CODE OPTIMISATION

Mohammed Farhan
S7CSE-B18

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

char S[50],expr[50],expr1[50],post[50],o;
int top,max,n,num=0;
char item,x,y,j,res='0';

void push(char);
char pop(void);
int ISP(char);
int ICP(char);
int OP(char,int,int);

struct inter
{
  char operator, arg1,arg2,result;
}INTER[10];

void main()
{
  int i,j,k=0,flag=0,s,limit;
  top=-1,max=50;
  printf("Enter number of expression\n");
  scanf("%d",&limit);
  for(s=0;s<limit;s++)
 {

  printf("\n Enter the Infix expression : ");
  scanf("%s",expr);

  for(i=0;expr[i]!='\0';i++)
    {
      if(expr[i]=='=')
      break;
    }

  if(expr[i]!='=')
    strcpy(expr1,expr);
  else
    {
      flag=1;
      for(j=i+1;expr[j]!='\0';j++,k++)
      expr1[k]=expr[j];
      expr1[k]='\0';
    }

  for(i=0;expr1[i]!='\0';i++);
  expr1[i]=')';
  expr1[i+1]='\0';
```

```
  push('(');
  i=0,j=0;

  while(top>-1)
    {
      x=pop();
      item=expr1[i];

      if(isalpha(item))
      {
        push(x);
        post[j]=item;
        i++,j++;
      }
      else if(item==')')
      while(x!='(')
        {
          post[j]=x;
          i++,j++;
          x=pop();
        }
      else if((item=='+')||(item=='-')||(item=='*')||(item=='/')||(item=='^')||
(item=='('))
      if(ISP(x)>=ICP(item))
        {
          while(ISP(x)>=ICP(item))
            {
            post[j]=x;
            j++;
            x=pop();
            }
          push(x);
          push(item);
          i++;
        }
      else
        {
          push(x);
          push(item);
          i++;
        }
    }
  post[j]='\0';
  printf("\n The Postfix expression is :- ");
  if(flag==1)
    printf("%c%s=",expr[0],post);
  else
    printf("%s",post);

  top=-1;
  n=0,i=0;

  for(i=0;post[i]!='\0';i++);

  if(flag==1)
    {
      push(expr[0]);
      post[i]='=';
```

```
      post[i+1]='#';
      post[i+2]='\0';
    }
  else
    {
      post[i]='#';
      post[i+1]='\0';
    }

  i=0;

  while(post[i]!='#')
    {
      if(isalpha(post[i]))
      {
        push(post[i]);
        i++;
      }
      if((post[i]=='+')||(post[i]=='-')||(post[i]=='*')||(post[i]=='/')||
(post[i]=='^')||post[i]=='=')
      {
        if(post[i]=='=')
          {
            x=pop();
            y=pop();
            INTER[num].operator=post[i];
            INTER[num].arg1=x;
            INTER[num].arg2=' ';
            INTER[num].result=y;
            num++;
            i++;
          }
        else
          {
            x=pop();
            y=pop();
            INTER[num].operator=post[i];
            INTER[num].arg1=y;
            INTER[num].arg2=x;
            INTER[num].result=res;
            res++;
            push(INTER[num].result);
            num++;
            i++;
          }
      }
    }


  printf("\n Output :\n\n Op\tArg1\tArg2\tResult\n --\t----\t----\t------\n");
  for(i=0;i<num;i++)
    {
      printf(" %c",INTER[i].operator);
      if(isdigit(INTER[i].arg1))
      printf("\tt%c",INTER[i].arg1);
      else
      printf("\t%c",INTER[i].arg1);
      if(isdigit(INTER[i].arg2))
```

```
      printf("\tt%c",INTER[i].arg2);
      else
      printf("\t%c",INTER[i].arg2);
      if(isdigit(INTER[i].result))
      printf("\tt%c\n",INTER[i].result);
      else
      printf("\t%c\n",INTER[i].result);
     }
  printf("\n");



  for(i=0;i<num;i++)
      for(j=i+1;j<num;j++)
      {
        if(INTER[i].operator==INTER[j].operator)
          {
            if(INTER[i].arg1==INTER[j].arg1)
            {
              if(INTER[i].arg2==INTER[j].arg2)
                {
                  item=INTER[j].result;
                  for(k=j+1;k<num;k++)
                  {
                    if(INTER[k].arg1==item)
                      INTER[k].arg1=INTER[i].result;
                    if(INTER[k].arg2==item)
                      INTER[k].arg2=INTER[i].result;
                  }
                  for(k=j;k<num-1;k++)
                  {
                    INTER[k].operator=INTER[k+1].operator;
                    INTER[k].arg1=INTER[k+1].arg1;
                    INTER[k].arg2=INTER[k+1].arg2;
                    INTER[k].result=INTER[k+1].result;
                  }
                  num--;
                }
            }
          }
      }


for(i=0;i<num;i++)
      for(j=i+1;j<num;j++)
      {
        if(INTER[i].operator==INTER[j].operator)
          {
            if(INTER[i].arg1==INTER[j].arg1)
            {
              if(INTER[i].arg2==INTER[j].arg2)
                {
                  item=INTER[j].result;
                  for(k=j+1;k<num;k++)
                  {
                    if(INTER[k].arg1==item)
                      INTER[k].arg1=INTER[i].result;
                    if(INTER[k].arg2==item)
```

```
                    INTER[k].arg2=INTER[i].result;
                  }
                  for(k=j;k<num-1;k++)
                  {
                    INTER[k].operator=INTER[k+1].operator;
                    INTER[k].arg1=INTER[k+1].arg1;
                    INTER[k].arg2=INTER[k+1].arg2;
                    INTER[k].result=INTER[k+1].result;
                  }
                  num--;
                }
            }
          }
      }

  printf("\n Optimized Output :\n\n Op\tArg1\tArg2\tResult\n
--\t----\t----\t------\n");
  for(i=0;i<num;i++)
    {
      printf(" %c",INTER[i].operator);
      if(isdigit(INTER[i].arg1))
      printf("\tt%c",INTER[i].arg1);
      else
      printf("\t%c",INTER[i].arg1);
      if(isdigit(INTER[i].arg2))
      printf("\tt%c",INTER[i].arg2);
      else
      printf("\t%c",INTER[i].arg2);
      if(isdigit(INTER[i].result))
      printf("\tt%c\n",INTER[i].result);
      else
      printf("\t%c\n",INTER[i].result);
    }
  printf("\n");
}

}

int ISP(char expr1)
{
  if(expr1=='^')
    return(3);
  if((expr1=='*')||(expr1=='/'))
    return(2);
  if((expr1=='+')||(expr1=='-'))
    return(1);
  if(expr1=='(')
    return(0);
}

int ICP(char expr1)
{
  if(expr1=='^')
    return(4);
  if((expr1=='*')||(expr1=='/'))
    return(2);
  if((expr1=='+')||(expr1=='-'))
    return(1);
```

```
  if(expr1=='(')
    return(4);
}

void push(char expr1)
{
  top++;
  S[top]=expr1;
}

char pop(void)
{
  char expr1;
  expr1=S[top];
  top--;
  return(expr1);
}
```

OUTPUT

```
42813@user:/mnt/42813/compiler/intermediate$ ./a.out
Enter number of expression
2

 Enter the Infix expression : a=b+c

 The Postfix expression is :- abc+=
 Output :

 Op   Arg1  Arg2  Result
 --   ----  ----  ------
 +    b     c     t0
 =    t0          a


 Optimized Output :

 Op   Arg1  Arg2  Result
 --   ----  ----  ------
 +    b     c     t0
 =    t0          a


 Enter the Infix expression : d=(b+c)-(b+c)*t

 The Postfix expression is :- dbcbc++bc+t*-=
 Output :

 Op   Arg1  Arg2  Result
 --   ----  ----  ------
 +    b     c     t0
 =    t0          a
 +    b     c     t1
 +    c     t1    t2
 +    b     c     t3
 *    t3    t     t4
 -    t2    t4    t5
 =    t5          b


 Optimized Output :

 Op   Arg1  Arg2  Result
 --   ----  ----  ------
 +    b     c     t0
 =    t0          a
 +    c     t0    t2
 *    t0    t     t4
 -    t2    t4    t5
 =    t5          b
```