30/9/20

IBM19CS090

Mohammad Ibrahim Rahil.

Lab Program - 2.

WAP to convert a given valid parenthesised infix expression to postfix expression. The expression consists of single operands and the binary operators + (plus), - (minus), * (multiply) and / (divide).

Code:

```
#include <stdio.h>
#include <string.h>
#include <process.h>
int F (char symbol)
{
  switch (symbol)
  {
    case '+':
    case '-': return 2;
    case '*':
    case '/': return 4;
    case '^':
    case '$': return 5;
    case '(': return 0;
    case '#': return -1;
    default : return 8;
  }
}
int G (char symbol)
{
  switch (symbol)
  {
    case '+':
    case '-': return 1;
    case '*':
    case '/': return 3;
```

```
        case '^':
        case '$': return 6;
        case '(': return 9;
        case ')': return 0;
        default : return 7;
    }
}
void infix_postfix (char infix[], char postfix[])
{
  int top, i, j;
  char s[30], symbol;
  top = -1;
  s[++top] = '#';
  j = 0;
  for (i = 0; i < strlen (infix); i++)
  {
    symbol = infix[i];
    while (F(s[top]) > G(symbol))
    {
      postfix[j] = s[top--];
      j++;
    }
  }
  if (F(s[top]) != G(symbol))
    s[++top] = symbol;
  else
    top--;
  }
  while (s[top] != '#')
  {
    postfix[j++] = s[top--];
  }
```

```
    postfix[j] = '\0';
}
void main()
{
    char infix[20];
    char postfix[20];
    printf("Enter the valid infix expression \n");
    scanf("%s", infix);
    infix_postfix(infix, postfix);
    printf("The postfix exp is \n");
    printf("%s\n", postfix);
    getch();
}
```

Postfix.c

Saved

```c
#include<stdio.h>
#include<string.h>
int f(char sym)
{
  switch(sym)
  {
    case '+':
    case '-':return 2;
    case '*':
    case '/':return 4;
    case '^':
    case '$':return 5;
    case '(':return 0;
    case '#':return -1;
    default:return 8;
  }
}
int g(char sym)
{
  switch(sym)
  {
    case '+':
    case '-':return 1;
    case '*':
    case '/':return 3;
    case '^':
    case '$':return 6;
    case '(':return 9;
    case ')':return 0;
    default:return 7;
  }
}
void infix_postfix(char infix[],char postfix[])
{
  int top,i,j;
  char s[30],sym;
  top=-1;
  s[++top]='#';
  j=0;
  for(i=0;i<strlen(infix);i++)
  {
    sym=infix[i];
    while(f(s[top])>g(sym))
    {
      postfix[j]=s[top--];
      j++;
    }
    if(f(s[top])!=g(sym))
    s[++top]=sym;
    else
    top--;
  }
  while(s[top]!='#')
  postfix[j++]=s[top--];
  postfix[j]='\0';
}
void main()
{
  char infix[20];
  char postfix[20];
  printf("Enter the Valid Infix Expression\n");
  scanf("%s",infix);
  infix_postfix(infix,postfix);
  printf("The Postfix Expression is ");
  printf("\n%s\n",postfix);
}
```