STACK:

//ARRAY:

```
 #include<stdio.h>
#define MAX 5
int Stack[MAX], top = -1;
int IsFull();
int IsEmpty();
void Push(int ele);
void Pop();
void Top();
void Display();
int main()
{
int ch, e;
do
 {
 printf("1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT");
 printf("\nEnter your choice : ");
 scanf("%d", &ch);
 switch(ch)
 {
 case 1:
 printf("Enter the element : ");
 scanf("%d", &e);
 Push(e);
 break;
 case 2:
 Pop();
 break;
 case 3:
 Top();
 break;
 case 4:
 Display();
 break;
 }
 } while(ch <= 4);
return 0;
}
int IsFull()
{
if(top == MAX - 1)
 return 1;
else
 return 0;
```

```
}
int IsEmpty()
{
if(top == -1)
 return 1;
else
return 0;
}
void Push(int ele)
{
if(IsFull())
 printf("Stack
Overflow...!\n"); else
 {
 top = top + 1;
 Stack[top] = ele;
 }
}
void Pop()
{
if(IsEmpty())
 printf("Stack Underflow...!\n");
else
 {
 printf("%d\n", Stack[top]);
 top = top - 1;
 }
}
void Top()
{
if(IsEmpty())
 printf("Stack Underflow...!\n");
else
 printf("%d\n", Stack[top]);
}
void Display()
{
int i;
if(IsEmpty())
 printf("Stack Underflow...!\n");
else
 {
 for(i = top; i >= 0; i--)
 printf("%d\t", Stack[i]);
 printf("\n");
 }
}
```

OUTPUT:

```
1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
 Enter your choice : 1
 Enter the element : 10
 1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
 Enter your choice : 1
 Enter the element : 20
 1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
 Enter your choice : 1
 Enter the element : 30
 1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
 Enter your choice : 1
 Enter the element : 40
 1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
 Enter your choice : 1
 Enter the element : 50
 1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
 Enter your choice : 1
 Enter the element : 60
 Stack Overflow...!
 1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
 Enter your choice : 4
 50 40 30 20 10
 1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
 Enter your choice : 3
 50
 1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
 Enter your choice : 2
 50
 1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
 Enter your choice : 2
 40
 1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
 Enter your choice : 2
 30
 1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
 Enter your choice : 2
 20
 1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
 Enter your choice : 2
 10
 1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
 Enter your choice : 2
 Stack Underflow...!
```

```
1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
Enter your choice : 5
```

```
//LINKED LIST :
#include  <stdio.h>
#include <stdlib.h>
struct node
{
int Element;
struct node *Next;
}*List = NULL;
typedef struct node Stack;
int IsEmpty();
void Push(int e);
void Pop();
void Top();
void Display();
int main()
{
int ch, e;
do
 {
 printf("1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT");
 printf("\nEnter your choice : ");
 scanf("%d", &ch);
 switch(ch)
 {
 case 1:
 printf("Enter the element : ");
 scanf("%d", &e);
 Push(e);
 break;
 case 2:
 Pop();
 break;
 case 3:
 Top();
 break;
 case 4:
 Display();
 break;
 }
 } while(ch <= 4);
return 0;
}
int IsEmpty()
{
if(List == NULL)
```

```
 return 1;
else
return 0;
}
void Push(int e)
{
Stack *NewNode =
 malloc(sizeof(Stack)); NewNode-
 >Element = e;
if(IsEmpty())
 NewNode->Next = NULL;
else
 NewNode->Next = List;
 List = NewNode;
}
void Pop()
{
if(IsEmpty())
 printf("Stack is Underflow...!\n");
else
 {
 Stack*TempNode;
 TempNode = List;
 List = List->Next;
 printf("%d\n", TempNode->Element);
 free(TempNode);
 }
}
void Top()
{
if(IsEmpty())
 printf("Stack is Underflow...!\n");
else
 printf("%d\n", List->Element);
}
void Display()
{
if(IsEmpty())
 printf("Stack is Underflow...!\n");
else
 {
 Stack *Position;
 Position = List;
 while(Position != NULL)
 {
```

```
 printf("%d\t", Position->Element);
 Position = Position->Next;
 }
 printf("\n");
 }
}
```

OUTPUT:

```
1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
Enter your choice : 1
Enter the element : 10
1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
Enter your choice : 1
Enter the element : 20
1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
Enter your choice : 1
Enter the element : 30
1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
Enter your choice : 1
Enter the element : 40
1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
Enter your choice : 1
Enter the element : 50
1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
Enter your choice : 4
50 40 30 20 10
1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
Enter your choice : 3
50
1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
Enter your choice : 2
50
1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
Enter your choice : 2
40
1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
Enter your choice : 2
30
1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
Enter your choice : 2
20
1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
Enter your choice : 2
10
1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
Enter your choice : 2
Stack is
Underflow...!
    1.PUSH 2.POP 3.TOP 4.DISPLAY 5.EXIT
Enter your choice : 5
```

INFIX TO POSTFIX:

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 20
    int Stack[MAX], top = -1;
    char expr[MAX], post[MAX];
    void Push(char sym);
    char Pop();
    char Top();
    int Priority(char sym);
    int main()
    {
    int i;
    printf("Enter the infix expression : ");
    gets(expr);
    for(i = 0; i < strlen(expr); i++)
     {
     if(expr[i] >= 'a' && expr[i] <= 'z')
     printf("%c", expr[i]);
     else if(expr[i] == '(')
     Push(expr[i]);
     else if(expr[i] == ')')
     {
     while(Top() != '(')
     printf("%c", Pop());
     Pop();
     }
     else
     {
     while(Priority(expr[i])<=Priority(Top()) && top!=-
     1) printf("%c", Pop());
     Push(expr[i]);
     }
     }
    for(i = top; i >= 0; i--)
     printf("%c", Pop());
    return 0;
    }
    void Push(char sym)
    {
     top = top + 1;
     Stack[top] = sym;
    }
    char Pop()
```

```
{
char e;
 e = Stack[top];
top = top - 1;
return e;
}
char Top()
{
return Stack[top];;
}
int Priority(char sym)
{
int p = 0;
switch(sym)
 {
 case '(':
 p = 0;
 break;
 case '+':
 case '-':
 p = 1;
 break;
 case '*':
 case '/':
 case '%':
 p = 2;
 break;
 case '^':
 p = 3;
 break;
 }
return p;
}
```

OUTPUT:

```
Enter the infix expression : a/b^c+d*e-f*g
abc^/de*+fg*-
```

EVALUATION OF ARITHMETIC EXPRESSION:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 20
struct node
{
int Element;
struct node *Next;
}*List = NULL;
typedef struct node Stack;
void Push(int e);
int Pop();
int main()
{
int i, a, b, c, e;
char expr[MAX];
printf("Enter the postfix expression : ");
gets(expr);
for(i = 0; i < strlen(expr); i++)
 {
 if(expr[i]=='+'||expr[i]=='-'||expr[i] =='*'||expr[i]=='/')
 {
 b = Pop();
 a = Pop();
 switch(expr[i])
 {
 case '+':
 c = a + b;
 Push(c);
 break;
 case '-':
 c = a - b;
 Push(c);
 break;
 case '*':
 c = a * b;
 Push(c);
 break;
 case '/':
 c = a / b;
 Push(c);
 break;
 }
 }
 else
 {
 printf("Enter the value of %c : ", expr[i]);
 scanf("%d", &e);
 Push(e);
 }
 }
printf("The result is %d", Pop());
return 0;
}
void Push(int e)
{
Stack *NewNode = malloc(sizeof(Stack));
 NewNode->Element = e;
if(List == NULL)
NewNode->Next = NULL;
else
 NewNode->Next = List;
 List = NewNode;
}
int Pop()
{
int e;
Stack
 *TempNode;
 TempNode = List;
 List = List->Next;
 e = TempNode-
>Element;
free(TempNode);
return e;
}
```

OUTPUT:

```
Enter the postfix expression : abc+*d*
Enter the value of a : 2
Enter the value of b : 3
Enter the value of c : 4
Enter the value of d : 5
The result is 70
```

QUEUE:

//ARRAY:

```
#include <stdio.h>
#define MAX 5
int Queue[MAX], front = -1, rear = -1;
int IsFull();
int IsEmpty();
void Enqueue(int ele);
void Dequeue();
void Display();
int main()
{
int ch, e;
do
 {
 printf("1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT");
 printf("\nEnter your choice : ");
 scanf("%d", &ch);
 switch(ch)
 {
 case 1:
 printf("Enter the element : ");
 scanf("%d", &e);
 Enqueue(e);
 break;
 case 2:
 Dequeue();
 break;
 case 3:
 Display();
 break;
 }
 } while(ch <= 3);
return 0;
}
int IsFull()
{
if(rear == MAX - 1)
 return 1;
else
 return 0;
}
int IsEmpty()
{
if(front == -1)
```

```
 return 1;
else
return 0;
}
void Enqueue(int ele)
{
if(IsFull())
 printf("Queue is Overflow...!\n");
else
 {
 rear = rear + 1;
 Queue[rear] = ele;
 if(front == -1)
 front = 0;
 }
}
void Dequeue()
{
if(IsEmpty())
 printf("Queue is Underflow...!\n");
else
 {
 printf("%d\n", Queue[front]);
 if(front == rear)
 front = rear = -1;
 else
 front = front + 1;
 }
}
void Display()
{
int i;
if(IsEmpty())
 printf("Queue is Underflow...!\n");
else
 {
 for(i = front; i <= rear; i++)
 printf("%d\t", Queue[i]);
 printf("\n");
 }
}
```

OUTPUT:

```
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 1
Enter the element : 10
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 1
Enter the element : 20
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 1
Enter the element : 30
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 1
Enter the element : 40
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 1
Enter the element : 50
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 1
Enter the element : 60
Queue is Overflow...!
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 3
10 20 30 40 50
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 2
10
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 2
20
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 2
30
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 2
40
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 2
50
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 2
Queue is
Underflow...!
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 3
Queue Underflow...!
```

```
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 4
```

```
//LINKED LIST:
#include  <stdio.h>
#include <stdlib.h>
struct node
{
int Element;
struct node *Next;
}*Front = NULL, *Rear = NULL;
typedef struct node Queue;
int IsEmpty(Queue *List);
void Enqueue(int e);
void Dequeue();
void Display();
int main()
{
int ch, e;
do
 {
 printf("1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT");
 printf("\nEnter your choice : ");
 scanf("%d", &ch);
 switch(ch)
 {
 case 1:
 printf("Enter the element : ");
 scanf("%d", &e);
 Enqueue(e);
 break;
 case 2:
 Dequeue();
 break;
 case 3:
 Display();
 break;
 }
 } while(ch <= 3);
return 0;
}
int IsEmpty(Queue *List)
{
if(List == NULL)
 return 1;
else
 return 0;
}
```

```
void Enqueue(int e)
{
Queue *NewNode =
 malloc(sizeof(Queue)); NewNode-
 >Element = e;
 NewNode->Next = NULL;
if(Rear == NULL)
 Front = Rear = NewNode;
else
 {
 Rear->Next = NewNode;
 Rear = NewNode;
 }
}
void Dequeue()
{
if(IsEmpty(Front))
 printf("Queue is Underflow...!\n");
else
 {
 Queue *TempNode;
 TempNode = Front;
 if(Front == Rear)
 Front = Rear = NULL;
 else
 Front = Front->Next;
 printf("%d\n", TempNode->Element);
 free(TempNode);
 }
}
void Display()
{
if(IsEmpty(Front))
 printf("Queue is Underflow...!\n");
else
 {
 Queue *Position;
 Position = Front;
 while(Position != NULL)
 {
 printf("%d\t", Position->Element);
 Position = Position->Next;
 }
 printf("\n");
 }}
```

OUTPUT:

```
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 1
Enter the element : 10
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 1
Enter the element : 20
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 1
Enter the element : 30
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 1
Enter the element : 40
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 1
Enter the element : 50
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 3
10 20 30 40 50
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 2
10
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 2
20
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 2
30
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 2
40
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 2
50
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 2
Queue is
Underflow...!
1.ENQUEUE 2.DEQUEUE 3.DISPLAY 4.EXIT
Enter your choice : 4
```