OPERATOR PRECEDENCE PARSER

NADISH SHAJAHAN
S7 CSE B
21

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

char stack[100] = {'$'},ex[100];
int len = 0,top = 0,ip = 0;
int pt[9][9] = {{1,1,-1,-1,-1,-1,1,1,1},{1,1,-1,-1,-1,-1,-1,1,1},{1,1,1,1,-1,-1,-1,1,1},
                {1,1,1,1,-1,-1,-1,1,1},{1,1,1,1,-1,-1,-1,1,1},{1,1,1,1,1,9,9,1,1},
                {-1,-1,-1,-1,-1,-1,-1,0,1},{1,1,1,1,1,10,10,1,1},{-1,-1,-1,-1,-1,-1,-1,10,10}};

int val(char ch);
void push(char item);
char pop();
main()
{
        char item,a,b;
        printf("Enter the string : \n");
        gets(ex);
        len=strlen(ex);
        ex[len] = '$';
        ex[len+1] = '\0';
        if(ex[ip] == '+' || ex[ip] == '-' || ex[ip] == '/' || ex[ip] == '*' || ex[ip] == '^')
                {
                        printf("Expression is invalid!!\n");
                        exit(0);
                }
        while(ex[ip]!='\0')
        {
                if(ex[ip] == '+' || ex[ip] == '-' || ex[ip] == '/' || ex[ip] == '*' || ex[ip] == '^' || ex[ip] == '('
                        {
                                if(ex[ip+1] == '+' || ex[ip+1] == '-' || ex[ip+1] == '/' || ex[ip+1] == '*' || ex[ip+1] == '^'||
ex[ip+1] == '$' || ex[ip+1] == ')')
                                {
                                        printf("Expression is invalid!!\n");
                                        exit(0);
                                }
                        }
                a=val(stack[top]);
                b=val(ex[ip]);
                if(ex[ip]=='$' && stack[top]=='$')
                {
                        printf("Expression is valid.\n");
                        exit(0);
                }
                else if(a==-1 || b==-1)
                {
                        printf("Expression is invalid!!\n");
                        exit(0);
                }
                else
                {
                        if(pt[a][b]==-1 || pt[a][b]==0)
```

```
                {
                        push(ex[ip]);
                        ip++;
                }
                else if(pt[a][b]==1)
                {
                do
                {
                        item = pop();
                        a = val(stack[top]);
                        b = val(item);
                }while((pt[a][b])==1);
                }
                else
                {
                        printf("Expression is invalid!!\n");
                        exit(0);
                }
        }
    }
}

int val(char ch)
{
    if(ch=='+')
        return 0;
    else if(ch=='-')
        return 1;
    else if(ch=='*')
        return 2;
    else if(ch=='/')
        return 3;
    else if(ch=='^')
        return 4;
    else if((ch>='a' && ch<='z') || (ch>='A' && ch<='Z'))
        return 5;
    else if(ch=='(')
        return 6;
    else if(ch==')')
        return 7;
    else if(ch=='$')
        return 8;
    else
        return -1;
}

char pop()
{
    char item;
    item = stack[top];
    top--;
    return item;
}

void push(char item)
{
    top++;
    stack[top] = item;
```

```
}
```

OUTPUT

```
57613@user:/mnt/57613/s7comp/op_pre_parser$ gcc operator.c
57613@user:/mnt/57613/s7comp/op_pre_parser$ ./a.out
Enter the string :
a*k
Expression is valid.
57613@user:/mnt/57613/s7comp/op_pre_parser$ ./a.out
Enter the string :
(a+b)*f
Expression is valid.
```