Program 1

```
#include<stdio.h>
#include<string.h>
#include<ctype.h>
#define MAX 20
typedef struct stack
{
	int a[MAX];
	int top;
}st;
void init(st *p)
{
	p->top=-1;
}
int emp(st *p)
{
	return p->top==-1;
}
int overflow(st *p)
{
	return p->top==MAX-1;
}
void push(st *p,int key)
{
	if(overflow(p))
	{
		printf("its full\n");
	}
	else
	{
		p->top++;
```

```
		p->a[p->top]=key;
	}
}
int pop(st *p)
{
	if(emp(p))
	{
		printf("Its empty\n");
		return -1;
	}
	else
	{
		return p->a[p->top--];
	}
}
int peep(st *p)
{
	if(emp(p))
	{
		printf("Its empty\n");
		return -1;
	}
	else
	{
		return p->a[p->top];
	}
}
int next(char ch)
{
	switch(ch)
	{
```

```
                case '+':
                case '-': return 1;
                case '*':
                case '/': return 2;
                default: return 0;
        }
}
char* inf2pfx(char* infix, char* postfixx)
{
        st s;
        init(&s);
        char ch;
        char ch1;
        int j=0;
        for(int i = 0;i<strlen(infix); ++i)
        {
                ch = infix[i];
                switch(ch)
                {
                        case '(': push(&s, ch);
                                        break;

                        case ')': while((ch1 = pop(&s)) != '(')
                                                postfixx[j++] = ch1;

                                        break;
                        case '+':
                        case '-':
                        case '*':
                        case '/': while(!emp(&s) && next(peep(&s)) >= next(ch))
                                                postfixx[j++] = pop(&s);
```

```
                                        push(&s, ch);
                                                break;

                        default: postfixx[j++] = ch;

                }
        }

        while(!emp(&s))
        {
                postfixx[j++]=pop(&s);
        }
        postfixx[j]='\0';
        printf("postfix is %s\n",postfixx);
return(postfixx);

}

int popp(st *p, int *pe)
{
        if(emp(p))
                return 0;
        *pe=p->a[p->top];
        p->top--;
        return 1;
}

int main()
{
        st s;
```

```
int op1;int op2; int res;int i=0;
init(&s);
char infix[20];
printf("Enter input\n");
scanf("%s",infix); //2+3
char* postfix;
inf2pfx(infix,postfix);

while(postfix[i]!='\0')
{
        if(isdigit(postfix[i]))
        {
                push(&s,postfix[i]-'0');
        }
        else
        {
                popp(&s,&op2);
                popp(&s,&op1);
                switch(postfix[i])
                {
                        case '+' : res=op1+op2;
                                break;

                        case '-' : res=op1-op2;
                                break;

                        case '*' : res=op1*op2;
                                break;

                        case '/' : res=op1/op2;
                                break;
```

```
                        }
                        push(&s,res);
                }
                ++i;
        }
        popp(&s,&res);
        printf("eval is %d\n",res);
        return 0;
}
```

```
Enter input
5+9/8*(3-2/1)*5
postfix is 598/321/-*5*+
eval is 10
```

Program 2

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 20
typedef struct stack
{
   int a[MAX];
   int beg;
   int end;
} s;

void init(s *p)
{
   p->beg=MAX;
   p->end=0;
}
```

```
void push(s *p, int key, int code)
{
   if((p->end)>=(p->beg))
   {
      printf("Array is full. Insertion to either stack is not possible.\n");
   }
   else
   {
      if(code==1)
      {
         p->end++;
         p->a[p->end]=key;
      }
      else if(code==2)
      {
         p->beg--;
         p->a[p->beg]=key;
      }
      else
      {
         printf("Invalid type");
      }
   }
}

int pop(s *p, int code)
{
   int ele;
   if(code==1)
   {
      if(p->end==-1)
```

```
        {
            printf("This stack is empty\n");
            return -1;
        }
        else
        {
            p->end--;
            ele=p->a[p->end];
        }
    }
    else if(code==2)
    {
        if(p->beg==MAX)
        {
            printf("This stack is empty\n");
            return -1;
        }
        else
        {
            ele=p->a[p->beg];
            p->beg++;
        }
    }
    else
    {
        printf("Invalid type");
        return -1;
    }
    return ele;
}
```

```
int main()
{
   s s;
   init(&s);
   int choice,key;
   printf("Enter the number of your choice:\n1. Add to Container 1\n2.  Add to Container 2\n3. Remove from Container 1\n4. Remove from Container 2\nOr 0 to exit:\n");
   scanf("%d", &choice);
   while(choice)
   {
      switch(choice)
      {
         case 1:
               printf("Enter the element to push:\n");
               scanf("%d", &key);
               push(&s, key,1);
               break;
         case 2:
               printf("Enter the element to push:\n");
               scanf("%d", &key);
               push(&s, key, 2);
               break;
         case 3:
               key=pop(&s,1);
               if(key!=-1)
                  printf("Element is: %d\n", key);
               break;
          case 4:
               key=pop(&s,2);
               if(key!=-1)
                  printf("Element is: %d\n", key);
```

```
            break;
        default:
            printf("Invalid choice\n");
            break;


    }
    printf("Enter the number of your choice:\n1. Push to Stack 1\n2. Push to stack 2\n3. Pop from stack 1\n4. Pop from stack 2\nOr 0 to exit:\n");
    scanf("%d", &choice);
  }
  return 0;
}
```

```
Enter the number of your choice:
1. Add to Container 1
2.  Add to Container 2
3. Remove from Container 1
4. Remove from Container 2
Or 0 to exit:
1
Enter the element to push:
5
Enter the number of your choice:
1. Push to Stack 1
2. Push to stack 2
3. Pop from stack 1
4. Pop from stack 2
Or 0 to exit:
2
Enter the element to push:
4
Enter the number of your choice:
1. Push to Stack 1
2. Push to stack 2
3. Pop from stack 1
4. Pop from stack 2
Or 0 to exit:
1
Enter the element to push:
2
Enter the number of your choice:
1. Push to Stack 1
2. Push to stack 2
```

```
Enter the number of your choice:
1. Push to Stack 1
2. Push to stack 2
3. Pop from stack 1
4. Pop from stack 2
Or 0 to exit:
3
Element is: 5
Enter the number of your choice:
1. Push to Stack 1
2. Push to stack 2
3. Pop from stack 1
4. Pop from stack 2
Or 0 to exit:
4
Element is: 4
Enter the number of your choice:
1. Push to Stack 1
2. Push to stack 2
3. Pop from stack 1
4. Pop from stack 2
Or 0 to exit:
4
This stack is empty
Enter the number of your choice:
1. Push to Stack 1
2. Push to stack 2
3. Pop from stack 1
4. Pop from stack 2
Or 0 to exit:
```