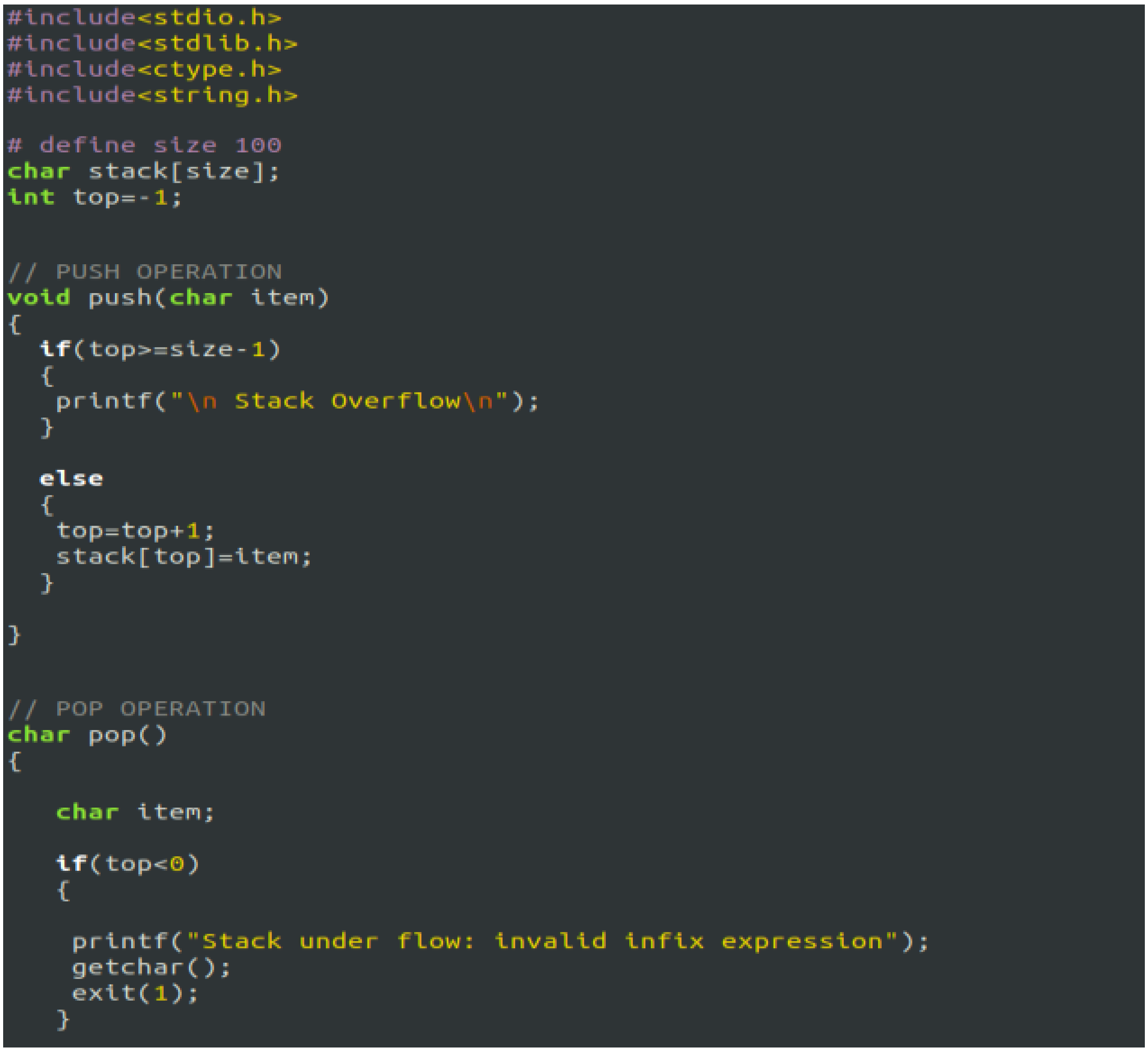
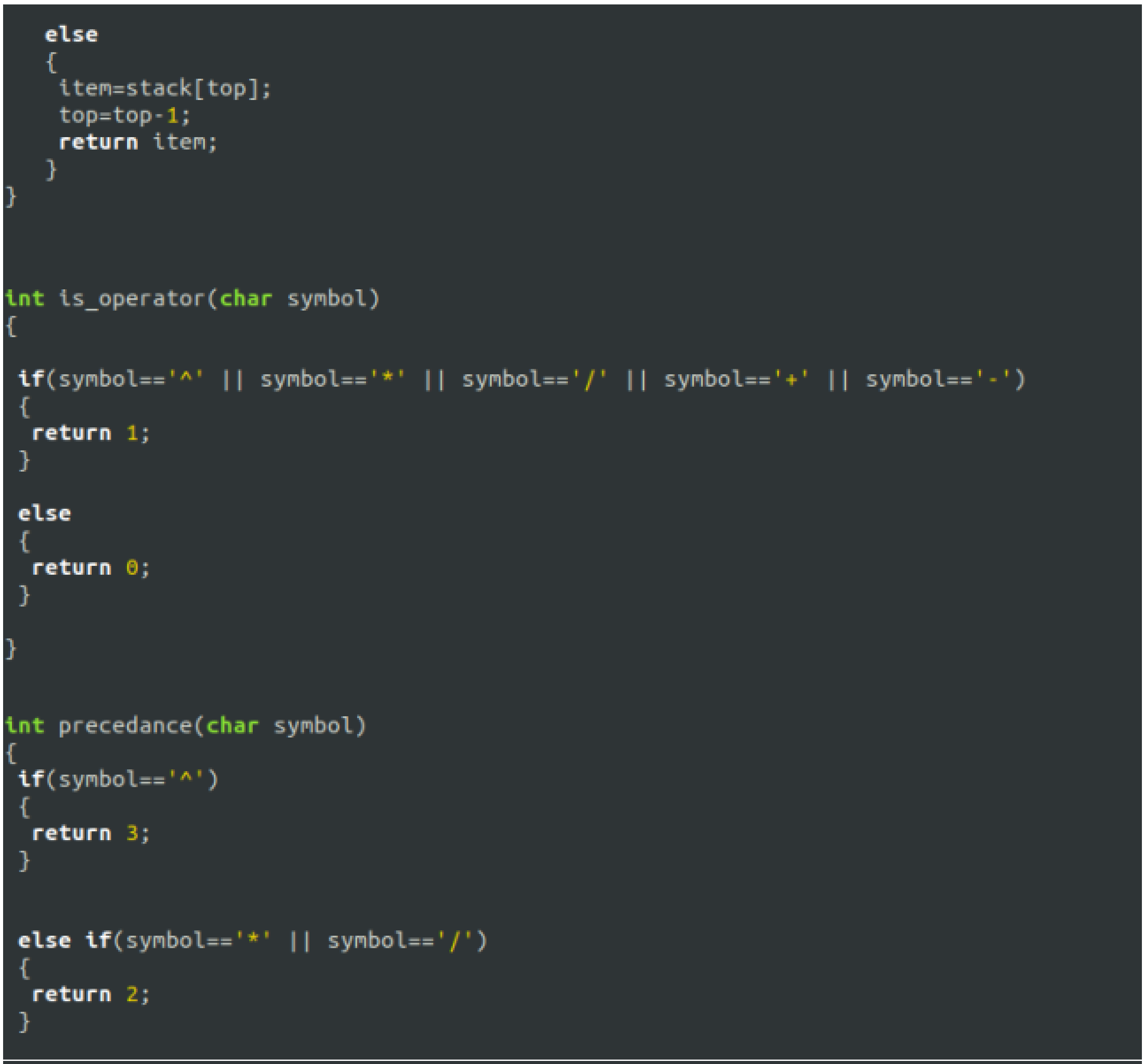
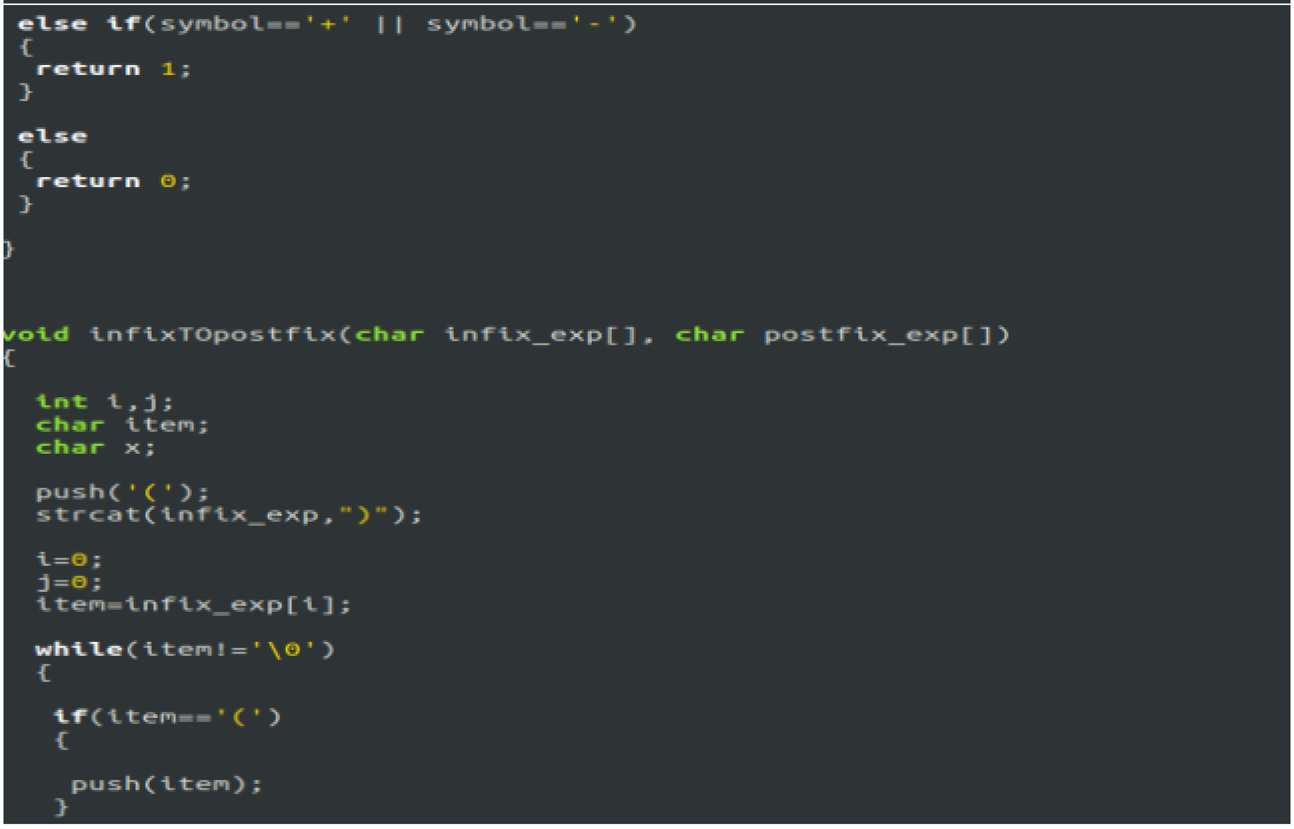
INPUT:



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

# define size 100
char stack[size];
int top=-1;


// PUSH OPERATION
void push(char item)
{
  if(top>=size-1)
  {
   printf("\n Stack Overflow\n");
  }

  else
  {
   top=top+1;
   stack[top]=item;
  }

}


// POP OPERATION
char pop()
{

   char item;

   if(top<0)
   {

    printf("Stack under flow: invalid infix expression");
    getchar();
    exit(1);
   }
```

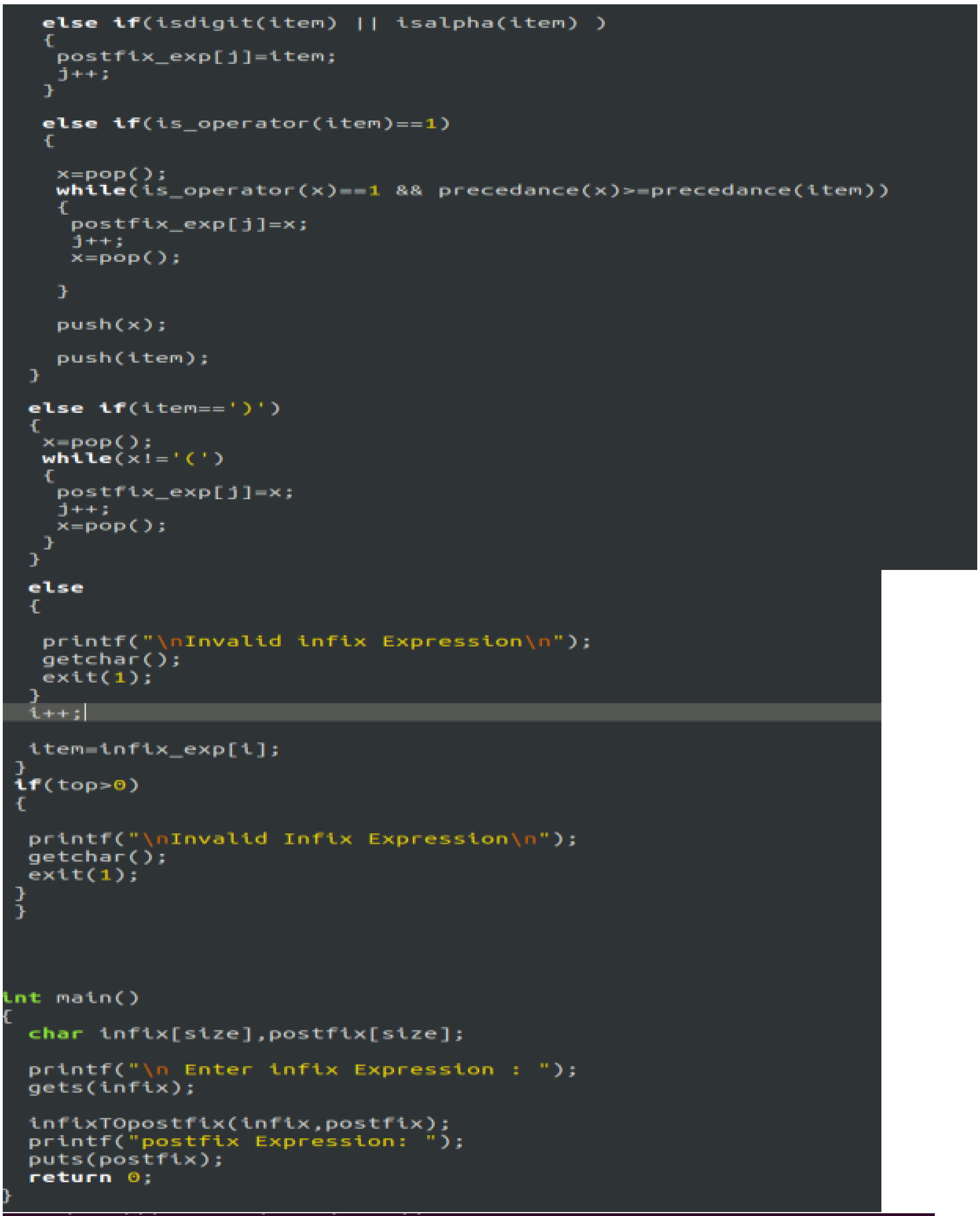


```
   else
   {
    item=stack[top];
    top=top-1;
    return item;
   }
}


int is_operator(char symbol)
{

 if(symbol=='^' || symbol=='*' || symbol=='/' || symbol=='+' || symbol=='-')
 {
  return 1;
 }

 else
 {
  return 0;
 }

}


int precedance(char symbol)
{
 if(symbol=='^')
 {
  return 3;
 }


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

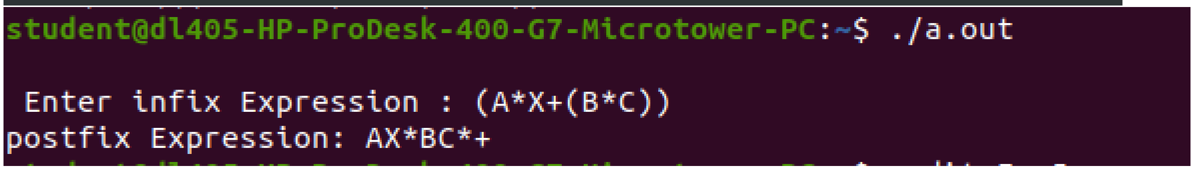


```
 else if(symbol=='+' || symbol=='-')
 {
  return 1;
 }

 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);
   }
```

OUTPUT:



```c
   else if(isdigit(item) || isalpha(item) )
   {
    postfix_exp[j]=item;
    j++;
   }

   else if(is_operator(item)==1)
   {

    x=pop();
    while(is_operator(x)==1 && precedance(x)>=precedance(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);
 }
 }



int main()
{
  char infix[size],postfix[size];

  printf("\n Enter infix Expression : ");
  gets(infix);

  infixTOpostfix(infix,postfix);
  printf("postfix Expression: ");
  puts(postfix);
  return 0;
}
```



```
student@dl405-HP-ProDesk-400-G7-Microtower-PC:~$ ./a.out

 Enter infix Expression : (A*X+(B*C))
postfix Expression: AX*BC*+
```