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

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

// Define push operation

void push(char item) {
      if (top >= SIZE-1) {
      printf("Stack Overflow");
      }
      else {
           top = top + 1;
           stack[top] = item;
      }
}

// Define pop operation
char pop() {
     char item;
     if (top<0) {
     printf("Stack underflow: invalid infix expression");
     getchar();
     // underflow may occur for invalid expression
     // where (and) are not matched
     exit(1);
     }
     else {
     item = stack[top];
     top = top - 1;
     return(item);
     }
}


```

```c
int is_operator(char symbol)
{
      if (symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+' || symbol == '-') {
                return 1;
      }
      else {
                return 0;
      }
}


int precedence(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);
             }
             else if (isdigit(item) || isalpha(item) )
             {
                   postfix_exp[j] = item;
                   j++;
             }
             else if (is_operator(item) == 1) {
              x=pop();
              while (is_operator(x) == 1 && precedence(x)>=precedence(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("\n Invalid infix expression.\n");
                    getchar();
                    exit(1);
                   }
                    i++;
                    item = infix_exp[i];
             }
             if(top>0)
             {
             printf("Invalid Expression\n");
             getchar();
             exit(1);
             }
             postfix_exp[j]='\0';
}
int main() {
    char infix[SIZE], postfix[SIZE];
    printf("\n Enter Infix expression: ");
    gets(infix);

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

```
dl07@itadmin:~$ gedit exp6.c
dl07@itadmin:~$ gcc exp6.c
exp6.c: In function ‘main’:
exp6.c:126:5: warning: implicit declaration of function ‘gets’; did you mean ‘fgets’? [-Wimplicit-function-declaration]
  126 |     gets(infix);
      |     ^~~~
      |     fgets
/usr/bin/ld: /tmp/cc7KPbtD.o: in function `main':
exp6.c:(.text+0x3a1): warning: the `gets' function is dangerous and should not be used.
dl07@itadmin:~$ ./a.out

 Enter Infix expression: A+B*C
Postfix Expression: ABC*+
```