```c
/*
Program 5a: Design, Develop and Implement a Program in C for the
Evaluation of Suffix expression with single digit operands and
operators: +, -, *, /, %,^
*/

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

//Here evaluate is a user defined function.

double evaluate(char symbol, double op1, double op2)
{
    switch(symbol)                          //Various operations
    {
        case '+': return (op1+op2);
        case '-': return (op1-op2);
        case '*': return (op1*op2);
        case '/': return (op1/op2);
        case '$':
        case '^': return pow(op1,op2); //same operation even for upward arrow symbol
    }
}

/*
Here $ and ^ have the same meaning and does the same function of calcaulating
the Power of the operands.
i.e Operand 1 to the power of operand 2.
This is why we have not mentioned return value for $ in switch sttement above
*/

void main()
{
    double A[20];               //Name of the stack.
    double result;             // Stores the evaluated result
    double op1, op2;          // Stores the two operators
    int i, top;
    char postfix[20];          /* Stores the postfix expression.
                                  Here Postfix expression is stored as a string */

    char symbol;              //Stores the symbols - +,-,*,/,%,^

    printf("Enter the postfix expression:\n");
    scanf("%s",postfix);

    top=-1;                           //Refers to top of the stack or empty stack

    for(i=0;i<strlen(postfix);i++)     //Till the entire length of Stack
    {
        symbol=postfix[i];

/*
Checking for a digit. Since postfix is considered as a string,
we are going to subtract ASCII value of 0.(ASCII value of 0 is 48)

if the scanned symbol is an operand, we push symbol directly onto stack.
*/
        if(isdigit(symbol))
            A[++top]=symbol-'0';      //If only operand is encountered, add to Stack

/*
if the scanned symbol is an operator, we do evaluation first and
then the result of the evaluation is put back in the stack
*/
```

```
67          else
68          {
69              op2=A[top--];               //If operator is encountered, do evaluation.
70              op1=A[top--];
71              result=evaluate(symbol,op1,op2);
72              A[++top]=result;            //Push the evaluated result also onto stack.
73          }
74      }
75
76      result=A[top--];                   //Get the final result that is there in stack.
77
78      printf("The value is : %f",result); //Prints the final result.
79
80  }
```

Output:

```
Enter the postfix expression:
456*+
The value is : 34.000000


Enter the postfix expression:
231*+9-
The value is : -4.000000


Enter the postfix expression:
623+-382/+*2^3+
The value is : 52.000000
```