NAME: TANISHA C A

ROLLNO:230701390

EX-5: Infix to Postfix

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

#define SIZE 20
 int s[SIZE]; int
top = -1; char
input[20]; char
post[20] = " ";
 void push(char
ch); char pop();
int in_precedence(char ch);
int stack_precedence(char ch);
void string_concate(char ch);
 int main()
{     int i;

    // Input the string
printf("Enter the string: ");
scanf("%s", input);
    printf("string = %s\n", input);

    // Initializing the stack with '#' symbol
top = 0;     s[top] = '#';

    // Process the input string character by character
for (i = 0; input[i] != '\0'; i++) {         if
(input[i] >= 'a' && input[i] <= 'z') {
            // If the character of the input string is operand, then
append it to the postfix string
string_concate(input[i]);
        } else {
            // If the character of the input string is
operator/brackets, then check the precedence
            // while the precedence of the input operator < precedence
of stack operator, pop and append to postfix string             while
((in_precedence(input[i])) <
(stack_precedence(s[top]))) {
char Temp = pop();
string_concate(Temp);
            }
            // Push the opening bracket and operator onto the stack if
the precedence of input operator is greater than the stack operator
if (in_precedence(input[i]) != stack_precedence(s[top])) {
push(input[i]);
            } else {
                pop(); // To pop the opening bracket
```

```
            }
        }
    }

    // When we reach the end of the string and if there is anything
left in the stack, pop it and append to the postfix string
while (s[top] != '#') {         string_concate(pop());
    }
    printf("Postfix: %s\n", post);

    return 0;
}
int in_precedence(char ch) {
switch (ch) {         case
'(': return 7;         case
'^': return 6;         case
'*':
        case '/': return 3;
case '+':
        case '-': return 1;
case ')': return 0;
default: return -1;
    }
}
int stack_precedence(char ch) {
switch (ch) {         case '(':
return 0;         case '^':
return 5;         case '*':
        case '/': return 4;
case '+':
        case '-': return 2;
case '#': return -1;
default: return -1;
    }
}  void push(char ch) {
if (top == SIZE - 1) {
printf("Overflow\n");
    } else {
top = top + 1;
s[top] = ch;
    }
}  char pop() {     char c;
if (top == -1) {
printf("Underflow\n");
return -1;     } else {
c = s[top];
        top = top - 1;
    }
return c;
}  void string_concate(char
ch) {     int len =
strlen(post);     post[len] =
ch;
    post[len + 1] = '\0'; // Null-terminate the string
printf("\npost = %s\n", post);
}
```

Output

Enter the expression : (a+b)

Symbols are balanced...! Enter

the expression : ((a+b) Missing

closing symbol...! Enter the

expression : (a+b)) Missing

opening symbol...! Enter the

expression : (a+b] Mismatched

symbol...!