PRACTICE PROGRAMS

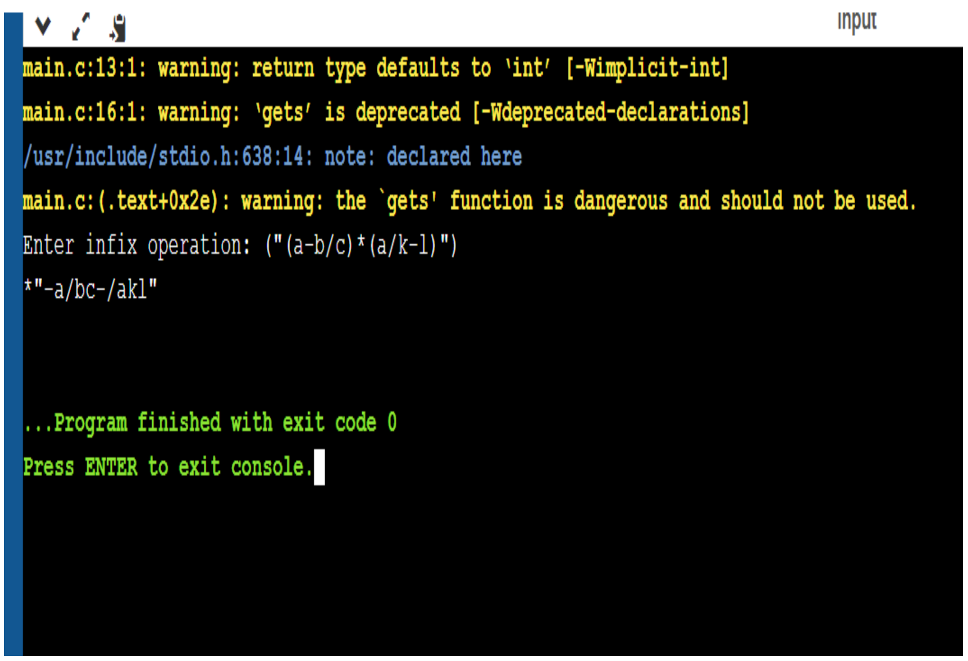
PROGRAM—1

```
# include <stdio.h>

# include <string.h>

# define MAX 20

void infixtoprefix(char infix[20], char prefix[20]);

void reverse(char array[30]);

char pop();

void push(char symbol);

int isOperator(char symbol);

int prcd(char symbol);

int top = -1;

char stack[MAX];


main() {

char infix[20], prefix[20], temp;

printf("Enter infix operation: ");

gets(infix);

infixtoprefix(infix, prefix);

reverse(prefix);

puts((prefix));

}

void infixtoprefix(char infix[20], char prefix[20]) {

int i, j = 0;

char symbol;
```

```
stack[++top] = '#';

reverse(infix);

for (i = 0; i < strlen(infix); i++) {

symbol = infix[i];

if (isOperator(symbol) == 0) {

  prefix[j] = symbol;

  j++;

} else {

  if (symbol == ')') {

    push(symbol);

  } else if (symbol == '(') {

    while (stack[top] != ')') {

      prefix[j] = pop();

      j++;

    }

    pop();

  } else {

    if (prcd(stack[top]) <= prcd(symbol)) {

      push(symbol);

    } else {

      while (prcd(stack[top]) >= prcd(symbol)) {

        prefix[j] = pop();

        j++;

      }

      push(symbol);
```

```
   }


    }

  }


  }


 while (stack[top] != '#') {

   prefix[j] = pop();

   j++;

  }

 prefix[j] = '\0';

}


void reverse(char array[30]) {


int i, j;

char temp[100];

for (i = strlen(array) - 1, j = 0; i + 1 != 0; --i, ++j) {

 temp[j] = array[i];

}

temp[j] = '\0';

strcpy(array, temp);//copying temp array to array


}
```

```
 char pop() {

 char a;

 a = stack[top];

 top--;

 return a;

 }


void push(char symbol) {

top++;

stack[top] = symbol;

}


int prcd(char symbol) {


switch (symbol) {

 case '+':

  case '-':

   return 2;

   break;

 case '*':

  case '/':

   return 4;

   break;

  case '$':
   case '^':
    return 6;
    break;
   case '#':
   case '(':
   case ')':
    return 1;
    break;
  }
}

int isOperator(char symbol) {
switch (symbol) {
case '+':
case '-':
case '*':
case '/':
case '^':
case '$':
case '&':
case '(':
case ')':
  return 1;
  break;
default:
  return 0;


 }

}
```

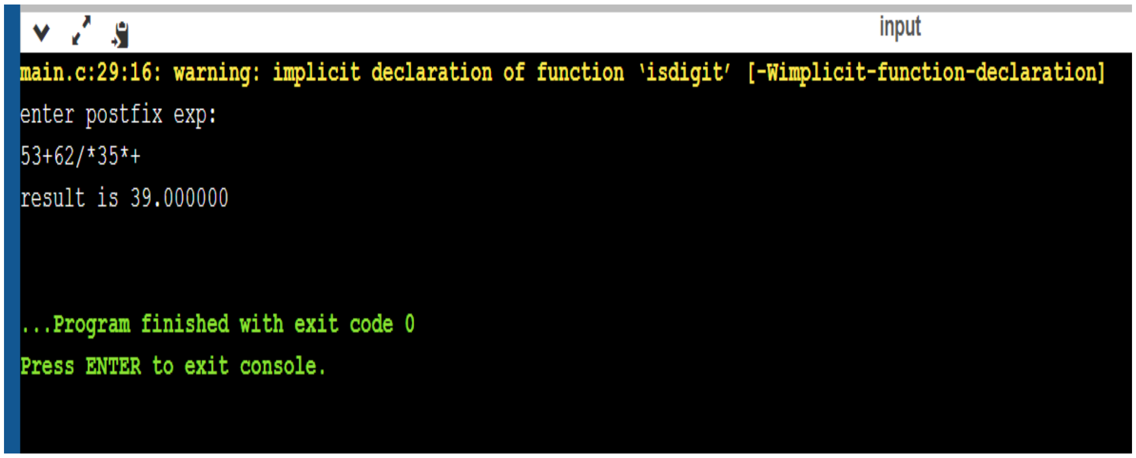


```
main.c:13:1: warning: return type defaults to 'int' [-Wimplicit-int]
main.c:16:1: warning: 'gets' is deprecated [-Wdeprecated-declarations]
/usr/include/stdio.h:638:14: note: declared here
main.c:(.text+0x2e): warning: the `gets' function is dangerous and should not be used.
Enter infix operation: ("(a-b/c)*(a/k-l)")
*"-a/bc-/akl"


...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM—2

```
#include <stdio.h>

#include<math.h>

#include<string.h>

double compute(char symbol, double op1, double op2)

{

   switch(symbol)

   {

      case '+':return op1+op2;

      case '-':return op1-op2;
```

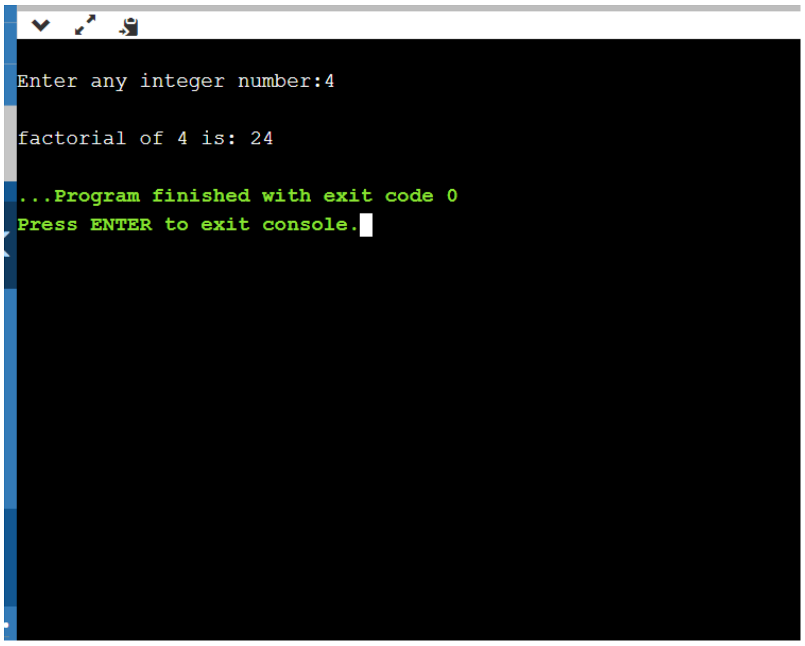
```
    case '*':return op1*op2;

    case '/':return op1/op2;

    case '$':

    case '^':return pow(op1,op2);

}

}

int main()

{

    double s[20];

    double res;

    double op1, op2;

    int top, i;

    char postfix[20], symbol;

    printf("enter postfix exp:\n");

    scanf("%s",postfix);

    top=-1;

    for(i=0;i<strlen(postfix);i++)

    {

        symbol=postfix[i];

        if(isdigit(symbol))

        s[++top]=symbol-'0';

         else

        {

            op2=s[top--];

            op1=s[top--];
            res=compute(symbol,op1,op2);

            s[++top]=res;

        }

    }

    res=s[top--];

    printf("result is %f\n",res);

    return 0;

}
```



```
main.c:29:16: warning: implicit declaration of function 'isdigit' [-Wimplicit-function-declaration]
enter postfix exp:
53+62/*35*+
result is 39.000000


...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM—3

```
#include<stdio.h>

int find_factorial(int);

int main()

{

   int num, fact;

   //Ask user for the input and store it in num

   printf("\nEnter any integer number:");
   scanf("%d",&num);


   //Calling our user defined function

   fact =find_factorial(num);


   //Displaying factorial of input number

   printf("\nfactorial of %d is: %d",num, fact);

   return 0;

}

int find_factorial(int n)

{

   //Factorial of 0 is 1

   if(n==0)

      return(1);


   //Function calling itself: recursion

   return(n*find_factorial(n-1));

}
```

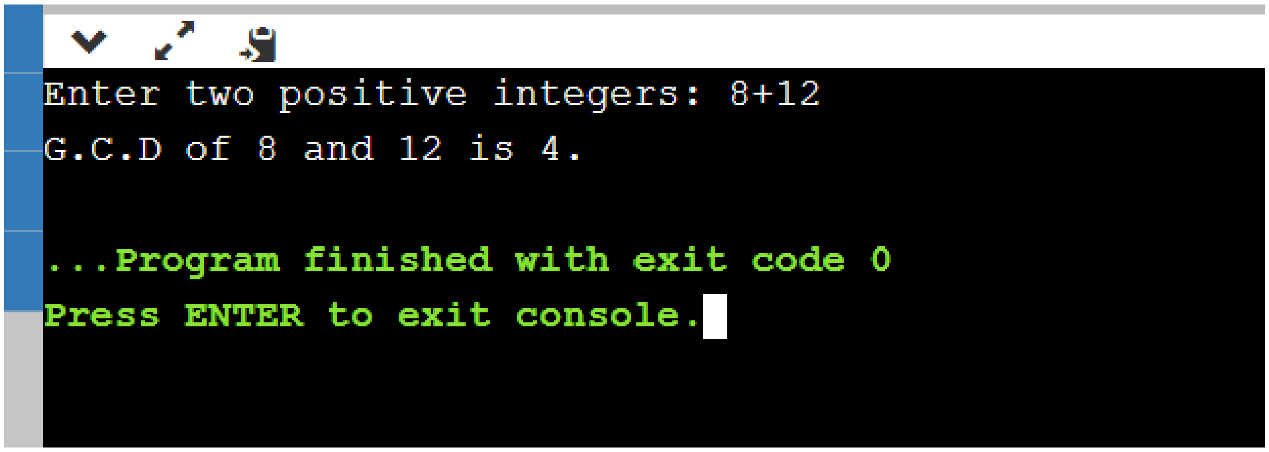


```
Enter any integer number:4

factorial of 4 is: 24

...Program finished with exit code 0
Press ENTER to exit console.
```

PROGRAM—4

```
#include <stdio.h>

int hcf(int n1, int n2);

int main() {

    int n1, n2;

    printf("Enter two positive integers: ");

    scanf("%d %d", &n1, &n2);

    printf("G.C.D of %d and %d is %d.", n1, n2, hcf(n1, n2));

    return 0;

}


int hcf(int n1, int n2) {

    if (n2 != 0)

        return hcf(n2, n1 % n2);

    else
```

```
        return n1;

}
```



```
Enter two positive integers: 8+12
G.C.D of 8 and 12 is 4.

...Program finished with exit code 0
Press ENTER to exit console.
```