NAME: BALDHA SMIT
BRANCH: C.E.
DIVISION: A1
SEMESTER: 03
SUBJECT: Data structure
PRACTICAL: 06

AIM: Implement a program to convert infix notation to postfix notation using stack.

Code:

```
#include<stdio.h>
#include<stdlib.h>
#include<ctype.h>
#include<string.h>
#include<time.h>
#define SIZE 100
char stack[SIZE];
int top = -1;

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

char pop()
{
	char item ;
	if(top <0)
```

```
        {
                printf("stack under flow: invalid infix expression");
                getchar();
                exit(1);
        }
        else
        {
                item = stack[top];
                top = top-1;
                return(item);
        }
}


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);
}

int rank(char str[])
{
   int r=0, i ;
   for(i=0 ; str[i] != '\0' ; i++)
   {
```

```
                if(isalpha(str[i]) || isdigit(str[i]) )
                r = r + 1 ;

                else if(is_operator(str[i]))
                r = r - 1 ;
    }
    printf("\nRank of %s = %d",str,r);
    return(r) ;
}

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("\nInvalid infix Expression.\n");
                        getchar();
                        exit(1);
                }
                i++;
                item = infix_exp[i];
        }
        if(top>0)
        {
                printf("\nInvalid infix Expression.\n");
                getchar();
                exit(1);
        }
        postfix_exp[j] = '\0';
}




int main()
{
   printf("enrollment no:190130107007\npractical no:6\t");
   time_t curtime;
   time(&curtime);
   printf("Current time = %s\n\n", ctime(&curtime));


        char infix[SIZE], postfix[SIZE];
        printf("ASSUMPTION: The infix expression contains single letter variables and single
digit constants only.\n");
        printf("\nEnter Infix expression : ");
        gets(infix);
```

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

        if(rank(postfix) !=1)
                printf("\nThe expression is Invalid");

        else
          printf("\nThe expression is Valid");

        return 0;
}
```

OUTPUT:

1)

```
enrollment no:190130107007
practical no:6  Current time = Tue Aug 04 17:36:13 2020


ASSUMPTION: The infix expression contains single letter variables and single digit constants only.

Enter Infix expression : a+b-c*d/e+f*h
Postfix Expression: ab+cd*e/-fh*+

Rank of ab+cd*e/-fh*+ = 1
The expression is Valid
--------------------------------
Process exited after 17.32 seconds with return value 0
Press any key to continue . . .
```

2)

```
enrollment no:190130107007
practical no:6  Current time = Tue Aug 04 17:41:30 2020


ASSUMPTION: The infix expression contains single letter variables and single digit constants only.

Enter Infix expression : a+(b*d-e)*f+g
Postfix Expression: abd*e-f*+g+

Rank of abd*e-f*+g+ = 1
The expression is Valid
--------------------------------
Process exited after 53.68 seconds with return value 0
Press any key to continue . . .
```