Infix to postfix

Aim:

The aim of the provided code is to convert an infix expression to a postfix expression in c

Algorithm:

input: Read the infix expression from the user.

Process the Expression:

Initialize an empty stack.

Loop through each character of the expression.

If it's an operand (alphabet), output it directly.

If it's an operator:

While the stack is not empty and the precedence of the current operator is less than or equal to the precedence of the operator at the top of the stack, pop and output the top operator.

Push the current operator onto the stack.

If it's an opening parenthesis, push it onto the stack.

If it's a closing parenthesis:

Pop and output operators from the stack until an opening parenthesis is encountered.

Discard the opening parenthesis.

Ignore spaces.

If it's any other character, print "Invalid expression" and terminate.

Output: After processing all characters, pop and output any remaining operators from the stack.

Program:

```
#include <stdio.h>
char str[25];
int stack[25];
int top=-1;

int priority(char sym)
{
   if(sym=='*'||sym=='/'||sym=='%')
   return 2;
   else if(sym=='+'||sym=='-')
   return 1;
   else
   return 0;
}

void push(int num)
{
   top++;
   stack[top]=num;
}

void pop()
```

```
{
    if(top<0)
    {
        printf("\n---Stack Underflow---\n");
    }
    else
    {
        top--;
    }
}

char peek()
{
    if(top<0)
    return '\0';
    else
    return stack[top];
}

void display()
{
    if(top<0)
    printf("\nThe stack is empty\n");

    for(int i=top;i>=0;i--)
    {
        printf(" | %d |\n",stack[i]);
    }
}

void check_precedence(char in_exp)
{
    char in_stack=peek();

    if(priority(in_stack) >= priority(in_exp))
    {
        printf("%c",in_stack);
        pop();
        check_precedence(in_exp);
    }

    else
    {
        push(in_exp);
    }
}

int main()
{
    printf("Enter expression containing lowercase alphabets and operators (+,-,*,/,%)\n");
    scanf("%[^\n]s",str);
    for(int i=0;str[i]!='\0';i++)
    {
        if((str[i]>='a'&& str[i]<='z')||(str[i]>='A' && str[i]<='Z'))
        {
            printf("%c",str[i]);
        }

        else if(str[i]=='+'||str[i]=='-'||str[i]=='*'||str[i]=='/')
        {
            check_precedence(str[i]);
        }

        else if(str[i]==' ')
        continue;

        else if(str[i]=='(')
        push(str[i]);

        else if(str[i]==')')
        {
            while(peek()!='(')
            {
                printf("%c",peek());
                pop();
            }
            pop();
        }
        else
        {
            printf("Invalid expression ");
            break;
        }
    }

    while(top!=-1)
    {
        printf("%c",peek());
        pop();
    }
    return 0;
}
```
Output:

```
Enter expression containing lowercase alphabets and operators (+,-,*,/,%)
a b +
ab+
```
Result:

The output is verified successfully for the above program.