DATA STRUCTURES LAB WORK

Submitted By

Varun Gaur

CE – 2B (2022-23)

210410107077/ 21BECEG102

Sardar Vallabhbhai Institute of Technology (S.V.I.T.)

Vasad - 388306

INDEX/ CONTENTS

S. No	Title	Page No.
1.	Introduction To Pointers. Call By Value, Call By Reference	03
2.	Introduction to Dynamic Memory Allocation DMA Functions malloc (), calloc(), free()	05
3.	PUSH, POP, PEEP, CHANGE & DISPLAY	08
4.	Infix to Postfix	11
5.	INSERT, DELETE AND DISPLAY, Queue	15
6.	Insert, Delete and Display using Circular Queue	18
7.	Nodes Using Linked List	21
8.	Stack using Linked List	34
9.	Queues using Linked List	37
10.	Doubly Linked List	39
11.	Circular Linked List	43
12.	Bubble Sort	54
13.	Quick Sort	56
14.	Merge Sort	58
15.	Binary Tree	61

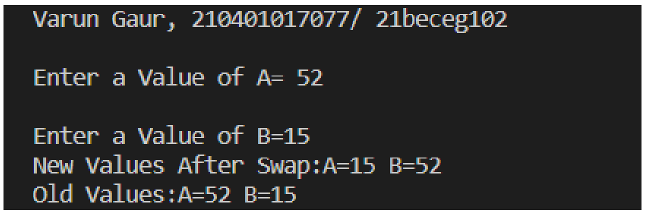
PRACTICAL I

Introduction to Pointers: Call By Value & Call By Reference

CALL BY VALUE

- In call by value method, the value of the actual parameters is copied into the formal parameters.
- In call by value method, we cannot modify the value of the actual parameter by the formal parameter.
- In call by value, different memory is allocated for actual and formal parameters since the value of the actual parameter is copied into the formal parameter.
- The actual parameter is the argument which is used in the function call whereas formal parameter is the argument which is used in the function definition.

```c
#include<stdio.h>
void swap(int,int);
int main()
{
    printf("Varun Gaur, 210401017077/ 21beceg102\n");
    int a,b;
    printf("\nEnter a Value of A=");
    scanf("%d",&a);
    printf("\nEnter a Value of B=");
    scanf("%d",&b);
    swap(a,b);
    printf("\nOld Values:");
    printf("A=%d B=%d \n",a,b);
}

void swap(int p,int q)
{
    int tmp;
    tmp=p;
    p=q;
    q=tmp;
    printf("New Values After Swap:");
    printf("A=%d B=%d",p,q);
}
```

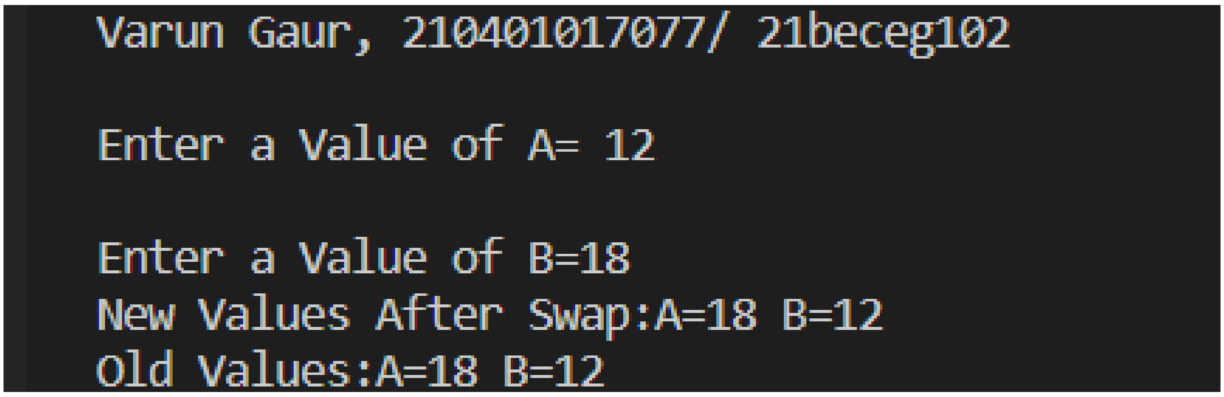


```
Varun Gaur, 210401017077/ 21beceg102

Enter a Value of A= 52

Enter a Value of B=15
New Values After Swap:A=15 B=52
Old Values:A=52 B=15
```

CALL BY REFERENCE

- In call by reference, the address of the variable is passed into the function call as the actual parameter.
- The value of the actual parameters can be modified by changing the formal parameters since the address of the actual parameters is passed.
- In call by reference, the memory allocation is similar for both formal parameters and actual parameters.
- All the operations in the function are performed on the value stored at the address of the actual parameters, and the modified value gets stored at the same address.

```c
#include<stdio.h>
void swap(int*,int*);
int main()
{
    printf("Varun Gaur, 210401017077/ 21beceg102\n");
    int a,b;
    printf("\nEnter a Value of A=");
    scanf("%d",&a);
    printf("\nEnter a Value of B=");
    scanf("%d",&b);
    swap(&a,&b);
    printf("\nOld Values:");
    printf("A=%d B=%d \n",a,b);
}
void swap(int *p , int *q)
{
    int tmp;
    tmp=*p;
    *p=*q;
    *q=tmp;
    printf("New Values After Swap:");
    printf("A=%d B=%d",*p,*q);
}
```



```
Varun Gaur, 210401017077/ 21beceg102

Enter a Value of A= 12

Enter a Value of B=18
New Values After Swap:A=18 B=12
Old Values:A=18 B=12
```

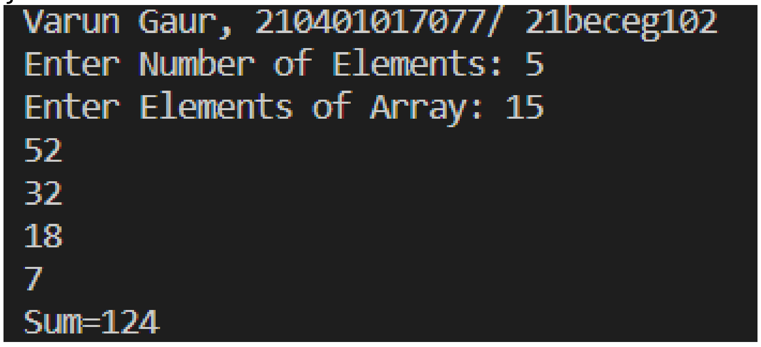
PRACTICAL II

Introduction to Dynamic Memory Allocation: malloc(), calloc(), free()

MALLOC ()

- malloc () is used to allocate a fixed amount of memory during the execution of a program
- malloc () allocates size in bytes of memory from heap, if the allocation succeeds, a pointer to the block of memory is returned else NULL is returned.
- Allocated memory space may not be contiguous.
- Each block contains a size, a pointer to the next block, and the space itself.
- The blocks are kept in ascending order of storage address, and the last block points to the first.
- The memory is not initialized. It initializes each block with default garbage value.

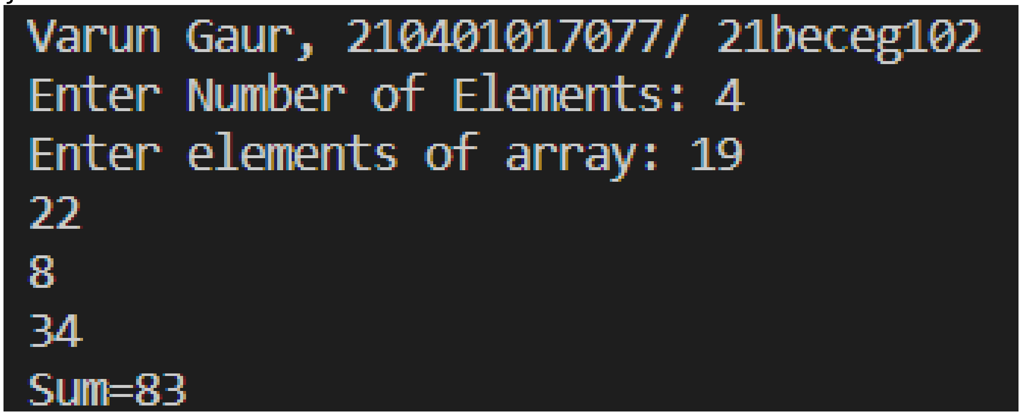
```
#include<stdio.h>
#include<stdlib.h>
int main(){
    printf("Varun Gaur, 210401017077/ 21beceg102\n");
    int *ptr;
    int n,i,sum=0;
    printf("Enter Number of Elements: ");
    scanf("%d",&n);
    ptr=(int*)malloc(n*sizeof(int));
    printf("Enter Elements of Array: ");
    for(i=0;i<n;++i)
    {
        scanf("%d",ptr+i);
        sum+=*(ptr+i);
    }
    printf("Sum=%d",sum);
    free(ptr);
    return 0;
}
```



```
Varun Gaur, 210401017077/ 21beceg102
Enter Number of Elements: 5
Enter Elements of Array: 15
52
32
18
7
Sum=124
```

CALLOC ()

```
#include<stdio.h>
#include<stdlib.h>
int main() {
    printf("Varun Gaur, 210401017077/ 21beceg102\n");
    int n,i,*ptr,sum=0;
    printf("Enter Number of Elements: ");
    scanf("%d",&n);
    ptr=(int*)calloc(n,sizeof(int)); //memory allocated using calloc
    printf("Enter elements of array: ");
    for(i=0;i<n;++i)
    {
        scanf("%d",ptr+i);
        sum+=*(ptr+i);
    }
    printf("Sum=%d",sum);
    free(ptr);
return 0;
}
```

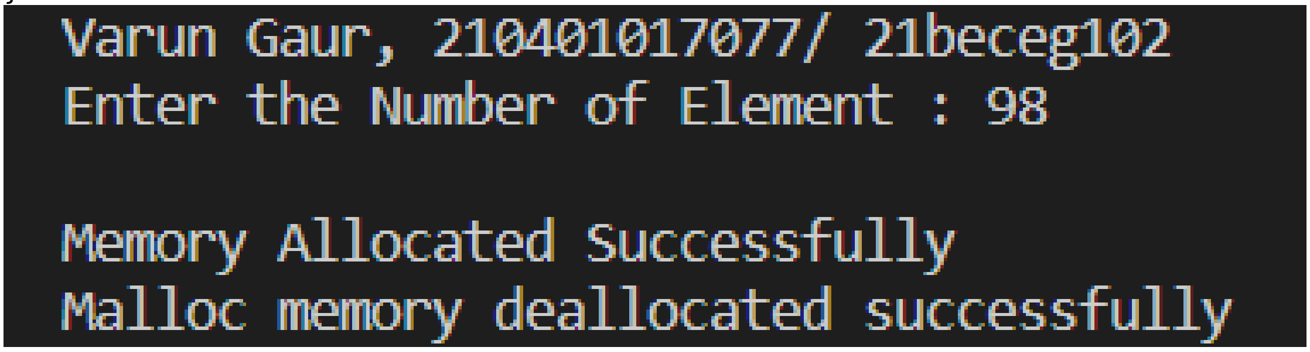


```
Varun Gaur, 210401017077/ 21beceg102
Enter Number of Elements: 4
Enter elements of array: 19
22
8
34
Sum=83
```

FREE ()

- “free” method in C is used to dynamically de-allocate the memory.
- The memory allocated using functions malloc() and calloc() is not de-allocated on their own.
- Hence the free() method is used, whenever the dynamic memory allocation takes place.
- It helps to reduce wastage of memory by freeing it.

```
#include<stdio.h>
#include<stdlib.h>
void main ()
{
    printf("Varun Gaur, 210401017077/ 21beceg102\n");
    int* ptr;
    int n, i;
    n = 5;
    printf("Enter the Number of Element : ");
    scanf("%d", &n);
    ptr = (int*) malloc (n*sizeof(int));
    if (ptr == NULL)
    {
        printf("\n Memory not allocated ");
    }
    else
    {
        printf("\nMemory Allocated Successfully");
        free(ptr);
        printf("\nMalloc memory deallocated successfully");
    }
}
```



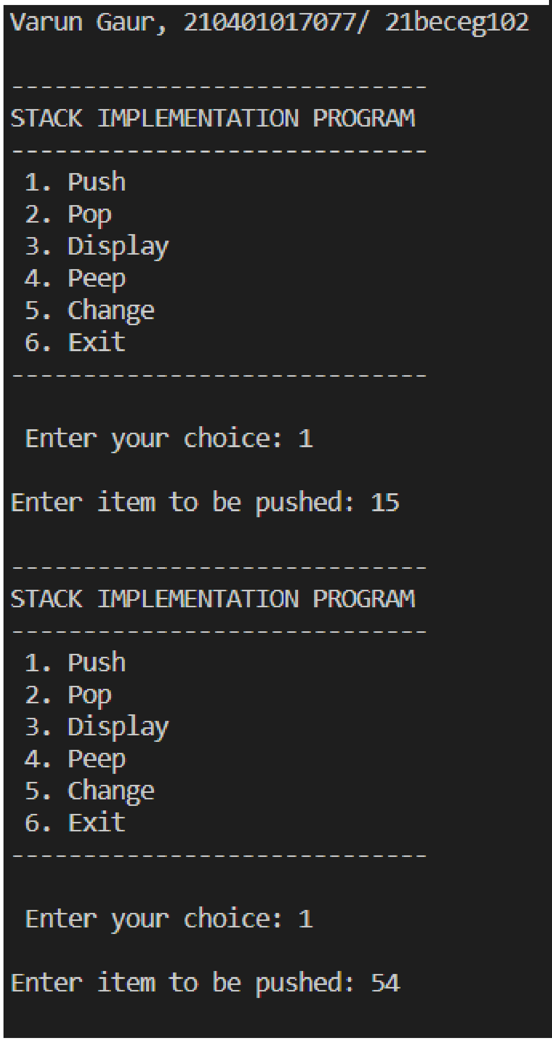
```
Varun Gaur, 210401017077/ 21beceg102
Enter the Number of Element : 98

Memory Allocated Successfully
Malloc memory deallocated successfully
```

PRACTICAL III

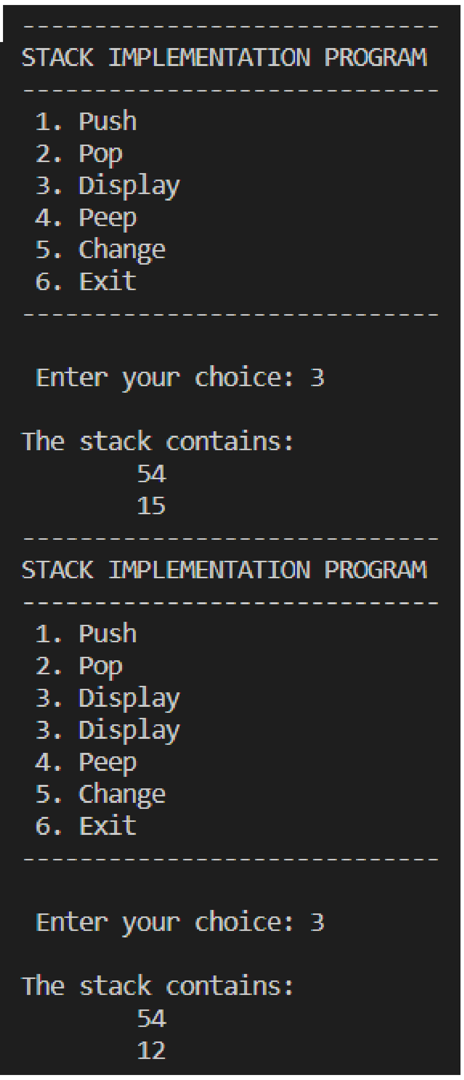
PUSH(), POP(), PEEP(), CHANGE(), DISPLAY()

Implement a program for stack that performs following operations using array. PUSH (b) POP (c) PEEP (d) CHANGE (e) DISPLAY

```
#include<stdio.h>
#define size 5
struct stack
{
    int a[size],top;
    int temp[size], tos;
}s;
void push(int item)
{
    s.a[++s.top] = item;
}
int pop()
{
    return s.a[s.top--];
}
void display()
{
    int i;
    printf("\nThe stack contains: ");
    for(i = s.top; i>=0; i--)
    {
    printf("\n\t%d", s.a[i]);
    }
}
void peep()
{
    printf("\n\tTop : %d", s.top);
    printf("\n\tValue: %d",s.a[s.top]);
}
void change(int row, int new_element)
{
    int i;
    int j = -1;
    printf("\n\tTop: %d", s.top);
    for(i=s.top; i>row; i--)
    {
        s.temp[++j] = s.a[s.top--];
    }
    s.a[s.top] = new_element;
    for(i = j; i>-1; i--)
    {
        s.a[++s.top] = s.temp[j--];
    }
}
int main()
{
    printf("Varun Gaur, 210401017077/ 21beceg102\n");
    s.top = -1;
    int item, choice, row, new_element;
    char ans;
    do{
    printf("\n----------------------------");
    printf("\nSTACK IMPLEMENTATION PROGRAM\n");
    printf("----------------------------");
    printf("\n 1. Push\n 2. Pop\n 3. Display\n 4. Peep\n 5. Change \n 6. Exit\n");
    printf("----------------------------\n");
    printf("\n Enter your choice: ");
    scanf("%d", &choice);
    switch(choice){
        case 1:
        if(s.top >= size-1)
        {
            printf("\nStack overflow..\n");
            break;
        }
        printf("\nEnter item to be pushed: ");
        scanf("%d", &item);
        push(item);
        break;

        case 2:
        if(s.top == -1)
        {
            printf("\n..Stack underflow..\n");
            break;
        }
            pop();
            break;

        case 3:
        display();
        break;

        case 4:
        peep();
        break;

        case 5:
        printf("\n\tEnter row no : ");
        scanf("%d",&row);
        printf("\n\tEnter new element: ");
        scanf("%d", &new_element);
        change(row, new_element );
        break;

        case 6:
        return 0;
 }
 }
 while(choice != 6);
 return 0;
}
```



```
Varun Gaur, 210401017077/ 21beceg102

-----------------------------
STACK IMPLEMENTATION PROGRAM
-----------------------------
 1. Push
 2. Pop
 3. Display
 4. Peep
 5. Change
 6. Exit
-----------------------------

 Enter your choice: 1

Enter item to be pushed: 15

-----------------------------
STACK IMPLEMENTATION PROGRAM
-----------------------------
 1. Push
 2. Pop
 3. Display
 4. Peep
 5. Change
 6. Exit
-----------------------------

 Enter your choice: 1

Enter item to be pushed: 54
```



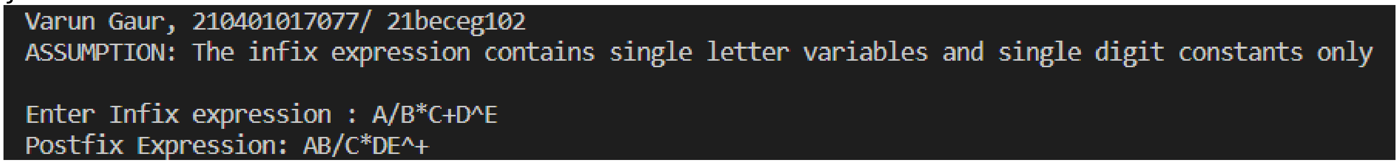
```
-----------------------------
STACK IMPLEMENTATION PROGRAM
-----------------------------
 1. Push
 2. Pop
 3. Display
 4. Peep
 5. Change
 6. Exit
-----------------------------

 Enter your choice: 3

The stack contains:
        54
        15
-----------------------------
STACK IMPLEMENTATION PROGRAM
-----------------------------
 1. Push
 2. Pop
 3. Display
 3. Display
 4. Peep
 5. Change
 6. Exit
-----------------------------

 Enter your choice: 3

The stack contains:
        54
        12
```

PRACTICAL IV

Infix to Postfix

Implement a program to convert infix notation to postfix notation using stack.

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<string.h>
#define SIZE 100
char stack[SIZE];
int top = -1;
void push(char item)
{
    if(top >= SIZE-1)
    {
        printf("\nStack Overflow.");
    }
    else
    {
        top = top+1;
        stack[top] = item;
        }
}
char pop()
{
    char item ;
    if(top <0)
    {
        printf("stack under flow: invalid infix expression");
        getchar();
        /* underflow may occur for invalid expression */
        /* where ( and ) are not matched */
        exit(1);
    }
    else
    {
        item = stack[top];
        top = top-1;
        return(item);
    }
}
int is_operator(char symbol)
{
    if(symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+' || symbol
=='-')
    {
        return 1;
    }
    else
    {return 0;}
}
int precedence(char symbol)
{
    if(symbol == '^')
    {return(3);}

    else if(symbol == '*' || symbol == '/')
    {return(2);}

    else if(symbol == '+' || symbol == '-')
    {return(1);}

    else
    {return(0);}
}
void InfixToPostfix(char infix_exp[], char postfix_exp[])
{
    int i, j;
    char item;
    char x;
    push('(');
    strcat(infix_exp,")");
    i=0; j=0;
    item=infix_exp[i];
    while(item != '\0')
    {
        if(item == '(')
        {
            push(item);
        }
        else if( isdigit(item) || isalpha(item))
        {
            postfix_exp[j] = item;
            j++;
            }
            else if(is_operator(item) == 1)
            {
                x=pop();
                while(is_operator(x) == 1 && precedence(x)>= precedence(item))
                {
                    postfix_exp[j] = x; j++;
                    x = pop();
                }
                push(x);
                push(item);
            }

            else if(item == ')')
            {
                x = pop();
                while(x != '(')
                {
                    postfix_exp[j] = x;
                    j++;
                    x = pop();
                }
            }

        else
        {
            printf("\nInvalid infix Expression.\n");
            getchar();
            exit(1);
        }

        i++;
        item = infix_exp[i];
    }
    if(top>0)
    {
        printf("\nInvalid infix Expression.\n");
        getchar();
        exit(1);
    }
    if(top>0)
    {
        printf("\nInvalid infix Expression.\n");
        getchar();
        exit(1);
    }
    postfix_exp[j] = '\0';
    }
int main()
{
    printf("Varun Gaur, 210401017077/ 21beceg102\n");

    char infix[SIZE], postfix[SIZE];
    printf("ASSUMPTION: The infix expression contains single letter variables and
single digit constants only \n");
    printf("\nEnter Infix expression : ");
    gets(infix);
    InfixToPostfix(infix,postfix);
    printf("Postfix Expression: ");
    puts(postfix);
    return 0;
}
```



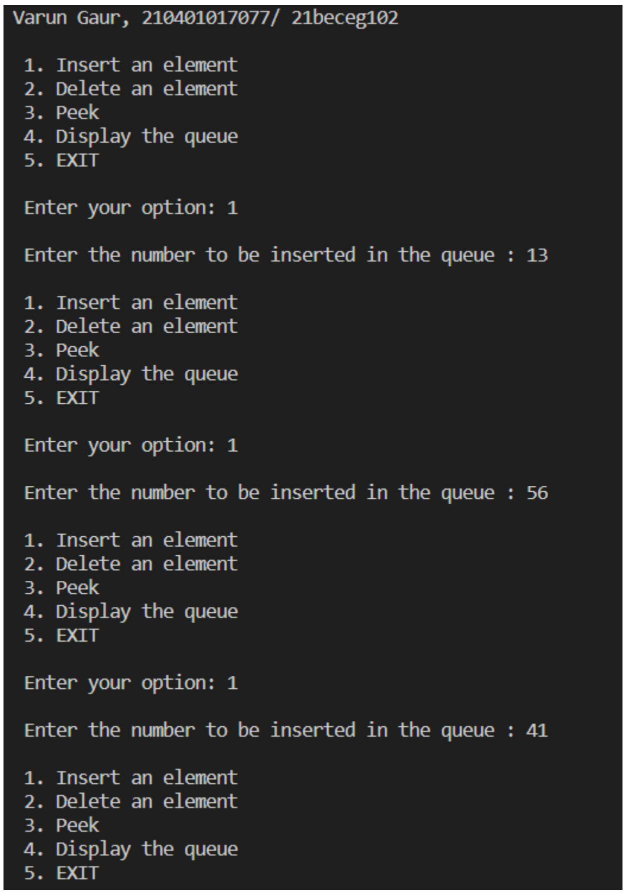
```
Varun Gaur, 210401017077/ 21beceg102
ASSUMPTION: The infix expression contains single letter variables and single digit constants only

Enter Infix expression : A/B*C+D^E
Postfix Expression: AB/C*DE^+
```

PRACTICAL V

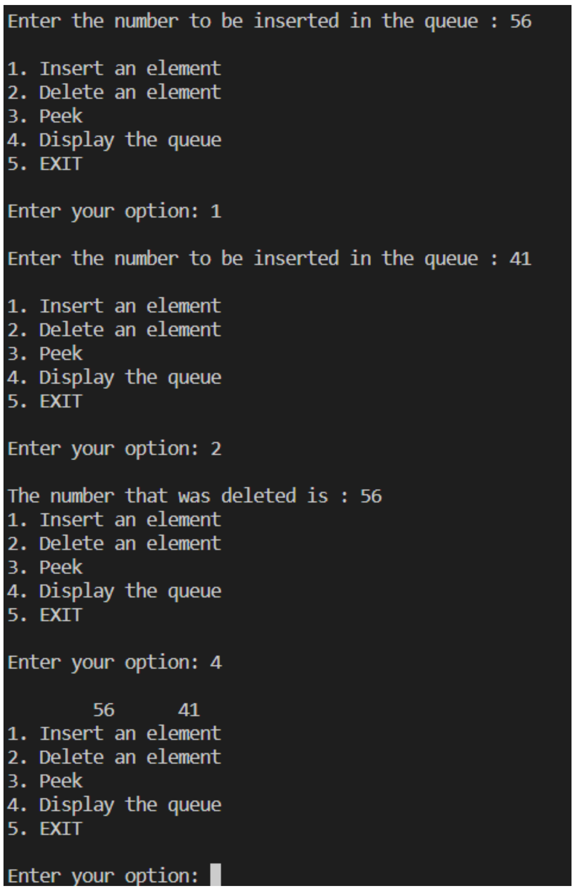
INSERT, DELETE AND DISPLAY using Queue

Write a program to implement QUEUE using arrays that performs the following operations (a)INSERT (b)DELETE (c)DISPLAY

```
#include<stdio.h>
#include<conio.h>
#define MAX 10
int queue[MAX];
int front = -1,rear = -1;
void insert(void);
int delete_element(void);
int peek(void);
void display(void);
void main()
{
    printf("Varun Gaur, 210401017077/ 21beceg102\n");
    int option, val;
    do
    {
        printf("\n 1. Insert an element");
        printf("\n 2. Delete an element");
        printf("\n 3. Peek");
        printf("\n 4. Display the queue");
        printf("\n 5. EXIT");
        printf("\n \n Enter your option: ");
        scanf("%d",&option);
switch(option)
{
    case 1:
    insert();
    break;

    case 2:
    val= delete_element();
    printf("\n The number that was deleted is : %d",val);
    break;

    case 3:
    val=peek();
    printf("\n The first value in the queue is: %d",val);
    break;

    case 4:
    display();
    break;
}
}
while(option != 5);
   getch();
   return;
}
void insert()
{
    int num;
    printf(" \n Enter the number to be inserted in the queue : ");
    scanf("%d", &num);
    if(rear == MAX-1)
    printf("\n OVERFLOW");

    if (front == -1 && rear == -1)
    front = rear=0;

    else
    rear++;
    queue[rear] = num;
}
int delete_element()
{
    int val;
    if(front == -1 || front>rear)
    {
        printf("\n UNDERFLOW");
        return -1;
    }
    else
    {
        front++;
        val = queue[front];
        return val;
    }
}
int peek()
{
    return queue[front];
}
void display()
{
    int i;
    printf("\n");
    for(i = front; i<=rear;i++)
    printf("\t %d",queue[i]);
}
```



```
Varun Gaur, 210401017077/ 21beceg102

 1. Insert an element
 2. Delete an element
 3. Peek
 4. Display the queue
 5. EXIT

 Enter your option: 1

 Enter the number to be inserted in the queue : 13

 1. Insert an element
 2. Delete an element
 3. Peek
 4. Display the queue
 5. EXIT

 Enter your option: 1

 Enter the number to be inserted in the queue : 56

 1. Insert an element
 2. Delete an element
 3. Peek
 4. Display the queue
 5. EXIT

 Enter your option: 1

 Enter the number to be inserted in the queue : 41

 1. Insert an element
 2. Delete an element
 3. Peek
 4. Display the queue
 5. EXIT
```



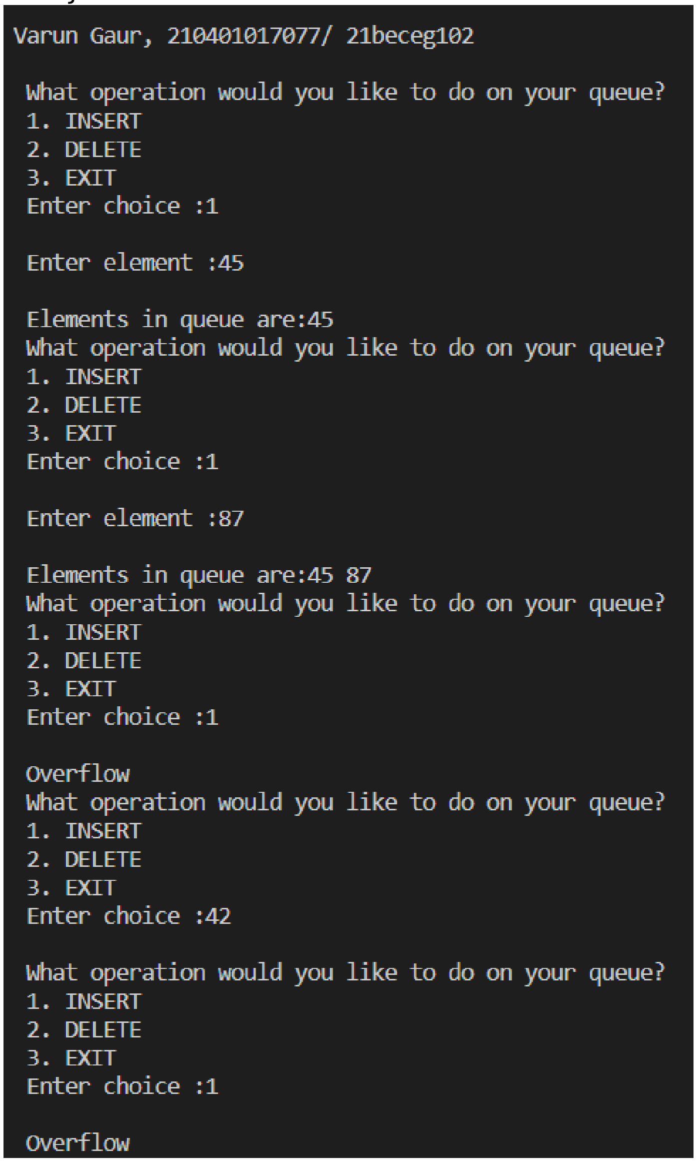
```
Enter the number to be inserted in the queue : 56

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. EXIT

Enter your option: 1

Enter the number to be inserted in the queue : 41

1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. EXIT

Enter your option: 2

The number that was deleted is : 56
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. EXIT

Enter your option: 4

        56      41
1. Insert an element
2. Delete an element
3. Peek
4. Display the queue
5. EXIT

Enter your option: 
```

PRACTICAL VI

INSERT, DELETE AND DISPLAY using Circular Queue

Write a program to implement Circular Queue using arrays that performs the following operations. (a)INSERT (b)DELETE (c)DISPLAY

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#define N 2
int r=-1,f=-1,queue[N];
void insert();
void del();
int i;
void main()
{
    printf("Varun Gaur, 210401017077/ 21beceg102\n");
    int ch;
    while(1)
    {
        printf("\n What operation would you like to do on your queue?");
        printf("\n 1. INSERT");
        printf("\n 2. DELETE");
        printf("\n 3. EXIT");
        printf("\n Enter choice :");
        scanf("%d",&ch);
        switch(ch)
        {
            case 1:
            insert();
            break;

            case 2:
            del();
            break;

            case 3:
            exit(0);
            }
        }
    }
void insert(void)

{
    if((f==0 && r==N-1) || (f==r+1))
    {
        printf("\n Overflow");
        return;
    }

    else
    {
        if(f==-1)
        f=r=0;

        else if(r==N-1)
        r=0;

        else
        r=r+1;

        printf("\n Enter element :");
        scanf("%d",&queue[r]);
        printf("\n Elements in queue are:");

        if(f<=r)
        for(i=f;i<=r;i++)
        printf("%d ",queue[i]);

        else
        {
            for(i=f;i<=N-1;i++)
            printf("%d ",queue[i]);

            for(i=0;i<=r;i++)
            printf("%d ",queue[i]);
        }
    }
}
void del(void)
{
    int item;
    if(f==-1)
    {
        printf("\n Underflow");
        return;
    }
    item = queue[f];

    if(f==r)
    f=r=-1;

    else if(f==N-1)
    f=0;
    else
    f=f+1;
    printf("\n Elements in queue are:");

    if(f<=r)
    for(i=f;i<=r;i++)
    printf("%d ",queue[i]);

    else
    {
        for(i=f;i<=N;i++)
        printf("%d ",queue[i]);
        for(i=1;i<=r;i++)
        printf("%d ",queue[i]);
        }
    }
```



```
Varun Gaur, 210401017077/ 21beceg102

 What operation would you like to do on your queue?
 1. INSERT
 2. DELETE
 3. EXIT
 Enter choice :1

 Enter element :45

 Elements in queue are:45
 What operation would you like to do on your queue?
 1. INSERT
 2. DELETE
 3. EXIT
 Enter choice :1

 Enter element :87

 Elements in queue are:45 87
 What operation would you like to do on your queue?
 1. INSERT
 2. DELETE
 3. EXIT
 Enter choice :1

 Overflow
 What operation would you like to do on your queue?
 1. INSERT
 2. DELETE
 3. EXIT
 Enter choice :42

 What operation would you like to do on your queue?
 1. INSERT
 2. DELETE
 3. EXIT
 Enter choice :1

 Overflow
```