AIM: TO IMPLEMENT THE INTERMEDIATE CODE GENERATION FOR THE GIVEN EXPRESSION IN C.

ALGORITHM:

- RSTACK – stack of registers, R0,...,R(r-1)
- TSTACK – stack of temporaries, T0,T1,...
- A call to Gencode(n) generates code to evaluate a tree T, rooted at node n, into the register top(RSTACK) ,and the rest of RSTACK remains in the same state as the one.

before the call

- A swap of the top two registers of RSTACK is needed at some points in the algorithm to ensure that a node is evaluated into the same register as its left child.

Procedure gencode(n);

{ /* case 0 */

If

n is a leaf representing

operand N and is the

leftmost child of its parent

then

print(LOAD N, top(RSTACK))

```
        }

/* case 1 */

else if

         n is an interior node with operator
         OP, left child n1, and right child n2
        Then

        if label(n2) == 0 then {
        let N be the operand for n2;
        gencode(n1);
        print(OP N, top(RSTACK));
        }

/* case 2 */

else if ((1 < label(n1) < label(n2))

and( label(n1) < r))

then {

swap(RSTACK); gencode(n2);

R := pop(RSTACK); gencode(n1);

/* R holds the result of n2 */

print(OP R, top(RSTACK));

push (RSTACK,R);

swap(RSTACK);

}
```

CODE:

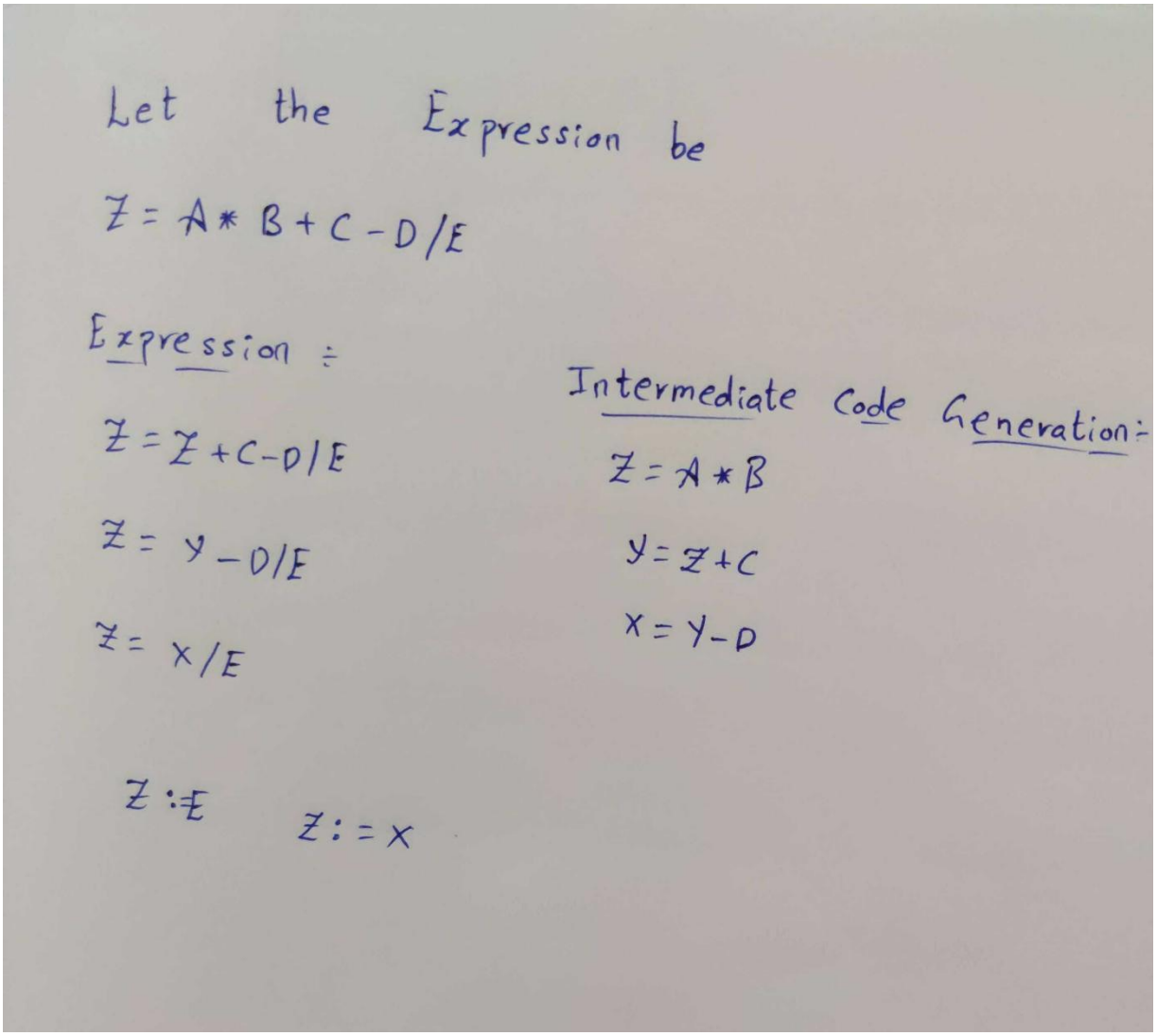
```
#include"stdio.h"
#include"string.h"
int i=1,j=0,no=0,tmpch=90;
char str[100],left[15],right[15];
void findopr();
void explore();
void fleft(int);
void fright(int);
struct exp
{
 int pos;
 char op;
}k[15];
void main()
{

 printf("\t\tINTERMEDIATE CODE GENERATION\n\n");
 printf("Enter the Expression :");
 scanf("%s",str);
 printf("The intermediate code:\t\tExpression\n");
 findopr();
 explore();

}
void findopr()
{
```

```
for(i=0;str[i]!='\0';i++)
 if(str[i]==':')
 {
 k[j].pos=i;
 k[j++].op=':';
 }
for(i=0;str[i]!='\0';i++)
 if(str[i]=='/')
 {
 k[j].pos=i;
 k[j++].op='/';
 }
for(i=0;str[i]!='\0';i++)
 if(str[i]=='*')
 {
 k[j].pos=i;
 k[j++].op='*';
 }
for(i=0;str[i]!='\0';i++)
 if(str[i]=='+')
 {
 k[j].pos=i;
 k[j++].op='+';
 }
for(i=0;str[i]!='\0';i++)
 if(str[i]=='-')
 {
 k[j].pos=i;
 k[j++].op='-';
 }
```

```
}
void explore()
{
 i=1;
 while(k[i].op!='\0')
 {
  fleft(k[i].pos);
  fright(k[i].pos);
  str[k[i].pos]=tmpch--;
  printf("\t%c := %s%c%s\t\t",str[k[i].pos],left,k[i].op,right);
  for(j=0;j <strlen(str);j++)
   if(str[j]!='$')
    printf("%c",str[j]);
  printf("\n");
  i++;
 }
 fright(-1);
 if(no==0)
 {
  fleft(strlen(str));
  printf("\t%s := %s",right,left);
 }
 printf("\t%s :=  %c",right,str[k[--i].pos]);
}
void fleft(int x)
{
 int w=0,flag=0;
 x--;
 while(x!= -1 &&str[x]!= '+' &&str[x]!='*'&&str[x]!='='&&str[x]!='\0'&&str[x]!='-
'&&str[x]!='/'&&str[x]!=':')
 {
```

```
  if(str[x]!='$'&& flag==0)

  {

  left[w++]=str[x];

  left[w]='\0';

  str[x]='$';

  flag=1;

  }

  x--;

 }

}

void fright(int x)

{

 int w=0,flag=0;

 x++;

 while(x!= -1 && str[x]!= '+'&&str[x]!='*'&&str[x]!='\0'&&str[x]!='='&&str[x]!=':'&&str[x]!='-
'&&str[x]!='/')

 {

  if(str[x]!='$'&& flag==0)

  {

  right[w++]=str[x];

  right[w]='\0';

  str[x]='$';

  flag=1;

  }

  x++;

 }

}
```

MANUAL CALCULATION:



OUTPUT:

```
12:MarchLong - (master)$ gpp DazzlingGCDPair.cpp
13:MarchLong - (master)$ ./a.out
               INTERMEDIATE CODE GENERATION

Enter the Expression :Z=A*B+C-D/E
The intermediate code:          Expression
        Z := A*B                Z=Z+C-D/E
        Y := Z+C                Z=Y-D/E
        X := Y-D                Z=X/E
        Z := E  Z :=  X14:MarchLong - (master)$
```

RESULT:

THE IMPLEMENTATION OF INTERMEDIATE CODE GENERATION OF THE EXPRESSION IS DONE SUCCESSFULLY IN C.