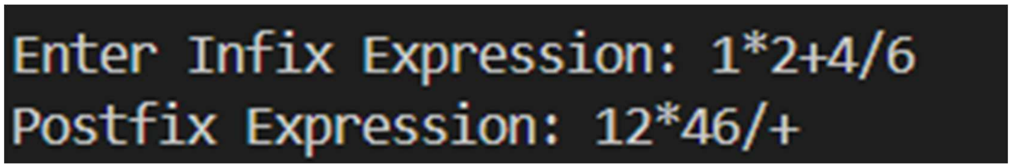
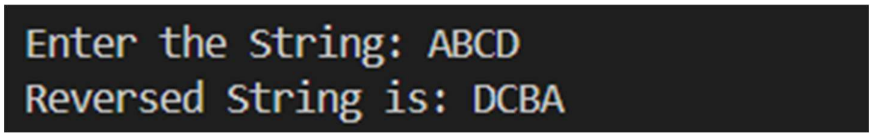
```
#include <stdio.h>

#include <ctype.h>


#define SIZE 100


struct Stack {

   int top;

   char arr[SIZE];

};


void push(struct Stack *s, char ch);

char pop(struct Stack *s);

int precedence(char ch);


/* Main Function */

void main() {

   struct Stack s;

   s.top = -1;


   char infix[SIZE], postfix[SIZE];

   int i, j = 0;


   printf("Enter Infix Expression: ");

   scanf("%s", infix);


   for (i = 0; infix[i] != '\0'; i++) {

      if (isalnum(infix[i])) {

         postfix[j++] = infix[i];
```

```
        }
        else if (infix[i] == '(') {
            push(&s, infix[i]);
        }
        else if (infix[i] == ')') {
            while (s.top != -1 && s.arr[s.top] != '(') {
                postfix[j++] = pop(&s);
            }
            pop(&s); // remove '('
        }
        else {
            while (s.top != -1 && precedence(s.arr[s.top]) >= precedence(infix[i])) {
                postfix[j++] = pop(&s);
            }
            push(&s, infix[i]);
        }
    }

    while (s.top != -1) {
        postfix[j++] = pop(&s);
    }

    postfix[j] = '\0';
    printf("Postfix Expression: %s\n", postfix);
}

/* Push operation */
void push(struct Stack *s, char ch) {
  if (s->top < SIZE - 1) {
    s->arr[++s->top] = ch;
  }
}

/* Pop operation */
char pop(struct Stack *s) {
  if (s->top != -1) {
    return s->arr[s->top--];
  }
  return -1;
}

/* Operator precedence */
int precedence(char ch) {
  if (ch == '+' || ch == '-')
    return 1;
  if (ch == '*' || ch == '/')
    return 2;
  return 0;
}
```



```
Enter Infix Expression: 1*2+4/6
Postfix Expression: 12*46/+
#include <stdio.h>

#include <string.h>


# define SIZE 100

struct Stack {

   int top;

   char arr[SIZE];

};


void push(struct Stack *s, char a);

char pop(struct Stack *s);

int check(struct Stack *s);

void main()

{

   struct Stack s;

   s.top = 0;

   char st[SIZE];

   printf("Enter the String: ");

   gets(st);

   for (int i = 0; st[i] != '\0'; i++){

      push(&s, st[i]);

   }

   for (int i = 0; s.top > 0; i++)

   {

      st[i] = pop(&s);

   }

   printf("Reversed String is: %s", st);

}
```

```
/***

Adds elements to the top of the stack

***/

void push(struct Stack *s, char a)

{

  // Checks if the stack is full

  if (s->top < SIZE)

  {

    s->arr[s->top] = a;

    s->top++;

  }

  else

  {

    printf("STACK IS FULL\n");

  }

}

/***

Remove the last element from the stack

***/

char pop(struct Stack *s)

{

  //Checks if the stack is not empty

  if (s->top > 0)

  {

    s->top--;

    return s->arr[s->top];

  }
  else
  {
    printf("NO ELEMENTS IN STACK\n");
  }
}
```



```
Enter the String: ABCD
Reversed String is: DCBA
```