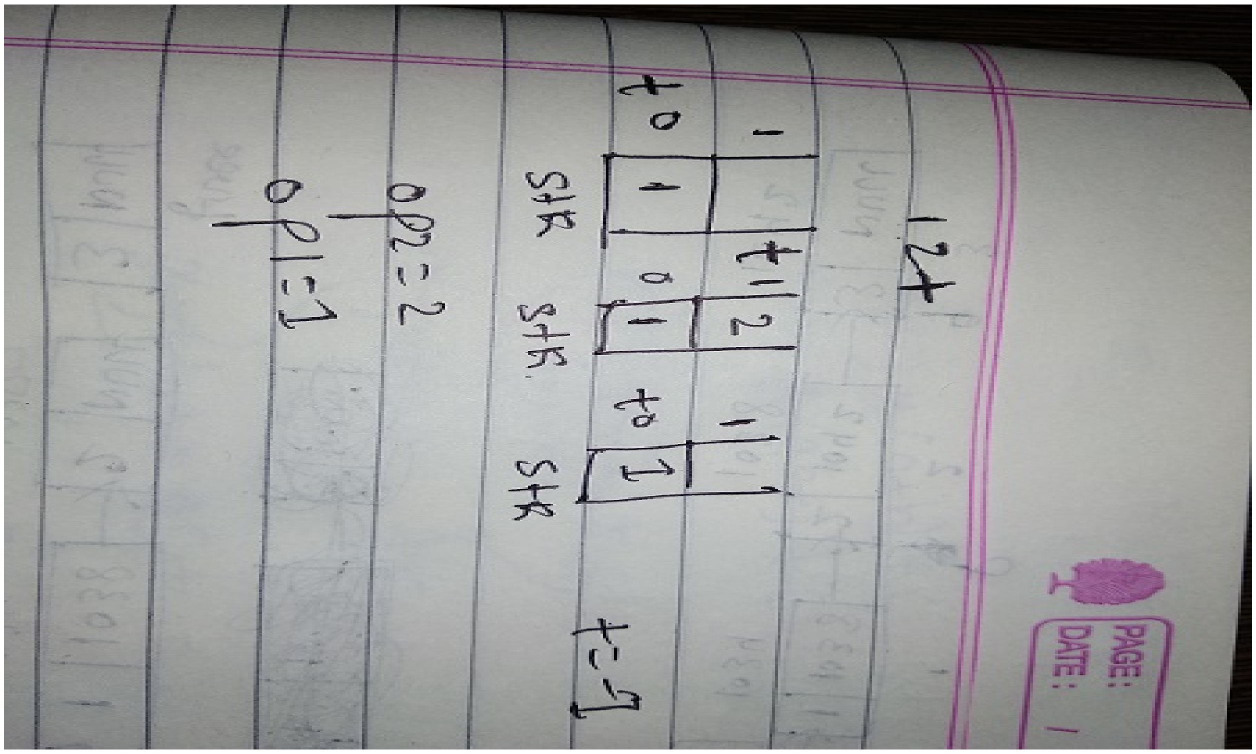
Evaluation of expressions



```
#include<stdio.h>

#include<math.h>

#define size 30

main()

{

        char expr[size];

        double result;

        printf("enter the postfix expression");

        scanf("%s", expr);   expr=12+

        result=eval(expr);

        printf("the result is %lf", result);

}
```

```
double eval(char exp[])
{                                               exp=12+
          double op1,op2,stk[size];
          char symb;
          int i,t;
          i=0;
          t=-1;
          do
          {
                    symb=exp[i];   exp=1 2 +
                    If(symb> ‘0’ && symb<’9’)
                    {
                                        stk[++t]=symb-‘0’;     asci value= 50   ascii value of 0 is 48
                                        else
                                        {
                                                  Op2=stk[t--];
                                                  Op1=stk[t--];
                                                  switch(symb)
                                                  {
                                                            case ‘+’ : stk[++t]=op1+op2;
                                                                        break;
                                                            case ‘-‘ : stk[++t]=op1-op2;
                                                                       break;
                                                            case ‘*’ : stk[++t]=op1*op2;
                                        break;
                              case ‘/’:  stk[++t]=op1/op2;
                                        break;
                              case ‘$’ : b=pow(op1,op2);
                                        stk[++t]=b;
                        }
                }
                        i++;
                }while(exp[i]!=’\o’);
                return (stk[t]);
        }
```

Infix to postfix conversion

Scan the character one at a time

i) If character is (then push it on stack
ii) If character is operand then output it in postfix string
iii) If character is operator and stack is empty push it on stack
iv) If precedence of operator on top of stack is higher or equal then the input operator then operator on top of stack needs to be outputted on postfix string and input operator is pushed onto stack. If not just push input character into stack.
v) If input character is) then pop elements of stack till we reach to corresponding (on stack.
vi) Pop all the element of stack into postfix string once we reach end of input string.

i)

x+(y*z)

input string   stack   output string

x	empty	x
+	+	x
(	+, (	x
Y	+,(	xy
*	+,(,*	xy
Z	+,(,*	xyz
)	+	xyz*
		xyz*+

ii) x*y+z

input string	stack	output string
x	empty	x
*	*	x
Y	*	xy
+	+	xy*
Z	+	xy*z+

iii) (A+(B-C)*D)

input string	stack	output string
(	(	
A	(	A
+	(,+	A
(	(,+,(	A
B	(,+,(	AB
-	(,+,(,-	AB

C	(,+,(,-	ABC
)	(,+	ABC-
*	(,+,*	ABC-
D	(,+,*	ABC-D
)		ABC-D*+

((a+(b-c)*d)+e+f)