EX NO 20 :

Write a C program to implement operator precedence parsing.

PROGRAM :

#include<stdio.h>

#include<string.h>

char \*input; int i=0;

char lasthandle[6],stack[50],handles[][5]={")E(","E\*E","E+E","i","E^E"};

//(E) becomes )E( when pushed to stack

int top=0,l;

char prec[9][9]={

/\*input\*/

/\*stack + - \* / ^ i ( ) $ \*/

/\* + \*/ '>', '>','<','<','<','<','<','>','>',

/\* - \*/ '>', '>','<','<','<','<','<','>','>',

/\* \* \*/ '>', '>','>','>','<','<','<','>','>',

/\* / \*/ '>', '>','>','>','<','<','<','>','>',

/\* ^ \*/ '>', '>','>','>','<','<','<','>','>',

/\* i \*/ '>', '>','>','>','>','e','e','>','>',

/\* ( \*/ '<', '<','<','<','<','<','<','>','e',

/\* ) \*/ '>', '>','>','>','>','e','e','>','>',

/\* $ \*/ '<', '<','<','<','<','<','<','<','>',

};

int getindex(char c)

{

switch(c)

{

case '+':return 0;

case '-':return 1;

case '\*':return 2;

case '/':return 3;

case '^':return 4;

case 'i':return 5;

case '(':return 6;

case ')':return 7;

case '$':return 8;

}

}

int shift()

{

stack[++top]=\*(input+i++); stack[top+1]='\0';

}

int reduce()

{

int i,len,found,t; for(i=0;i<5;i++)//selecting handles

{

len=strlen(handles[i]); if(stack[top]==handles[i][0]&&top+1>=len)

{

found=1; for(t=0;t<len;t++)

{

if(stack[top-t]!=handles[i][t])

{

found=0; break;

}

}

if(found==1)

{

stack[top-t+1]='E'; top=top-t+1;

strcpy(lasthandle,handles[i]);

stack[top+1]='\0';

return 1;//successful reduction

}

}

}

return 0;

}

void dispstack()

{

int j; for(j=0;j<=top;j++)

printf("%c",stack[j]);

}

void dispinput()

{

int j; for(j=i;j<l;j++)

printf("%c",\*(input+j));

}

int main()

{

int j;

input=(char\*)malloc(50\*sizeof(char));

printf("\nEnter the string\n");

scanf("%s",input); input=strcat(input,"$"); l=strlen(input);

strcpy(stack,"$"); printf("\nSTACK\tINPUT\tACTION"); while(i<=l)

{

shift(); printf("\n"); dispstack(); printf("\t"); dispinput(); printf("\tShift");

if(prec[getindex(stack[top])][getindex(input[i])]=='>')

{

while(reduce())

{

printf("\n");

dispstack(); printf("\t"); dispinput();

printf("\tReduced: E->%s",lasthandle);

}

}

}

if(strcmp(stack,"$E$")==0) printf("\nAccepted;");

else

printf("\nNot Accepted;");

}

OUTPUT :

