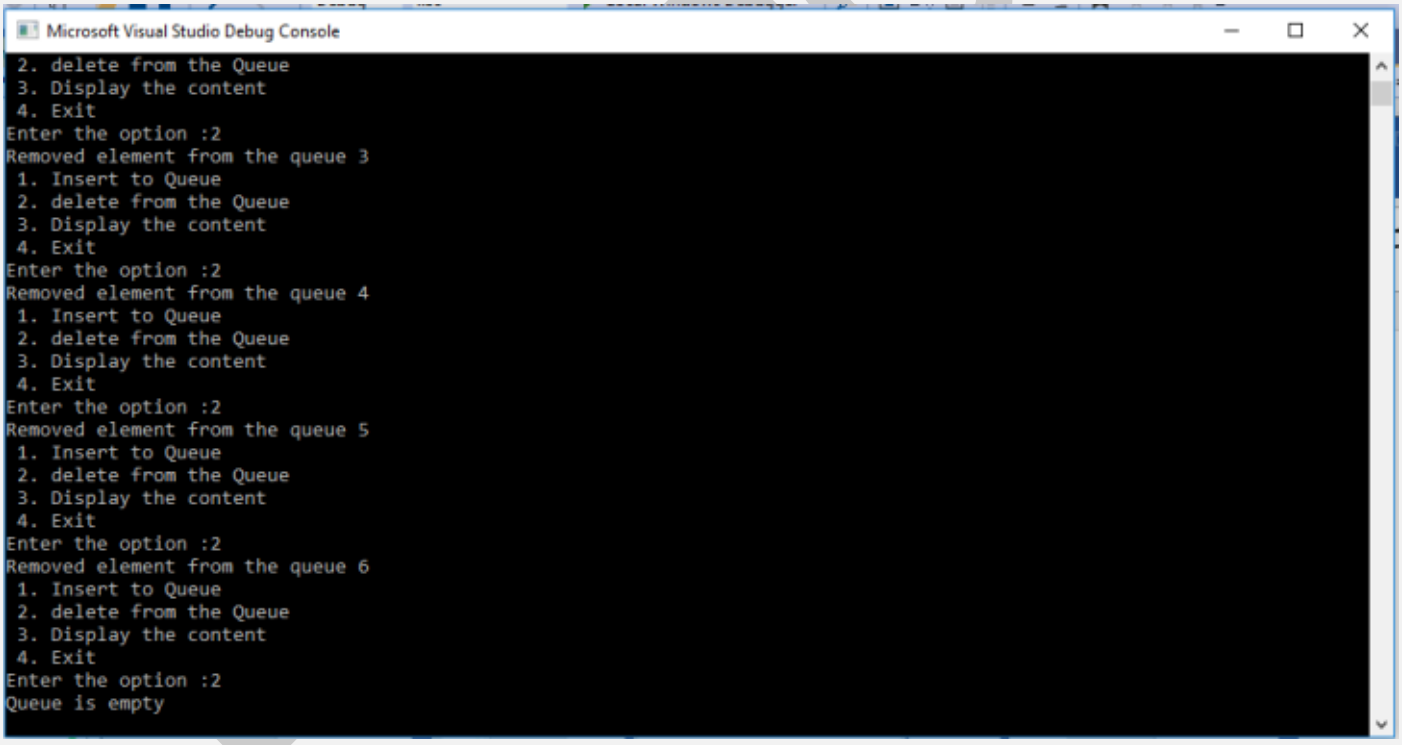
```
	}
}

int main()
{
	int size = 5;
	int queue[size];
	int front = -1;
	int rear = -1;
	int i;
	int x;
	do {
		printf("\n 1. Insert to Queue ");
		printf("\n 2. delete from the Queue )");
		printf("\n 3. Display the content ");
		printf("\n 4. Exit\n");
		printf("Enter the option :");
		scanf_s("%d", &i);
		switch (i)
		{
		case 1:  printf("Enter the element\n");
			scanf_s("%d", &x);
			enqueue(x, queue, &front, &rear, size);
			break;
		case 2: x = dequeue(queue, &front, &rear, size);
			if (x == -1)
				printf("Queue is empty\n");
			else
				printf("Removed element from the queue %d", x);
			break;
		case 3: display(queue, &front, &rear, size);
			break;
		case 4: break;
		}
	} while (i != 4);
	return 0;
}
```

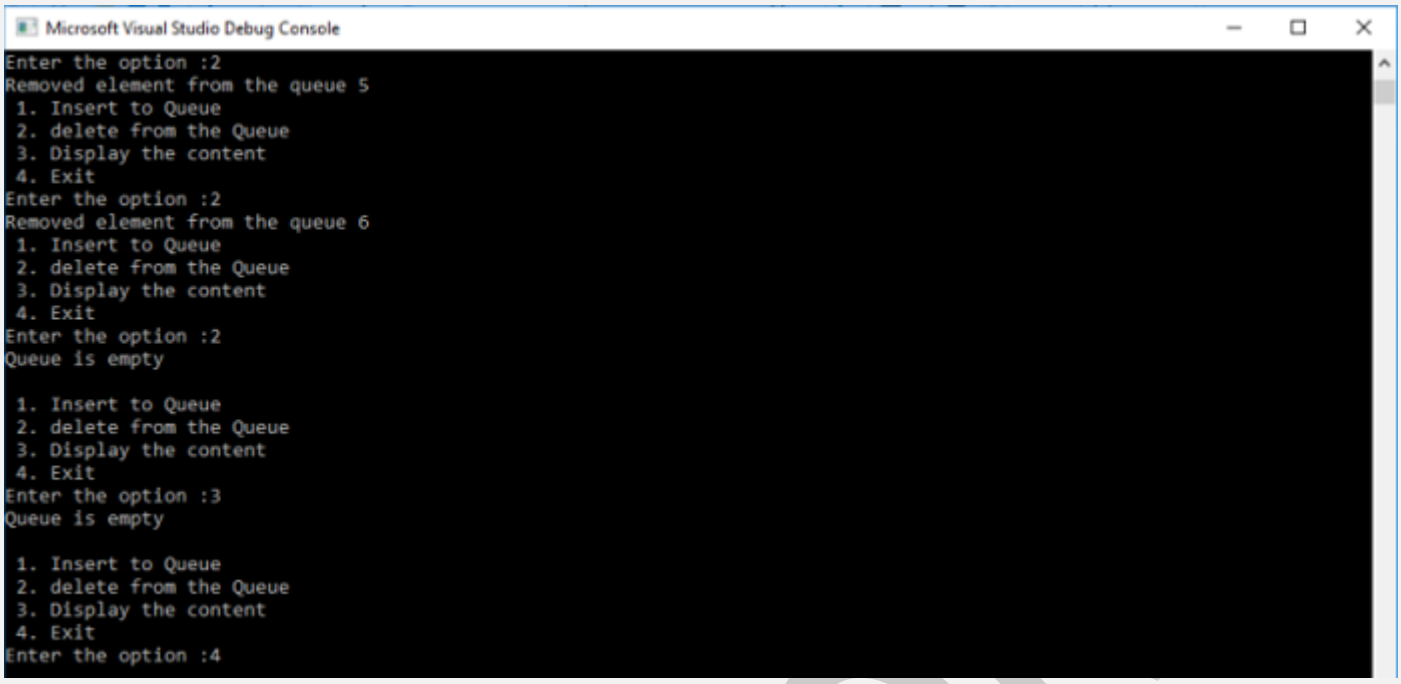
OUTPUT


Microsoft Visual Studio Debug Console
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :1
Enter the element
2
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :1
Enter the element
3
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :1
Enter the element
4
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :1


Microsoft Visual Studio Debug Console
Enter the option :1
Enter the element
4
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :1
Enter the element
5
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :1
Enter the element
6
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :1
Enter the element
7
Queue is full
1. Insert to Queue


Microsoft Visual Studio Debug Console
Enter the option :1
Enter the element
7
Queue is full
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :3
The queue is :
2
3
4
5
6
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 2
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 3
1. Insert to Queue


Microsoft Visual Studio Debug Console
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 3
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 4
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 5
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 6
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :2
Queue is empty


Microsoft Visual Studio Debug Console
Enter the option :2
Removed element from the queue 5
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :2
Removed element from the queue 6
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :2
Queue is empty
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :3
Queue is empty
1. Insert to Queue
2. delete from the Queue
3. Display the content
4. Exit
Enter the option :4

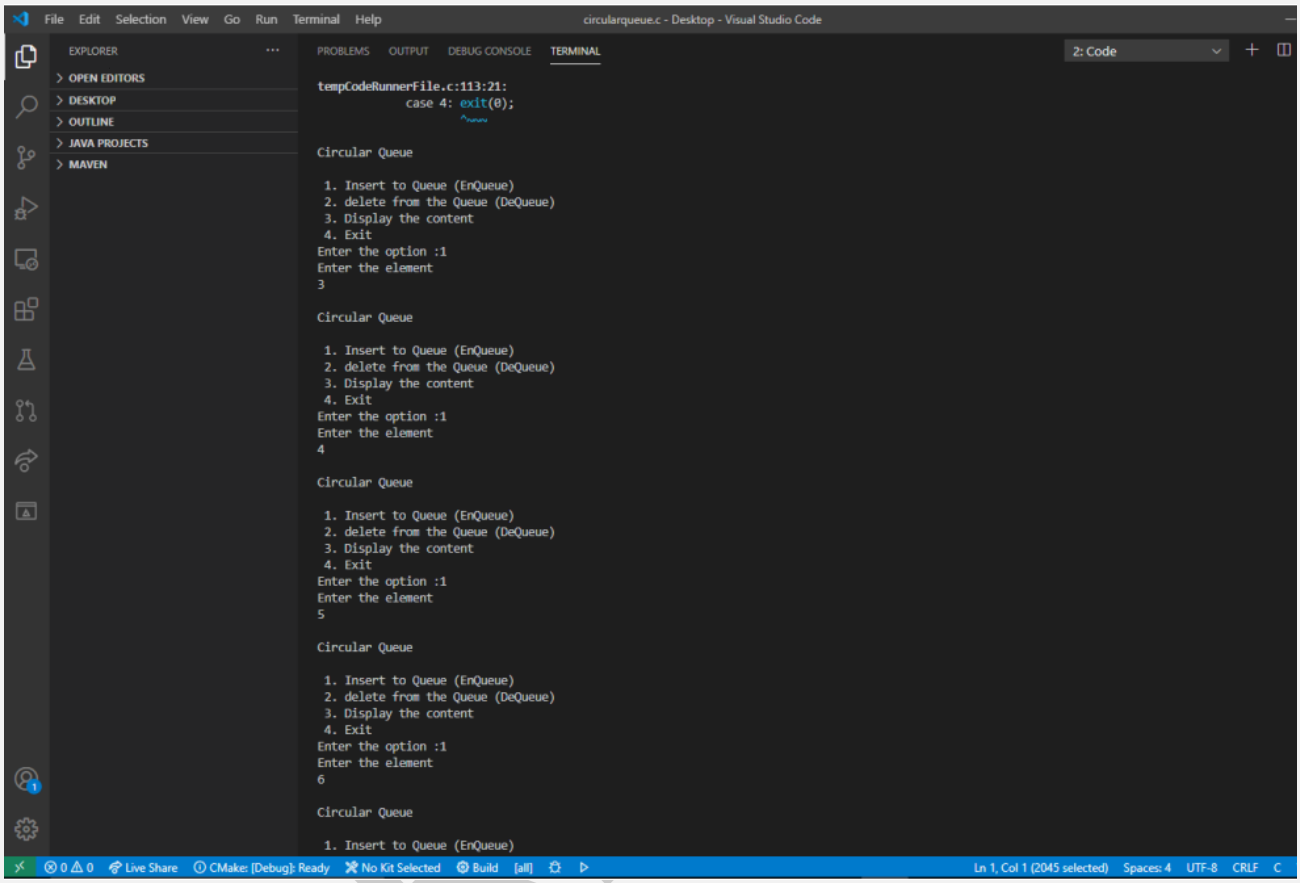
LAB PROGRAM 4 : **WAP to simulate the working of a circular queue of integers using an array. Provide the following operations. a) Insert b) Delete c) Display The program should print appropriate messages for queue empty and queue overflow conditions**

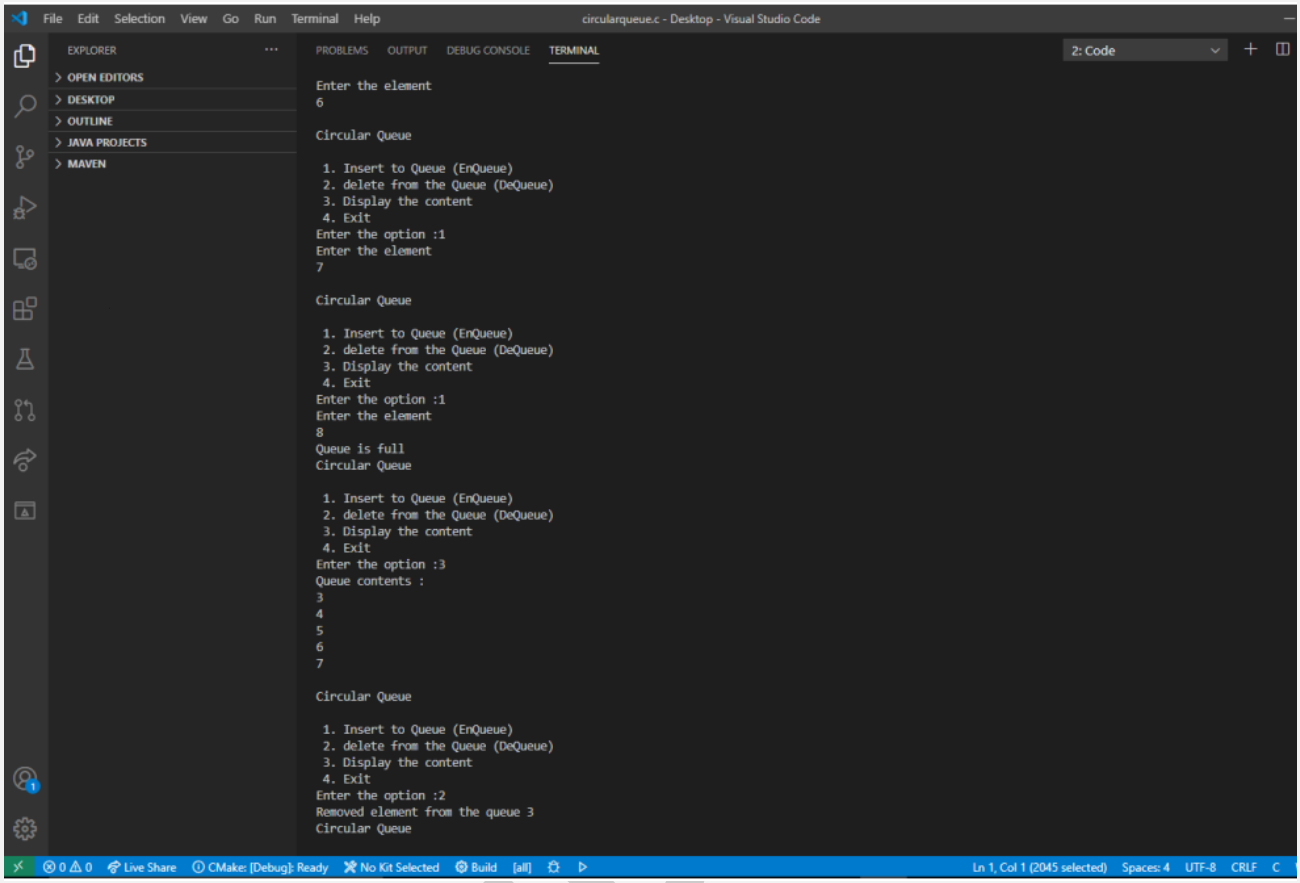
PROGRAM CODE

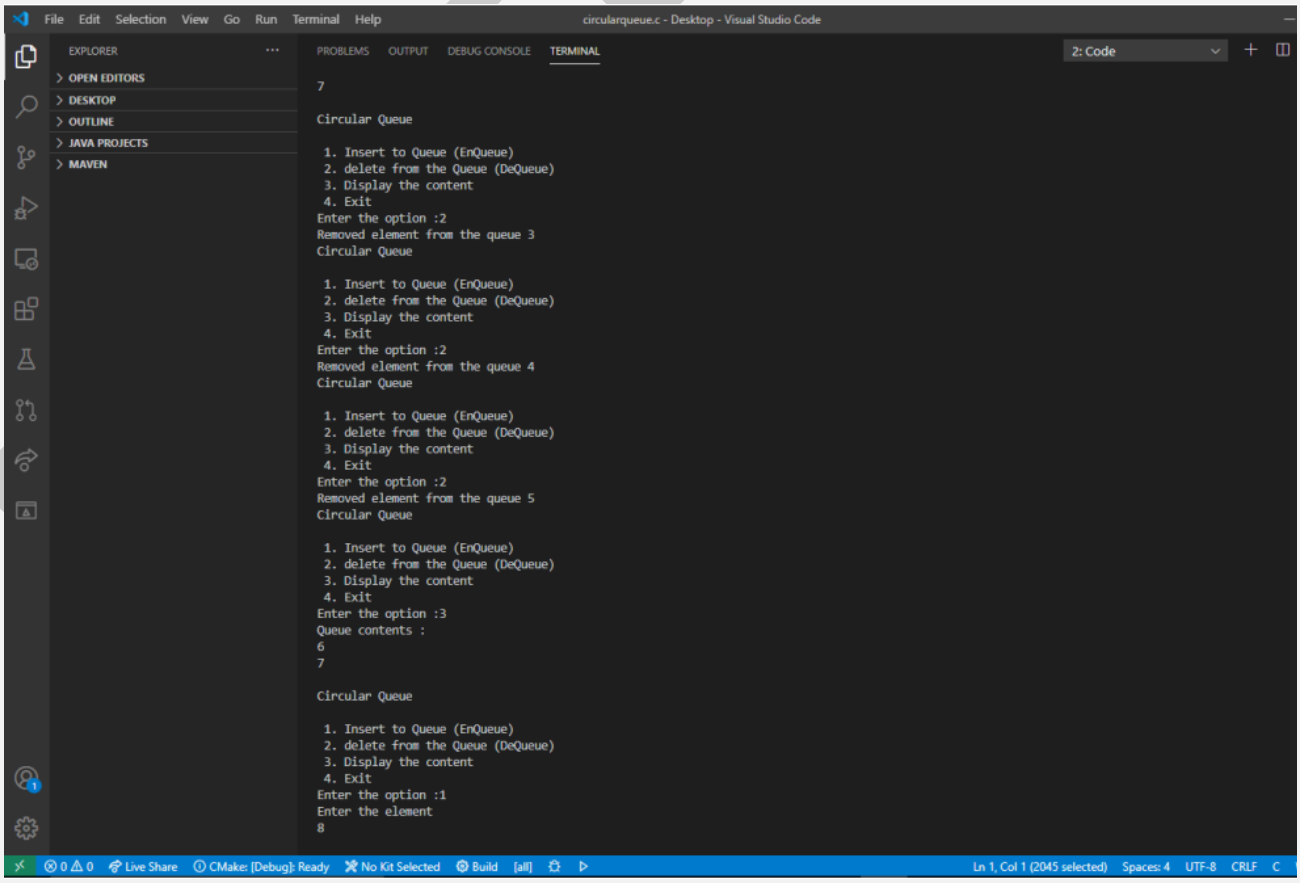
```
#include<stdio.h>


#define size 5
int front = -1;
int rear = -1;

int queue[size];

void enque(int n)
{
	if (front == 0 && rear == size - 1)
	{
		printf("Queue is full");
	}
	else if (front == rear + 1)
	{
		printf("Queue is full");
	}
	else if (front == -1 && rear == -1)
	{
		front++;
		rear++;
		queue[rear] = n;
	}
	else
	{
		rear = (rear + 1) % size;
		queue[rear] = n;
	}

}

int dequeue()
{
	if (front == -1 && rear == -1)
	{
		return -1;
	}
	else
	{
		int ele;
		ele = queue[front];
		if (front == rear)
		{
			front = -1;
			rear = -1;
		}
		else
```

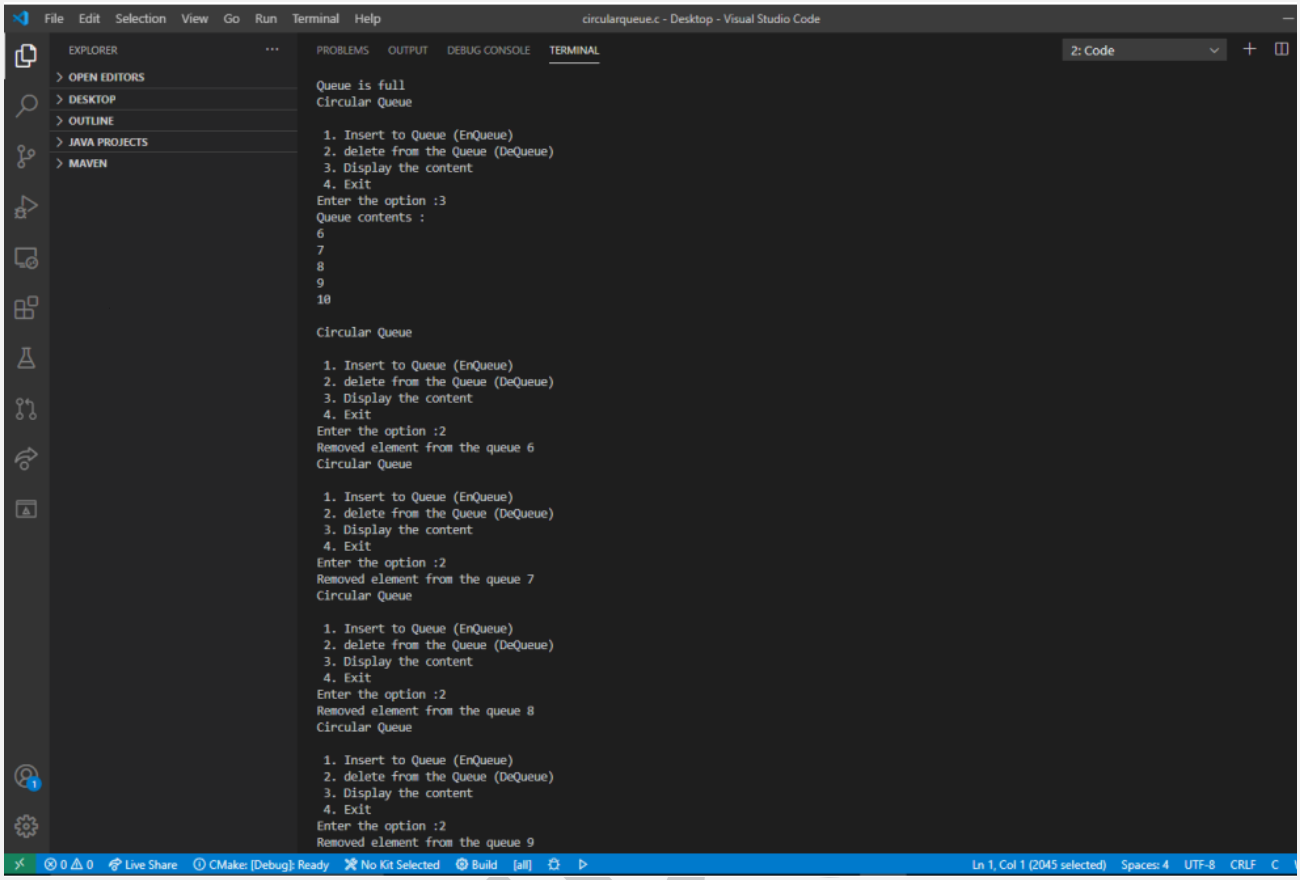
```
                {
                        front = (front + 1) % size;
                }
                return ele;
        }
}

void display()
{
        if (front == -1 && rear == -1)
        {
                printf("Queue is empty");
        }
        else
        {
                printf("Queue contents : \n");
                if (front <= rear)
                {
                        for (int i = front; i <= rear; i++)
                        {
                                printf("%d\n", queue[i]);
                        }
                }
                else
                {
                        for (int i = front; i <= size-1; i++)
                        {
                                printf("%d\n", queue[i]);
                        }
                        for (int i = 0; i <= rear; i++)
                        {
                                printf("%d\n", queue[i]);
                        }
                }
        }
}

int main()
{
        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_s("%d",&option);
         switch(option)
         {
             case 1:  printf("Enter the element\n");
                      scanf("%d",&item);
                      enque(item);
                      break;
             case 2: item=dequeue();
                     if(item==-1)
                         printf("Queue is empty");
                     else
                     printf("Removed element from the queue %d",item);
                     break;
             case 3: display();
                     break;
             case 4: exit(0);
         }
    } while (option!=4);
        return 0;
}
```

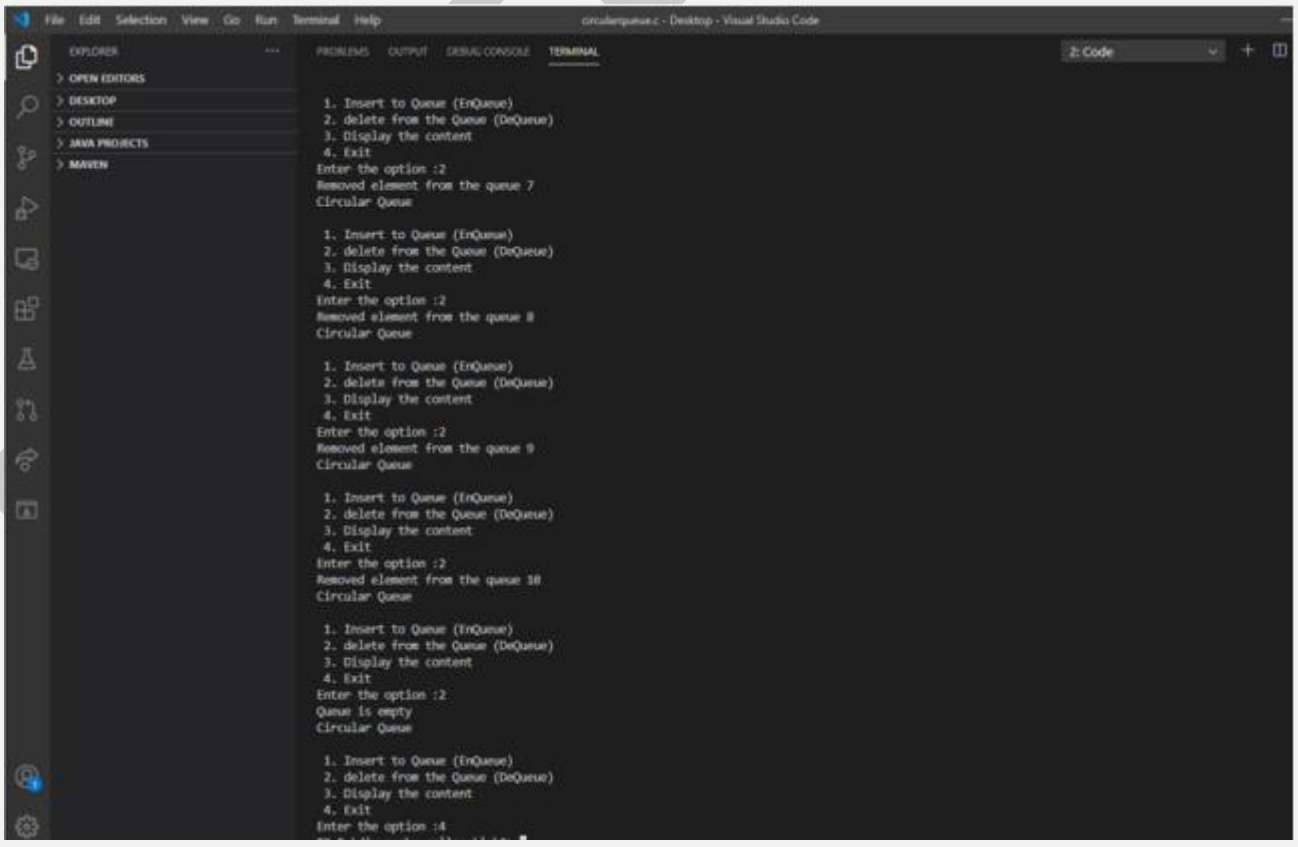
OUTPUT




File Edit Selection View Go Run Terminal Help
circularqueue.c - Desktop - Visual Studio Code
EXPLORER
OPEN EDITORS
DESKTOP
OUTLINE
JAVA PROJECTS
MAVEN
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: Code
Enter the element
6
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
8
Queue is full
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 :
3
4
5
6
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 3
Circular Queue
Live Share
CMake: [Debug]: Ready
No Kit Selected
Build
[all]
Ln 1, Col 1 (2045 selected)
Spaces: 4
UTF-8
CRLF
C


File Edit Selection View Go Run Terminal Help
circularqueue.c - Desktop - Visual Studio Code
EXPLORER
OPEN EDITORS
DESKTOP
OUTLINE
JAVA PROJECTS
MAVEN
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: Code
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 3
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
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 5
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 :
6
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
8
Live Share
CMake: [Debug]: Ready
No Kit Selected
Build
[all]
Ln 1, Col 1 (2045 selected)
Spaces: 4
UTF-8
CRLF
C


File Edit Selection View Go Run Terminal Help
circularqueue.c - Desktop - Visual Studio Code
EXPLORER
OPEN EDITORS
DESKTOP
OUTLINE
JAVA PROJECTS
MAVEN
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
2: Code
Queue is full
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 :
6
7
8
9
10
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 6
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 8
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
Live Share
CMake: [Debug]: Ready
No Kit Selected
Build
[all]
Ln 1, Col 1 (2045 selected)
Spaces: 4
UTF-8
CRLF
C


EXPLORER
OPEN EDITORS
DESKTOP
OUTLINE
JAVA PROJECTS
MAVEN
2: Code
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 8
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
Removed element from the queue 10
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 :4

LAB PROGRAM 5: WAP to Implement Singly Linked List with following operations a) a) Create a linked list. b) Insertion of a node at first position, at any position and at end of list. c) Display the contents of the linked list

PROGRAM CODE

```
#include<stdio.h>

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

struct node *head=NULL;

int length=0;

void insertend(int ele)
{
    struct node *newnode,*temp;
    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=ele;
    newnode->next=NULL;
    if(head==NULL)
    {
        head=newnode;
        length=1;
    }
    else
    {
        temp=(struct node*)malloc(sizeof(struct node));
        temp=head;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=newnode;
        length++;
    }

}

void insertfront(int ele)
{
    struct node *temp;
    temp=(struct node*)malloc(sizeof(struct node));
    temp->data=ele;
    temp->next=head;
    head=temp;
    length++;
}

void insertrandom(int ele,int pos)
{
    if(pos==1)
        insertfront(ele);
    else if(pos>=length+1)
```

```
        insertend(ele);
    else
    {
        struct node *inst;
        inst=(struct node*)malloc(sizeof(struct node));
        struct node *temp;
        temp=(struct node*)malloc(sizeof(struct node));
        temp=head;
        for(int i=1;i<pos-1;i++)
        {
               temp=temp->next;
        }
        inst->data=ele;
        inst->next=temp->next;
        temp->next=inst;
        length++;

    }

}


void display()
{
    struct node *temp;
    temp=(struct node*)malloc(sizeof(struct node));
    temp=head;
    if(temp==NULL)
    {
        printf("\n List is empty \n");
    }
    else
    {
        printf("\nThe contents of the list are :\n");
        while(temp!=NULL)
        {
            printf("%d\n",temp->data);
            temp=temp->next;
        }
    }

}

int main()
{
    int choice,ele,pos;
    char ch;
    do
    {
    printf("\n1. Inset at end \n2.Insert at front \n3.Insert at random position
\n4. Display  \n5.exit");
    printf("\nEnter your choice : ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1: printf("Enter the element to be inserted\n");
                scanf("%d",&ele);
                insertend(ele);
                break;
        case 2: printf("Enter the element to be inserted\n");
                scanf("%d",&ele);
```

```
                insertfront(ele);
                break;
        case 3: printf("Enter the element to be inserted\n");
                scanf("%d",&ele);
                printf("Enter the position \n");
                scanf("%d",&pos);
                insertrandom(ele,pos);
                break;
        case 4: display();
                break;
    }
    }while(choice!=5);
    return 0;
}
```

OUTPUT

```
1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5.exit
Enter your choice : 1
Enter the element to be inserted
4

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5.exit
Enter your choice : 1
Enter the element to be inserted
5

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5.exit
Enter your choice : 1
Enter the element to be inserted
7

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5.exit
Enter your choice : 2
Enter the element to be inserted
3

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5.exit
Enter your choice : 2
```

```
1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5.exit
Enter your choice : 2
Enter the element to be inserted
2

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5.exit
Enter your choice : 4

The contents of the list are :
2
3
4
5
7

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5.exit
Enter your choice : 3
Enter the element to be inserted
6
Enter the position
5

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5.exit
Enter your choice : 4
```

```
1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5.exit
Enter your choice : 4

The contents of the list are :
2
3
4
5
6
7

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5.exit
Enter your choice : 3
Enter the element to be inserted
8
Enter the position
9

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5.exit
Enter your choice : 4

The contents of the list are :
2
3
4
5
6
7
8
```

```
The contents of the list are :
2
3
4
5
6
7
8

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5.exit
Enter your choice : 3
Enter the element to be inserted
10
Enter the position
7

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5.exit
Enter your choice : 4

The contents of the list are :
2
3
4
5
6
7
10
8

1. Inset at end
2.Insert at front
```

```
1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5.exit
Enter your choice : 4

The contents of the list are :
2
3
4
5
6
7
10
8

1. Inset at end
2.Insert at front
3.Insert at random position
4. Display
5.exit
Enter your choice : 5
```

LAB PROGRAM 6: WAP to Implement Singly Linked List with following operations a) a) Create a linked list. b) Deletion of first element, specified element and last element in the list. c) Display the contents of the linked list

PROGRAM CODE

```
#include<stdio.h>

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

struct node *head=NULL;

int length=0;

void insertend(int ele)
{
    struct node *newnode,*temp;
    newnode=(struct node*)malloc(sizeof(struct node));
    newnode->data=ele;
    newnode->next=NULL;
    if(head==NULL)
    {
        head=newnode;
        length=1;
    }
    else
    {
        temp=(struct node*)malloc(sizeof(struct node));
        temp=head;
        while(temp->next!=NULL)
        {
            temp=temp->next;
        }
        temp->next=newnode;
        length++;
    }

}


void deletefront()
{
    if(length==0)
    {
        printf("\nList is empty.\n");
    }
    else
    {   struct node *temp;
        temp=(struct node*)malloc(sizeof(struct node));
        temp=head;
        head=head->next;
        temp->next=NULL;
        length--;
```

```
        printf("\nThe element deleted is : %d",temp->data);
    }
}

void deleteend()
{
    if(length==0)
    {
        printf("\nList is empty.\n");
    }
    else
    {   struct node *temp;
        temp=(struct node*)malloc(sizeof(struct node));
        temp=head;
        while(temp->next->next!=NULL)
        {
            temp=temp->next;
        }
        struct node *del;
        del=(struct node*)malloc(sizeof(struct node));
        del=temp->next;
        temp->next=NULL;
        length--;
        printf("\nThe element deleted is : %d",del->data);
    }
}

void deleterandom(int pos)
{
    if(length==0)
        printf("\nList is empty.\n");
    else if(pos==1)
        deletefront();
    else if(pos>=length+1)
        deleteend();
    else
    {
        struct node *del;
        del=(struct node*)malloc(sizeof(struct node));
        struct node *temp;
        temp=(struct node*)malloc(sizeof(struct node));
        temp=head;
        for(int i=1;i<pos-1;i++)
        {
              temp=temp->next;
        }
        del=temp->next;
        temp->next=del->next;
        del->next=NULL;
        length--;
        printf("\nThe element deleted is : %d",del->data);

    }

}

void display()
{
    struct node *temp;
    temp=(struct node*)malloc(sizeof(struct node));
    temp=head;
```

```
    if(temp==NULL)
    {
        printf("\n List is empty \n");
    }
    else
    {
        printf("\nThe contents of the list are :\n");
        while(temp!=NULL)
        {
            printf("%d\n",temp->data);
            temp=temp->next;
        }
    }

}

int main()
{
    int choice,ele,pos;
    char ch;
    do
    {
    printf("\n1. Insert at end \n2. Display  \n3. Delete at front \n4.Delete at end
\n5.Delete at random \n6.exit");
    printf("\nEnter your choice : ");
    scanf("%d",&choice);
    switch(choice)
    {
        case 1: printf("Enter the element to be inserted\n");
                scanf("%d",&ele);
                insertend(ele);
                break;
        case 2: display();
                break;
        case 3: deletefront();
                break;
        case 4: deleteend();
                break;
        case 5: printf("\nEnter the position : ");
                scanf("%d",&pos);
                deleterandom(pos);
                break;
    }
    }while(choice!=6);
    return 0;
}
```

OUTPUT

```
1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 1
Enter the element to be inserted
3

1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 1
Enter the element to be inserted
4

1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 1
Enter the element to be inserted
5

1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 1
Enter the element to be inserted
6

1. Insert at end
2. Display
3. Delete at front
```

```
1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 2

The contents of the list are :
3
4
5
6

1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 4

The element deleted is : 6
1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice :
2

The contents of the list are :
3
4
5

1. Insert at end
```

```
1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 3

The element deleted is : 3
1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 2

The contents of the list are :
4
5

1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 5

Enter the position : 2

The element deleted is : 5
1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 2

The contents of the list are :
4
```

```
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 2

The contents of the list are :
4

1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 3

The element deleted is : 4
1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 3

List is empty.

1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 4

List is empty.

1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
```

```
1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 3

List is empty.

1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 4

List is empty.

1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 5

Enter the position : 2

List is empty.

1. Insert at end
2. Display
3. Delete at front
4.Delete at end
5.Delete at random
6.exit
Enter your choice : 6
```

LAB PROGRAM 7 : WAP Implement Single Link List with following operations a) a) Sort the linked list. b) Reverse the linked list. c) Concatenation of two linked lists

PROGRAM CODE

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

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

void insertAtEnd(struct node**head,int d){
      struct node *temp,*n;
      if(*head == NULL){
            temp = (struct node*)malloc(sizeof(struct node));
            temp->data = d;
            temp->next = NULL;
            *head = temp;
      }
      else {
            temp = *head;
            //go to the last node
            while(temp->next!=NULL){
                  temp = temp->next;
            }
            //adding node at the end
            n = (struct node*)malloc(sizeof(struct node));
            n->data = d;
            n->next = NULL;
            temp->next = n;
      }
}
void reverse(struct node**head) {
      struct node *prev,*cur,*next1;
      cur = *head;
     prev= NULL;
       next1=NULL;
      if(*head == NULL) {
            printf("Empty LIST\n");
            return;
      }

      while(cur!=NULL) {
        next1=cur->next;
        cur->next=prev;
        prev=cur;
            cur=next1;
      }
      *head = prev;
}

void concat(struct node**head1,struct node**head2){
      if(*head1==NULL) {
```

```
            *head1 = *head2;
            return;
      }
      if(*head2==NULL) {
            *head2 = *head1;
            return;
      }
      struct node*temp = *head1;
      while(temp->next!=NULL) {
            temp = temp->next;
      }
      temp->next = *head2;
}

struct node* merger(struct node*a,struct node*b) {
      //base case
      if(a==NULL) {
            return b;
      }
      if(b==NULL) {
            return a;
      }

      struct node*c = NULL;
      //rec case
      if(a->data < b->data) {
            c = a;
            c->next = merger(a->next,b);
      }
      else{
            c = b;
            c->next = merger(a,b->next);
      }
      return c;
}

struct node* MidPoint(struct node*head){
      if(head == NULL || head->next == NULL){
            return head;
      }

      struct node*fast = head->next;
      struct node*slow = head;

      while(fast != NULL && fast->next != NULL){
            fast = fast->next->next;
            slow = slow->next;
      }
      return slow;
}

struct node* MergeSort(struct node*head){
      if(head == NULL || head->next == NULL) {
            return head;
      }
      //rec case
      //1. Breaking into 2
      struct node* mid = MidPoint(head);
      struct node*a = head;
      struct node*b = mid->next;
```

```
	mid->next = NULL;

	//2. rec sort the two parts
	a = MergeSort(a);
	b = MergeSort(b);

	//3. Merging them
	struct node* c = merger(a,b);
	return c;
}

void display(struct node *head){
	while(head!=NULL){
		printf("%d-->",head->data);
		head = head->next;
	}
	printf("\n");
}


int main()
{
	struct node *head1=NULL,*head2=NULL,*head3=NULL,*head4=NULL,*ans=NULL;
	int data,n;
	printf("----SORTING----\n");
	printf("Enter the list to be sorted(Enter -1 to stop): \n");
	scanf("%d",&data);
	while(data!=-1) {
		insertAtEnd(&head1,data);
		scanf("%d",&data);
	}
	printf("List before sorting: ");
	display(head1);
	ans = MergeSort(head1);
	printf("List after sorting: ");
	display(ans);

	printf("\n----REVERSE----\n");
	printf("Enter the list to be reversed(Enter -1 to stop): \n");
	scanf("%d",&data);
	while(data!=-1) {
		insertAtEnd(&head2,data);
		scanf("%d",&data);
	}
	printf("List before reversing: ");
	display(head2);
	reverse(&head2);
	printf("List after reversing: ");
	display(head2);

	printf("\n----CONCATENATION----\n");
	printf("Enter the first list(Enter -1 to stop): \n");
	scanf("%d",&data);
	while(data!=-1) {
		insertAtEnd(&head3,data);
		scanf("%d",&data);
	}
	printf("Enter the second list(Enter -1 to stop): \n");
	scanf("%d",&data);
	while(data!=-1) {
		insertAtEnd(&head4,data);
```

```
            scanf("%d",&data);
        }

        printf("First List: ");
        display(head3);
        printf("Second List: ");
        display(head4);
        concat(&head3,&head4);
        printf("Concatenated List: ");
        display(head3);



        return 0;
}
```

OUTPUT

```
Enter the list to be sorted(Enter -1 to stop):
3
2
1
4
5
-1
List before sorting: 3-->2-->1-->4-->5-->
List after sorting: 1-->2-->3-->4-->5-->

----REVERSE----
Enter the list to be reversed(Enter -1 to stop):
4
3
2
1
-1
List before reversing: 4-->3-->2-->1-->
List after reversing: 1-->2-->3-->4-->

----CONCATENATION----
Enter the first list(Enter -1 to stop):
2
3
1
4
-1
Enter the second list(Enter -1 to stop):
5
7
6
8
-1
First List: 2-->3-->1-->4-->
Second List: 5-->7-->6-->8-->
Concatenated List: 2-->3-->1-->4-->5-->7-->6-->8-->
```

LAB PROGRAM 8 : WAP to implement Stack & Queues using Linked Representation

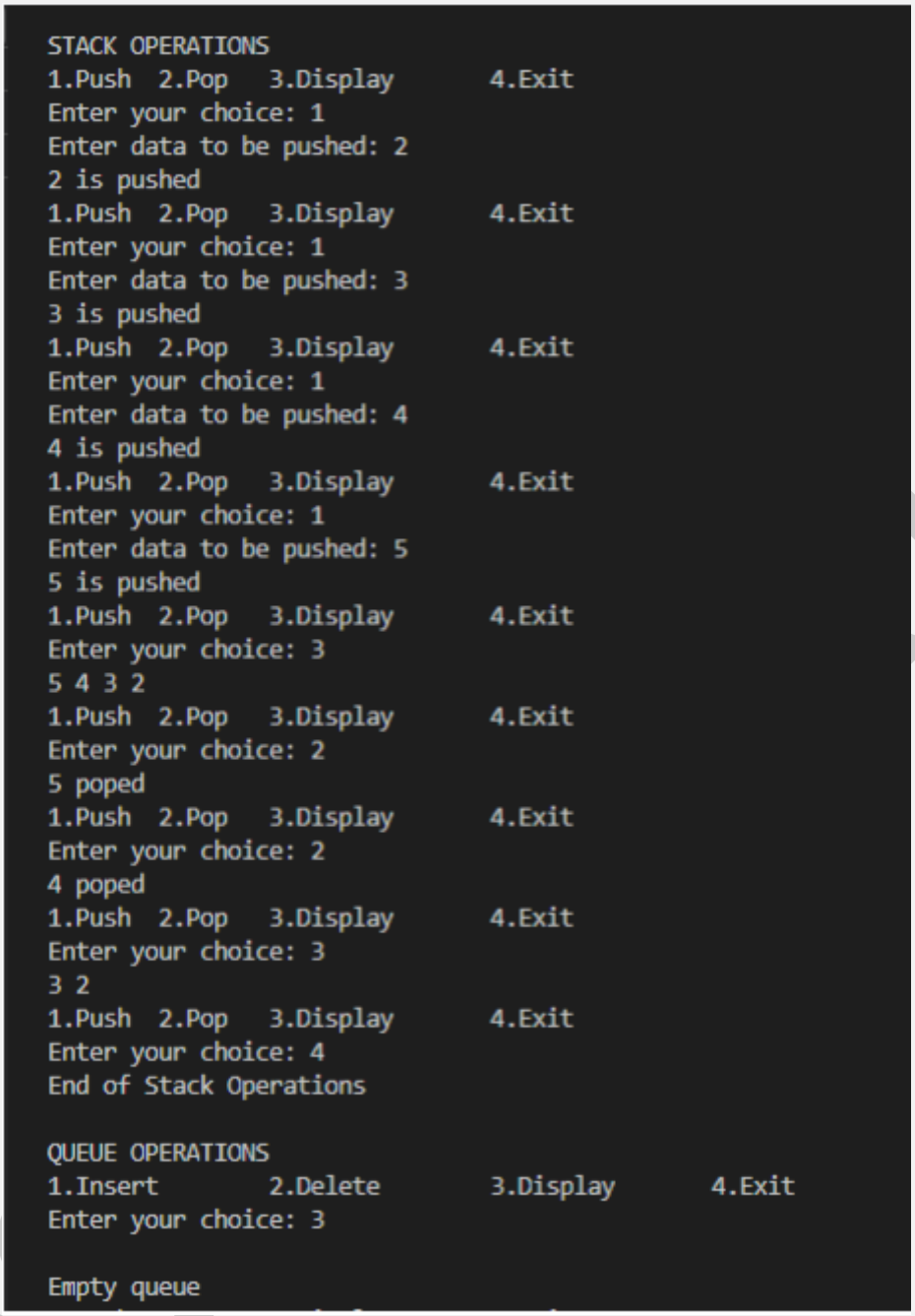
PROGRAM CODE

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

struct node{
      int data;
      struct node*next;
};
struct node*front;
struct node*rear;

void push(struct node**top,int d) {
      struct node*temp,n;

      temp = (struct node*)malloc(sizeof(struct node));

      if(temp == NULL) {
            printf("Stack is full\n");
      }

      temp->data = d;
      temp->next = *top;
      *top = temp;
      printf("%d is pushed\n",d);
}

void pop(struct node**top) {
      struct node*temp;

      if(*top==NULL) {
            printf("Stack Underflow\n");
            return;
      }

      temp = *top;
      printf("%d poped\n",temp->data);
      *top = (*top)->next;

      free(temp);
}

void display(struct node* top) {
      if(top == NULL){
            printf("No Elements Present in Stack\n");
            return;
      }

      while(top!=NULL) {
            printf("%d ",top->data);
            top = top->next;
      }
      printf("\n");
}
```

```
void insert(int d) {
      struct node*n;
      n = (struct node*)malloc(sizeof(struct node));
      if(n == NULL){
            printf("Queue Overflow\n");
            return;
      }
      n->data = d;
      if(front==NULL) {
            front = n;
            rear = n;
            front->next = NULL;
            rear->next = NULL;
      }
      else {
            rear->next = n;
            rear = n;
            rear->next = NULL;
      }
      printf("%d is inserted\n",d);
}

void delete() {
      struct node*temp;
      if(front == NULL) {
            printf("Queue Underflow\n");
            return;
      }
      temp = front;
      printf("%d deleted\n",temp->data);
      front = front->next;
      free(temp);
}

void display_queue() {

    struct node *temp;
    temp = front;
    if(front == NULL)
    {
        printf("\nEmpty queue\n");
    }
    else
    {   printf("\nQueue Elements: \n");
        while(temp != NULL)
        {
            printf("%d ",temp -> data);
            temp = temp -> next;
        }
        printf("\n");
    }
}

int main() {
      struct node*stack = NULL;
      printf("STACK OPERATIONS\n");
      printf("1.Push\t2.Pop\t3.Display\t4.Exit\n");
      int choice,item;
      printf("Enter your choice: ");
      scanf("%d",&choice);
      while(choice!=4) {
```

```
            switch(choice) {
                  case 1: printf("Enter data to be pushed: ");
                              scanf("%d",&item);
                              push(&stack,item);
                              break;

                  case 2: pop(&stack);
                              break;

                  case 3: display(stack);
                              break;
            }
            printf("1.Push\t2.Pop\t3.Display\t4.Exit\n");
            printf("Enter your choice: ");
            scanf("%d",&choice);
      }
      printf("End of Stack Operations\n\n");

      printf("QUEUE OPERATIONS\n");
      printf("1.Insert\t2.Delete\t3.Display\t4.Exit\n");
      printf("Enter your choice: ");
      scanf("%d",&choice);
      while(choice!=4) {
            switch(choice) {
                  case 1: printf("Enter data to be inserted: ");
                              scanf("%d",&item);
                              insert(item);
                              break;

                  case 2: delete();
                              break;

                  case 3: display_queue();
                              break;
            }
            printf("1.Push\t2.Pop\t3.Display\t4.Exit\n");
            printf("Enter your choice: ");
            scanf("%d",&choice);
      }
      printf("End Of Queue Operations\n");
      return 0;
}
```

OUTPUT


STACK OPERATIONS
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 1
Enter data to be pushed: 2
2 is pushed
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 1
Enter data to be pushed: 3
3 is pushed
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 1
Enter data to be pushed: 4
4 is pushed
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 1
Enter data to be pushed: 5
5 is pushed
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 3
5 4 3 2
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 2
5 poped
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 2
4 poped
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 3
3 2
1.Push 2.Pop 3.Display 4.Exit
Enter your choice: 4
End of Stack Operations
QUEUE OPERATIONS
1.Insert 2.Delete 3.Display 4.Exit
Enter your choice: 3
Empty queue

```
Empty queue
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 1
Enter data to be inserted: 2
2 is inserted
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 1
Enter data to be inserted: 3
3 is inserted
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 1
Enter data to be inserted: 4
4 is inserted
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 1
Enter data to be inserted: 5
5 is inserted
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 3

Queue Elements:
2 3 4 5
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 2
2 deleted
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 2
3 deleted
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 3

Queue Elements:
4 5
1.Push  2.Pop   3.Display       4.Exit
Enter your choice: 4
End Of Queue Operations
```

LAB PROGRAM 9: ***WAP Implement doubly link list with primitive operations a) a) Create a doubly linked list. b) Insert a new node to the left of the node. b) c) Delete the node based on a specific value. c) Display the contents of the list***

PROGRAM CODE

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

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

int length(node*head) {
	int count = 0;
	while(head!=NULL) {
		head = head->next;
		count++;
	}
	return count;
}
void insertAtEnd(node**head,int d) {
	node *n,*temp=*head;

	if(*head==NULL) {
		 *head = (node*)malloc(sizeof(node));
		(*head) ->prev = NULL;
		(*head) ->data = d;
		(*head) ->next = NULL;
```

```
	}

	else {
		//traverse to reach the last node
		while(temp->next!=NULL) {
			temp = temp->next;
		}

		n = (node*)malloc(sizeof(node));
		n->data = d;
		n->next = NULL;
		n->prev = temp;
		temp->next = n;
	}
	printf("%d was inserted in the list\n",d);
}

void insertLeft(node**h,int d,int ele) {
	node*head = *h;
	if(head->data == ele) {
		node*temp1 = NULL;
		temp1 = (node*)malloc(sizeof(node));
		temp1->prev = NULL;
		temp1->data = d;
		temp1->next = *h;
		(*h)->prev = temp1;
		*h = temp1;
		printf("%d was inserted at start\n",d);
		return;
	}
```

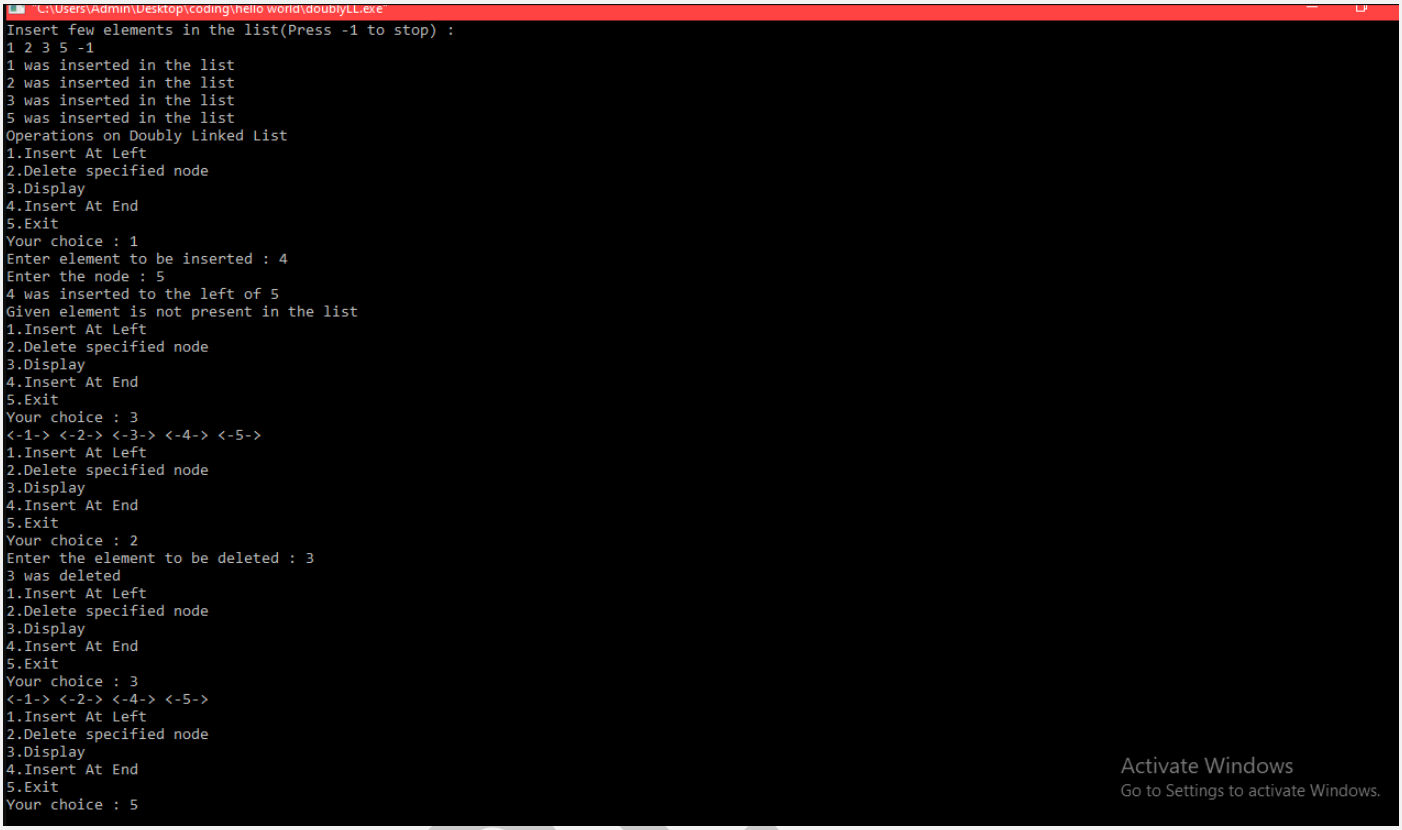
```
	node*temp;
	while(head!=NULL) {
		if(head->data == ele) {
			head = head->prev;
			temp = (node*)malloc(sizeof(node));
			temp->data = d;
			temp->prev = head;
			temp->next = head->next;
			temp->next->prev = temp;
			head->next = temp;
			printf("%d was inserted to the left of %d\n",d,ele);
			break;
		}
		else {
			head = head->next;
		}
	}
	printf("Given element is not present in the list\n");
}

void deleteNode(node**head,int d) {
	node *temp = *head;

	if(*head == NULL) {
		printf("No elements in the list to delete\n");
		return;
	}
	while(temp!=NULL) {
		if(temp->data == d) {
			if(temp == *head) {
				*head = (*head)->next;
```

```
                        (*head)->prev = NULL;
                }
                else if(temp->next == NULL) {
                        temp->prev->next = NULL;
                        free(temp);
                }
                else {
                        temp->prev->next = temp->next;
                        temp->next->prev = temp->prev;
                        free(temp);
                }
                printf("%d was deleted\n", d);
                return;
        }
        temp = temp->next;
    }
    printf("%d is not present in the list\n",d);
}



void display(node*head) {
    if(head==NULL) {
        printf("Empty List\n");
        return;
    }
    while(head!=NULL) {
        printf("<-%d-> ",head->data);
        head = head->next;
    }
    printf("\n");
}
```

```
int main() {
	node*head = NULL;
	int data,pos,opt;
	printf("Insert few elements in the list(Press -1 to stop) : \n");;
	scanf("%d",&data);
	while(data!=-1) {
		insertAtEnd(&head,data);
		scanf("%d",&data);
	}
	printf("Operations on Doubly Linked List\n");
	printf("1.Insert At Left\n2.Delete specified node\n3.Display\n4.Insert At
End\n5.Exit\n");
	printf("Your choice : ");
	scanf("%d",&opt);

	while(opt!=5) {
		switch(opt) {
			case 1: printf("Enter element to be inserted : ");
				scanf("%d",&data);
				printf("Enter the node : ");
				scanf("%d",&pos);
				insertLeft(&head,data,pos);
				break;

			case 2: printf("Enter the element to be deleted : ");
				scanf("%d",&data);
				deleteNode(&head,data);
				break;

			case 3: display(head);
```

```
                        break;


                case 4: printf("Enter data to be inserted at end: ");
                        scanf("%d",&data);
                        insertAtEnd(&head,data);
            }
            printf("1.Insert At Left\n2.Delete specified node\n3.Display\n4.Insert
At End\n5.Exit\n");
        printf("Your choice : ");
        scanf("%d",&opt);
      }
}
```

OUTPUT


"C:\Users\Admin\Desktop\coding\hello world\doublyLL.exe"
Insert few elements in the list(Press -1 to stop) :
1 2 3 5 -1
1 was inserted in the list
2 was inserted in the list
3 was inserted in the list
5 was inserted in the list
Operations on Doubly Linked List
1.Insert At Left
2.Delete specified node
3.Display
4.Insert At End
5.Exit
Your choice : 1
Enter element to be inserted : 4
Enter the node : 5
4 was inserted to the left of 5
Given element is not present in the list
1.Insert At Left
2.Delete specified node
3.Display
4.Insert At End
5.Exit
Your choice : 3
<-1-> <-2-> <-3-> <-4-> <-5->
1.Insert At Left
2.Delete specified node
3.Display
4.Insert At End
5.Exit
Your choice : 2
Enter the element to be deleted : 3
3 was deleted
1.Insert At Left
2.Delete specified node
3.Display
4.Insert At End
5.Exit
Your choice : 3
<-1-> <-2-> <-4-> <-5->
1.Insert At Left
2.Delete specified node
3.Display
4.Insert At End
5.Exit
Your choice : 5
Activate Windows
Go to Settings to activate Windows.

LAB PROGRAM 10 : ***Write a program a) to construct a binary Search tree. b) To traverse the tree using all the methods i.e., in-order, pre-order and post order c) to display the elements in the tree.***

PROGRAM CODE

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

typedef struct binary_node{
      int data;
      struct binary_node *left;
      struct binary_node *right;
}node;

/*node* newnode(int d) {
      node*root = (node*)malloc(sizeof node)
}*/

void insert(node**root,int d) {
      if(*root == NULL) {
            (*root) = (node*)malloc(sizeof (node));
            (*root)->left = NULL;
            (*root)->data = d;
            (*root)->right = NULL;
      }
      else {
            if(d<((*root)->data)){
                  insert(&((*root)->left),d);
            }
            else
                  insert(&((*root)->right),d);
      }
}


void inorder(node *root) {
      if(root == NULL)
            return;


      inorder(root->left);
      printf("%d ",root->data);
      inorder(root->right);
}

void preorder(node *root) {
      if(root == NULL) {
            return;
      }

      printf("%d ",root->data);
```

```
	preorder(root->left);
	preorder(root->right);
}

void postorder(node *root) {
	if(root == NULL)
		return;

	postorder(root->left);
	postorder(root->right);
	printf("%d ",root->data);
}

bool search(node*root,int key) {
	if(root == NULL)
		return false;

	if(root->data == key) {
		return true;
	}

	else if(key < root->data) {
		return search(root->left,key);
	}

	else {
		return search(root->right,key);
	}
}

int main() {
	node*root = NULL;
	int choice;
	int d;

	printf("1.Insert in
BST\n2.PreOrder\n3.InOrder\n4.PostOrder\n5.Search\n6.Exit\n");
	printf("Your choice: ");
	scanf("%d",&choice);

	while(choice!=6) {
		switch(choice) {
			case 1: printf("Enter element to be inserted: ");
				scanf("%d",&d);
				insert(&root,d);
				printf("%d inserted in the tree\n",d);
				break;

			case 2: printf("PreOrder traversal is: \n");
				preorder(root);
				printf("\n");
				break;

			case 3: printf("Inorder traversal is \n");
				inorder(root);
				printf("\n");
				break;

			case 4: printf("PostOrder traversal is \n");
				postorder(root);
				printf("\n");
```

```
                        break;

                case 5: printf("Enter element to be searched ");
                        scanf("%d",&d);
                        if(search(root,d)) {
                              printf("Element found!\n");
                        }
                        else{
                              printf("Element not found\n");
                        }
            }
            printf("Your next choice: ");
            scanf("%d",&choice);
      }
      return 0;
}
```

OUTPUT

```
1.Insert in BST
2.PreOrder
3.InOrder
4.PostOrder
5.Search
6.Exit
Your choice: 1
Enter element to be inserted: 5
5 inserted in the tree
Your next choice: 1
Enter element to be inserted: 4
4 inserted in the tree
Your next choice: 1
Enter element to be inserted: 7
7 inserted in the tree
Your next choice: 1
Enter element to be inserted:
1
1 inserted in the tree
Your next choice: 8
Your next choice: 1
Enter element to be inserted: 9
9 inserted in the tree
Your next choice: 1
Enter element to be inserted: 2
2 inserted in the tree
Your next choice: 1
Enter element to be inserted: 10
10 inserted in the tree
Your next choice: 1
```

```
9 inserted in the tree
Your next choice: 1
Enter element to be inserted: 2
2 inserted in the tree
Your next choice: 1
Enter element to be inserted: 10
10 inserted in the tree
Your next choice: 1
Enter element to be inserted: 3
3 inserted in the tree
Your next choice: 2
PreOrder traversal is:
5 4 1 2 3 7 9 10
Your next choice: 3
Inorder traversal is
1 2 3 4 5 7 9 10
Your next choice: 4
PostOrder traversal is
3 2 1 4 10 9 7 5
Your next choice: 5
Enter element to be searched 6
Element not found
Your next choice: 5
Enter element to be searched 3
Element found!
Your next choice: 6
```