```
/*  EVALUTION  OF  A  POSTFIX  EXPRESSION. Give space in between the operator
and operands  */
#include"stdio.h"
#include"string.h"
#define MAX 100
int stack[100],top=-1;
void push(int);
int pop(void);
int main()
      {
      int i,a,b,c,num;
      char s[MAX];
      char ch;
      printf("\n Enter a Postfix Expression:");
      gets(s);
      for(i=0;i<strlen(s);i++)
      {
      switch(s[i])
            {
            case '+':
                  a=pop();
                  b=pop();
                  c=a+b;
                  push(c);
                  i++;
                  break;
            case '-':
                  a=pop();
                  b=pop();
                  c=b-a;
                  push(c);
                  i++;
                  break;
            case '*':
                  a=pop();
                  b=pop();
                  c=b*a;
                  push(c);
                  i++;
                  break;
            case '/':
                  a=pop();
                  b=pop();
                  c=b/a;
                  push(c);
                  i++;
                  break;
            default:
                  num=0;
                  while(s[i]!=32)
                        {
                        num=num*10+ (s[i]-48);
                        i++;
                        }
                  push(num);
                 }
            }
      c=pop();
      printf("%d",c);
      return 0;
      }

void push(int num)
      {
      stack[++top] = num;
      }

int pop(void)
      {
      char c;
      c=stack[top];
      top=top-1;
      return c;
      }
```

```
/* Conversion of infix notation to postfix */
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#define MAX 100

void push(char);
char pop();

char stack[100],top=-1;

int main()
     {
     int i;
     char a[MAX];
     char ch;
     printf("\n Enter a Infix Expression:");
     gets(a);
     strcat(a,")");
     push('(');
     i=0;
     while(top!=-1)
           {
           switch(a[i])
                 {
                 case '(':
                       push('(');
                       break;

                 case ')':
                       do    {
                             ch=pop();
                             if(ch!='(')
                                   printf("%c",ch);
                             }while(ch!='(');
                       break;
                 case '+':
                       ch=pop();
                       while(ch!='(')
                             {
                             printf("%c",ch);
                       ch=pop();
                       }
                       push(ch);
                       push(a[i]);
                       break;
```

```
                case '-':
                      ch=pop();
                      while(ch!='(')
                            {
                            printf("%c",ch);
                            ch=pop();
                            }
                      push(ch);
                      push(a[i]);
                      break;
                case '*':
                      ch=pop();
                      while( ch=='/' || ch=='*')
                            {
                            printf("%c",ch);
                            ch=pop();
                            }
                      push(ch);
                      push(a[i]);
                      break;
                case '/':
                      ch=pop();
                      while(ch=='*' || ch=='/')
                            {
                            printf("%c",ch);
                            ch=pop();
                            }
                      push(ch);
                      push(a[i]);
                      break;
                default:
                      printf("%c",a[i]);
                }
          i++;
          }
    return 0;
    }

void push(char num)
    {
    stack[++top] = num;
    }
char pop(void)
    {
    char c;
    c=stack[top];
    top=top-1;
    return c;
    }
```