FR. Conceicao Rodrigues College of Engineering Department of Computer Engineering

8. Write a program to implement Restoring/Non Restoring Algorithm for Division.

1. Course, Subject & Experiment Details

Academic Year	**2023-24**	**Estimated Time**	**Experiment No. 8– 02 Hours**
Course & Semester	**S.E. (Computers) – Sem. III**	**Subject Name**	**Digital Logic & Computer Organization and Architecture**
Chapter No.	**2**	**Chapter Title**	**Data Representation and Arithmetic algorithms**
Experiment Type	**Software**	**Subject Code**	**CSC304**

Rubrics

Roll No	Date of Performance	Timeline (2)	Practical Skill & Applied Knowledge **(4)**	Output (4)	Total (10)
	Date of Submission:				

2. Aim & Objective of Experiment

- Understanding behaviour of Division algorithm for unsigned numbers
- Implementing Restoring / Non-restoring Division algorithms.

3. Problem Statement

Write a C/ Java / Python program to implement Restoring / Non restoring algorithm for Division.

4. Brief Theoretical Description

Division operation implements as follows: it position the divisor appropriately with respect to the dividend and performs a subtraction If the reminder is Zero or positive, a quotient bit of 1 is determined, the remainder is extended by another bit of the dividend, the divisor is repositioned, and another subtraction is performed. On the other hand, if the remainder is negative, a quotient bit of 0 is determined, the dividend is restored by adding back the divisor, and the divisor is repositioned for another subtraction.

Restoring Division

An n-bit positive divisor is loaded into register M and an n-bit positive dividend is loaded into register Q at the start of the operation. Register A is set to 0. After the division is complete, the n-bit quotients in register Q and the remainder is in register A. the required subtraction are facilitated by using 2’s complement arithmetic. The extra bit position at the left end of both A and M accommodates the sign bit during subtractions.

Non-restoring Division

The restoring division algorithm can be improved by avoiding the need for restoring A after an unsuccessful subtraction. Subtraction is said to be unsuccessful is the result is negative. Consider the sequence of operation that takes place after the subtraction operation in the preceding algorithm. If A is positive, we shift left and subtract M, that is, we perform 2A-M. If A is negative, we restore it by performing A+M, and then we shift it left and subtract M. This is equivalent to performing 2A+M. the q0 bit is appropriately set 0 or 1 after the correct operation has been performed.

Algorithm:

Restoring Division

Do the following n times:

1. Shift A and Q left one binary position.
2. Subtract M from A, and place the answer back in A.
3. If the sign of A is 1, set q0 to 0 add M back to A (that is restore A); otherwise, set q0 to 1.

Non-restoring Division

Step 1: do the following n times:

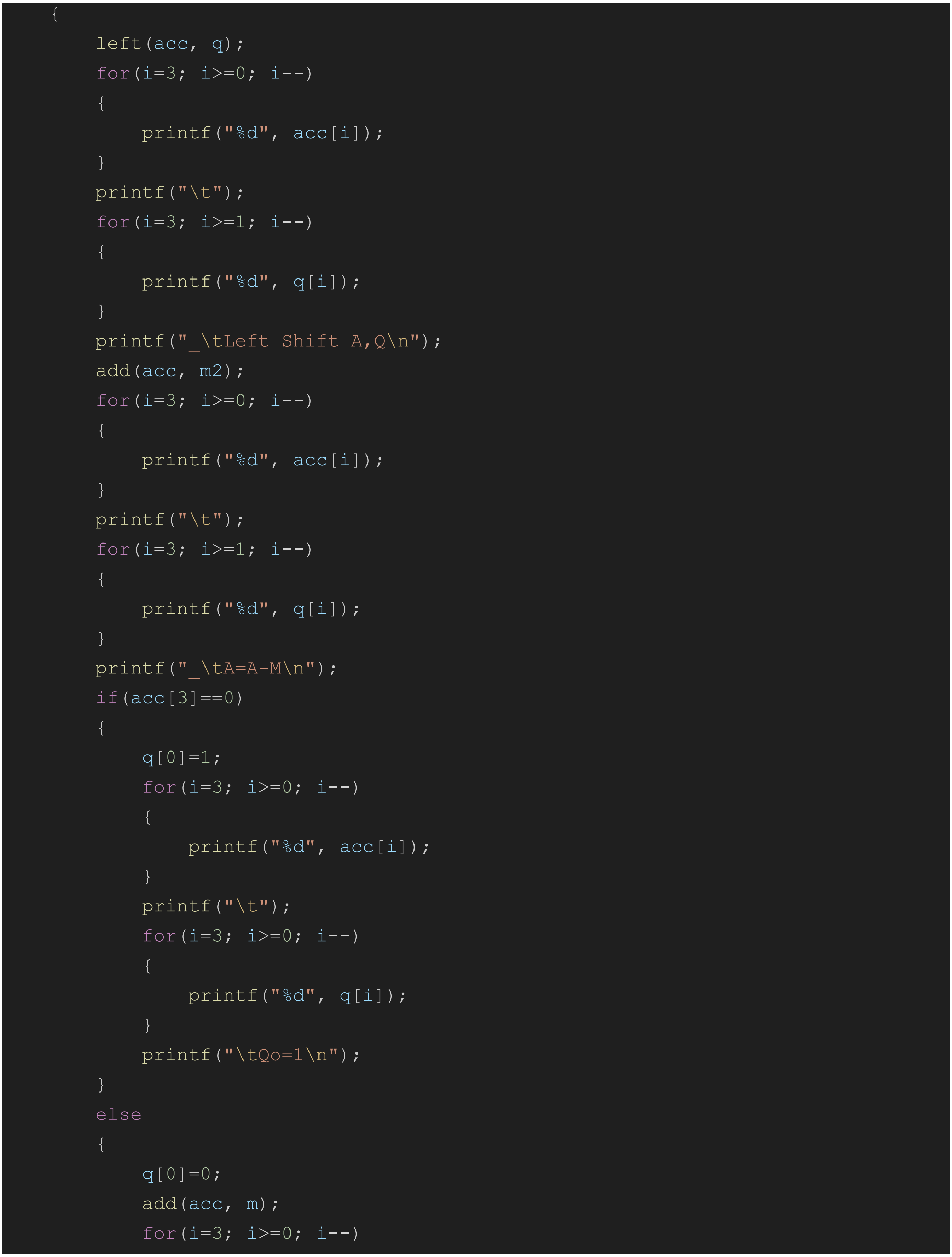
1. If the sign of A is 0, shift A and Q left one bit position and subtract M from A; otherwise, shift A and Q left and add M to A.
2. Now, if the sign of A is 0, set q0 to 1; otherwise, set q0 to 0.

Step 2: if the sign of A is 1, add M to A

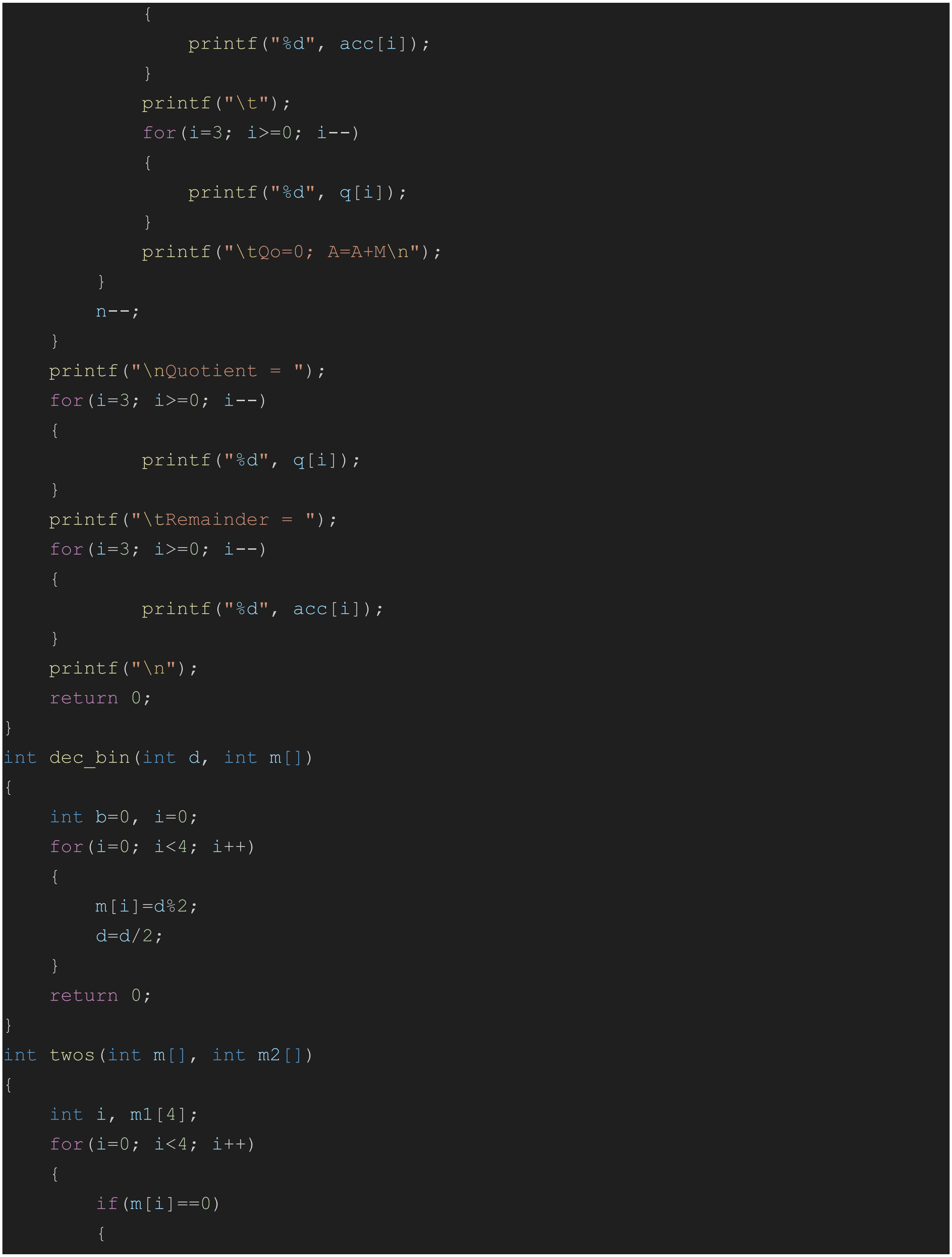
5. *Attach the program*

Restoring method:-

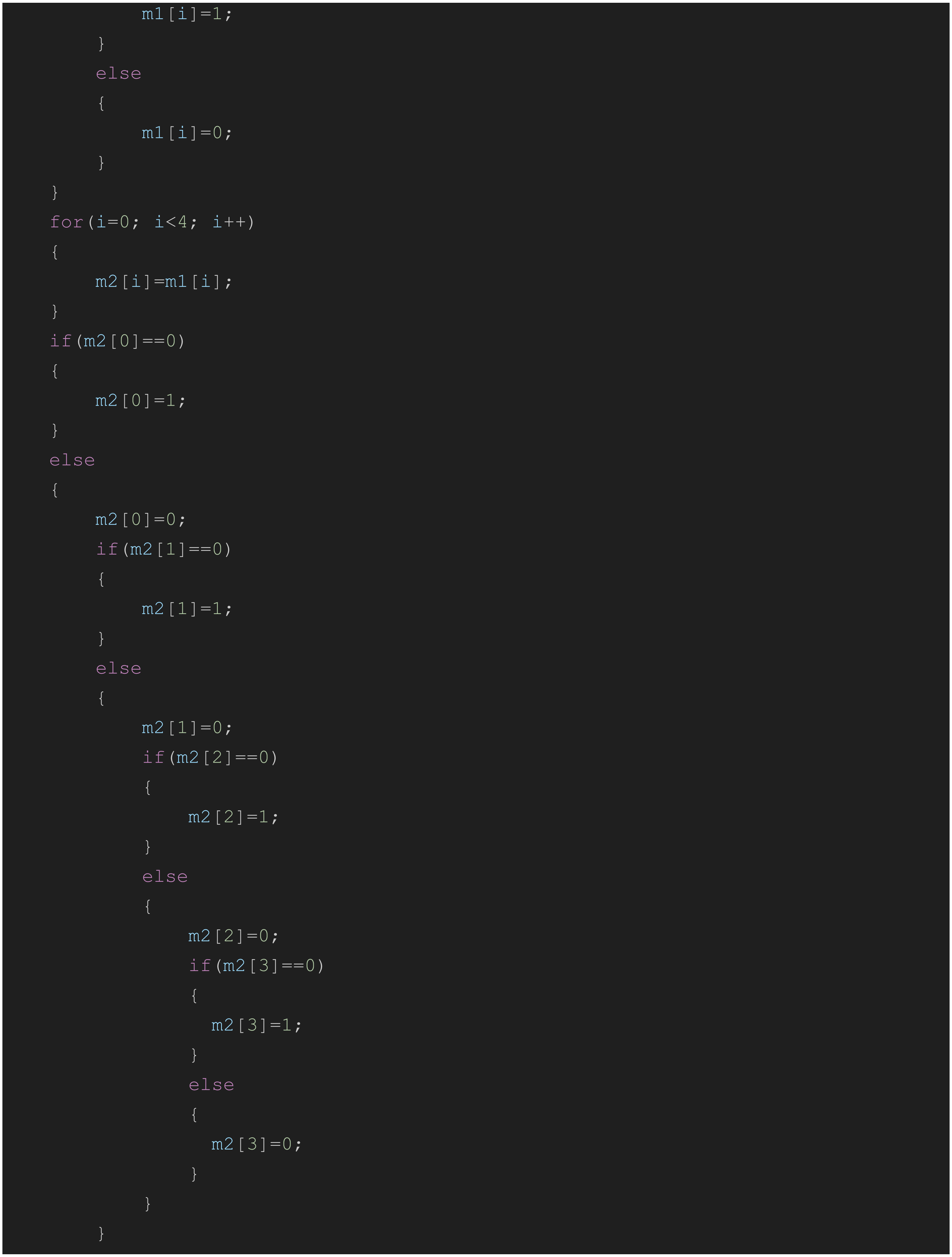
```
#include <stdio.h>
#include <stdlib.h>

int dec_bin(int, int []);
int twos(int [], int []);
int left(int [], int []);
int add(int [], int []);

int main()
{
    int a, b, m[4]={0,0,0,0}, q[4]={0,0,0,0}, acc[4]={0,0,0,0}, m2[4], i, n=4;
    printf("Enter the Dividend: ");
    scanf("%d", &a);
    printf("Enter the Divisor: ");
    scanf("%d", &b);
    dec_bin(a, q);
    dec_bin(b, m);
    twos(m, m2);
    printf("\nA\tQ\tComments\n");
    for(i=3; i>=0; i--)
    {
        printf("%d", acc[i]);
    }
    printf("\t");
    for(i=3; i>=0; i--)
    {
        printf("%d", q[i]);
    }
    printf("\tStart\n");
    while(n>0)
```



```
    {
        left(acc, q);
        for(i=3; i>=0; i--)
        {
            printf("%d", acc[i]);
        }
        printf("\t");
        for(i=3; i>=1; i--)
        {
            printf("%d", q[i]);
        }
        printf("_\tLeft Shift A,Q\n");
        add(acc, m2);
        for(i=3; i>=0; i--)
        {
            printf("%d", acc[i]);
        }
        printf("\t");
        for(i=3; i>=1; i--)
        {
            printf("%d", q[i]);
        }
        printf("_\tA=A-M\n");
        if(acc[3]==0)
        {
            q[0]=1;
            for(i=3; i>=0; i--)
            {
                printf("%d", acc[i]);
            }
            printf("\t");
            for(i=3; i>=0; i--)
            {
                printf("%d", q[i]);
            }
            printf("\tQo=1\n");
        }
        else
        {
            q[0]=0;
            add(acc, m);
            for(i=3; i>=0; i--)
```



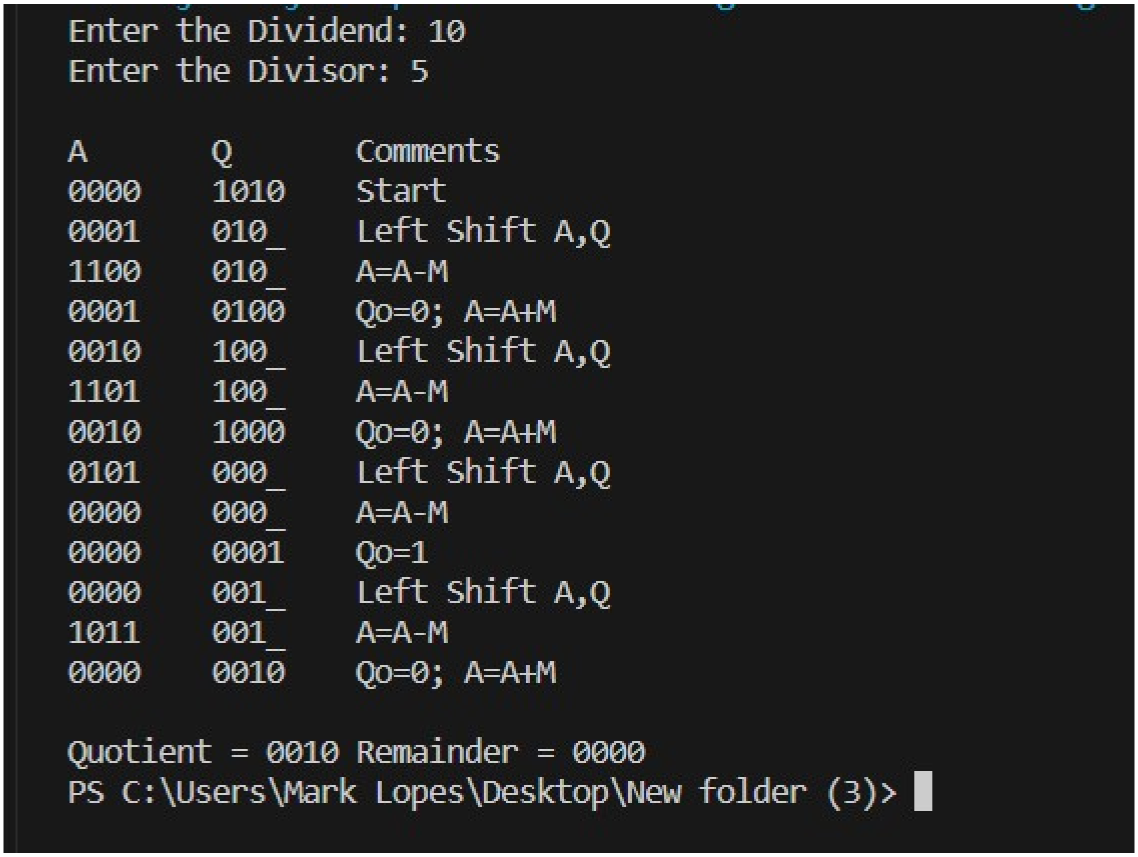
```
            {
                printf("%d", acc[i]);
            }
            printf("\t");
            for(i=3; i>=0; i--)
            {
                printf("%d", q[i]);
            }
            printf("\tQo=0; A=A+M\n");
        }
        n--;
    }
    printf("\nQuotient = ");
    for(i=3; i>=0; i--)
    {
            printf("%d", q[i]);
    }
    printf("\tRemainder = ");
    for(i=3; i>=0; i--)
    {
            printf("%d", acc[i]);
    }
    printf("\n");
    return 0;
}
int dec_bin(int d, int m[])
{
    int b=0, i=0;
    for(i=0; i<4; i++)
    {
        m[i]=d%2;
        d=d/2;
    }
    return 0;
}
int twos(int m[], int m2[])
{
    int i, m1[4];
    for(i=0; i<4; i++)
    {
        if(m[i]==0)
        {
```



```
        m1[i]=1;
    }
    else
    {
        m1[i]=0;
    }
}
for(i=0; i<4; i++)
{
    m2[i]=m1[i];
}
if(m2[0]==0)
{
    m2[0]=1;
}
else
{
    m2[0]=0;
    if(m2[1]==0)
    {
        m2[1]=1;
    }
    else
    {
        m2[1]=0;
        if(m2[2]==0)
        {
            m2[2]=1;
        }
        else
        {
            m2[2]=0;
            if(m2[3]==0)
            {
              m2[3]=1;
            }
            else
            {
              m2[3]=0;
            }
        }
    }
    }
    return 0;
}
int left(int acc[], int q[])
{
    int i;
    for(i=3; i>0; i--)
    {
        acc[i]=acc[i-1];
    }
    acc[0]=q[3];
    for(i=3; i>0; i--)
    {
        q[i]=q[i-1];
    }
}
int add(int acc[], int m[])
{
  int i, carry=0;
  for(i=0; i<4; i++)
  {
    if(acc[i]+m[i]+carry==0)
    {
      acc[i]=0;
      carry=0;
    }
    else if(acc[i]+m[i]+carry==1)
    {
      acc[i]=1;
      carry=0;
    }
    else if(acc[i]+m[i]+carry==2)
    {
      acc[i]=0;
      carry=1;
    }
    else if(acc[i]+m[i]+carry==3)
    {
      acc[i]=1;
      carry=1;
    }
  }
```

```
return 0;
}
```

Output:-



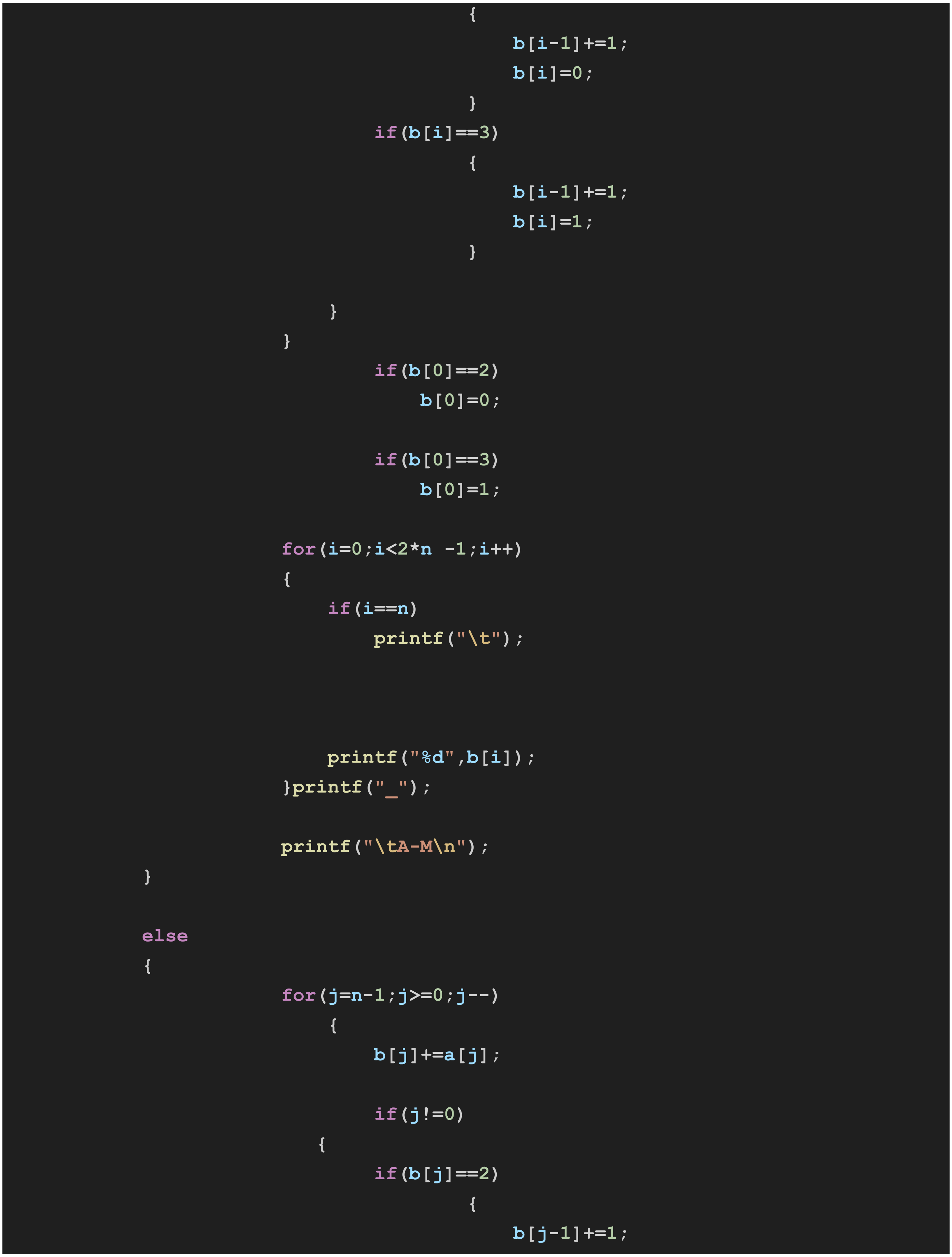
```
Enter the Dividend: 10
Enter the Divisor: 5

A       Q       Comments
0000    1010    Start
0001    010_    Left Shift A,Q
1100    010_    A=A-M
0001    0100    Qo=0; A=A+M
0010    100_    Left Shift A,Q
1101    100_    A=A-M
0010    1000    Qo=0; A=A+M
0101    000_    Left Shift A,Q
0000    000_    A=A-M
0000    0001    Qo=1
0000    001_    Left Shift A,Q
1011    001_    A=A-M
0000    0010    Qo=0; A=A+M

Quotient = 0010 Remainder = 0000
PS C:\Users\Mark Lopes\Desktop\New folder (3)>
```

Non-Restoring method

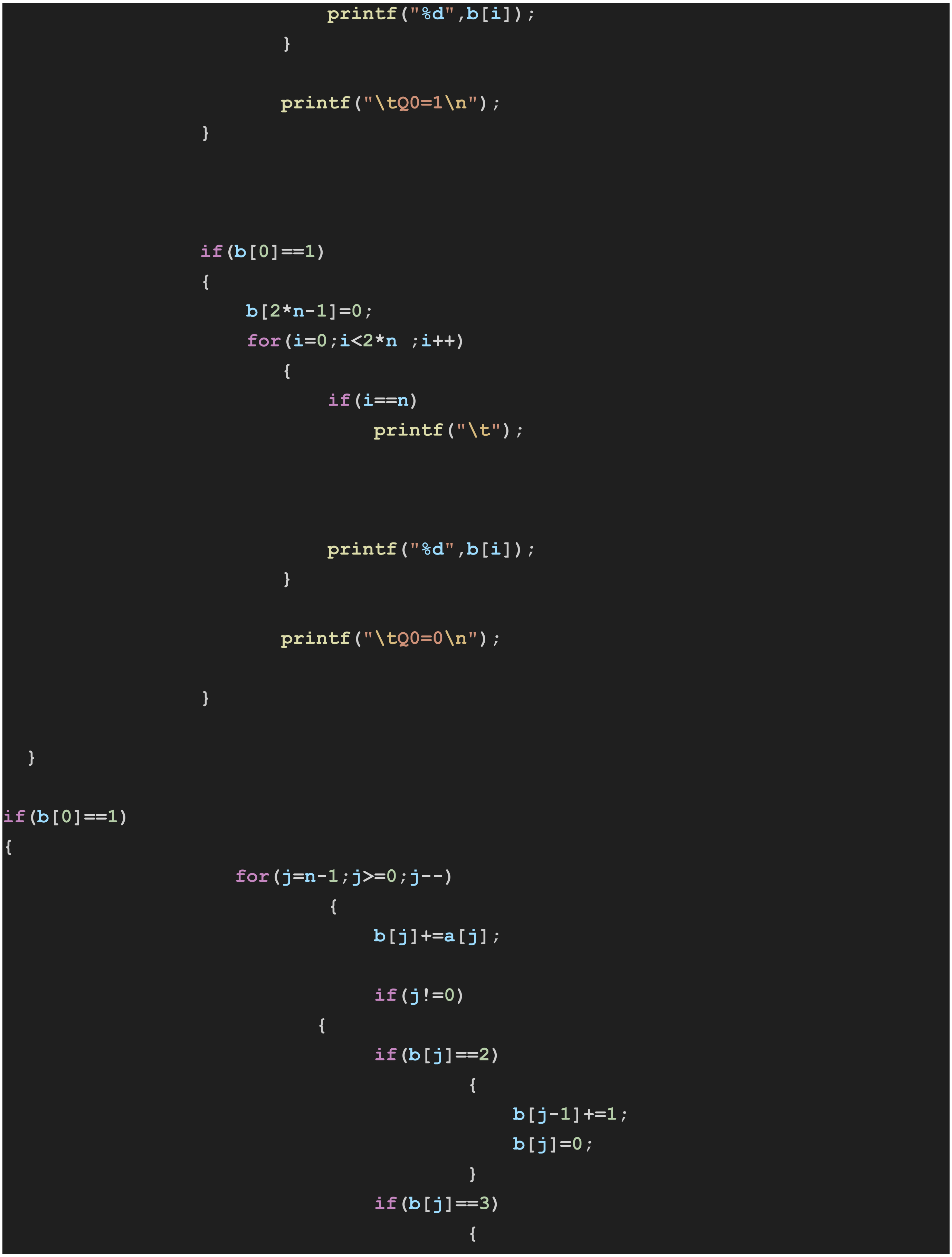
```
#include <math.h>
#include <stdio.h>
int main()
{
int a[50],a1[50],b[50],d=0,i,j;
  int n1,n2, c, k1,k2,n,k,quo=0,rem=0;
    printf("Enter the number of bits\n");
    scanf("%d",&n);
  printf("Enter the divisor and dividend\n");
  scanf("%d %d", &n1,&n2);

  for (c = n-1; c >= 0; c--)
```

```
{
  k1 = n1 >> c;

  if (k1 & 1)
    a[n-1-c]=1;
  else
   a[n-1-c]=0;

   k2 = n2 >> c;

  if (k2 & 1)
    b[2*n-1-c]=1;
  else
   b[2*n-1-c]=0;

}

for(i=0;i<n;i++)
{
    if(a[i]==0)
      a1[i]=1;
    else
      a1[i]=0;
}

a1[n-1]+=1;

if(a1[n-1]==2)
{
        for(i=n-1;i>0;i--)
    {
           if(a1[i]==2)
        {
            a1[i-1]+=1;
            a1[i]=0;
        }
    }
}
if(a1[0]==2)
  a1[0]=0;

for( i=0;i<n;i++)
```

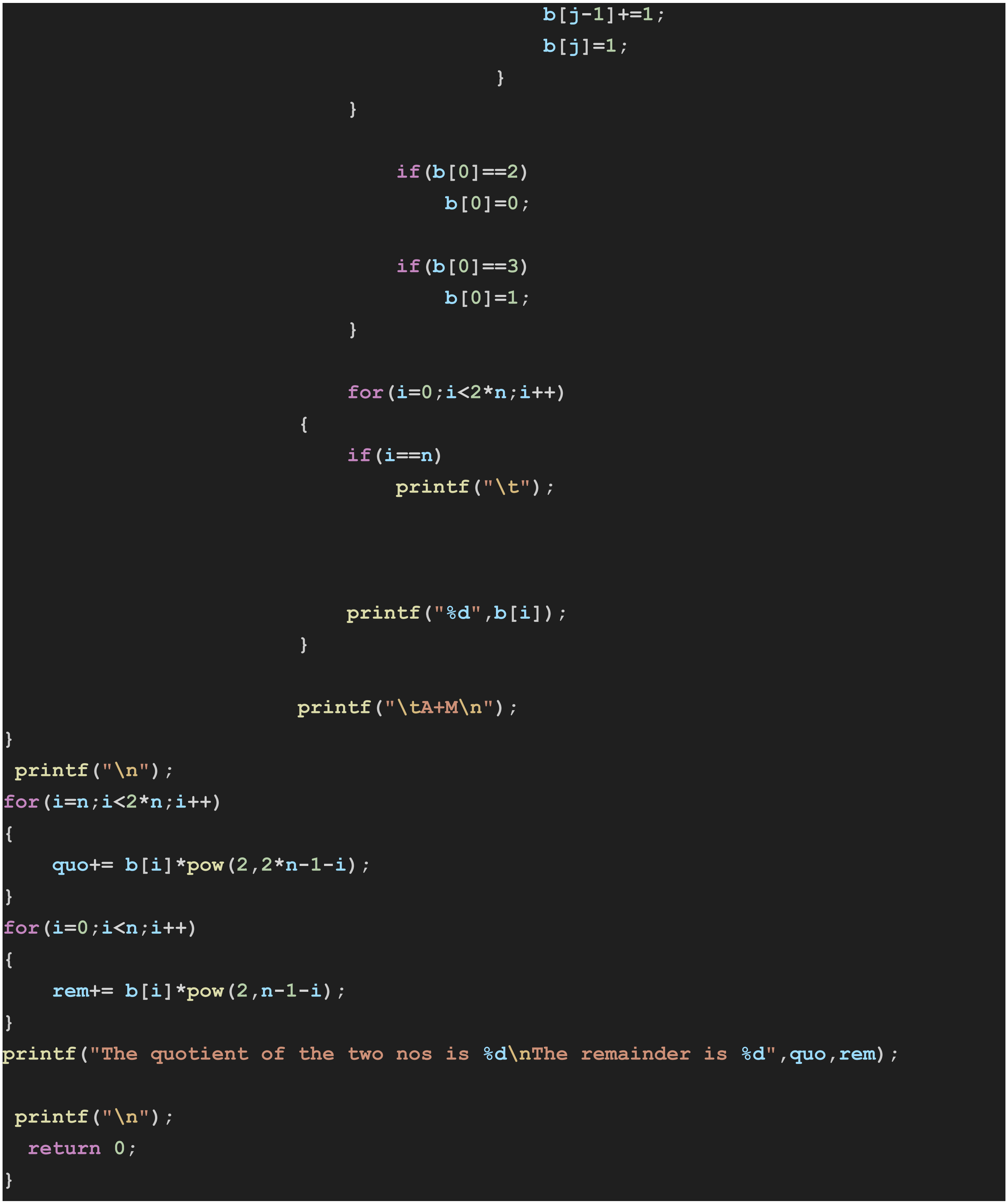
```
  {
      b[i]=0;

  }

printf("A\tQ\tPROCESS\n");

  for(i=0;i<2*n;i++)
{
    if(i==n)
        printf("\t");

    printf("%d",b[i]);
}
 printf("\n");

  for(k=0;k<n;k++)
  {
      for(j=0;j<2*n-1;j++)
        {
          b[j]=b[j+1];

        }

        for(i=0;i<2*n -1;i++)
        {
            if(i==n)
                printf("\t");
            printf("%d",b[i]);
        }printf("_");

        printf("\tLEFT SHIFT\n");

            if(b[0]==0)
            {
                        for(i=n-1;i>=0;i--)
                        {
                            b[i]+=a1[i];

                                if(i!=0)
                           {
                                if(b[i]==2)
```



```
                            {
                                b[i-1]+=1;
                                b[i]=0;
                            }
                    if(b[i]==3)
                            {
                                b[i-1]+=1;
                                b[i]=1;
                            }


                }
            }
                    if(b[0]==2)
                        b[0]=0;

                    if(b[0]==3)
                        b[0]=1;

            for(i=0;i<2*n -1;i++)
            {
                if(i==n)
                    printf("\t");



                printf("%d",b[i]);
            }printf("_");

            printf("\tA-M\n");
}

else
{
            for(j=n-1;j>=0;j--)
                {
                    b[j]+=a[j];

                    if(j!=0)
               {
                    if(b[j]==2)
                            {
                                b[j-1]+=1;
                                b[j]=0;
                            }
                    if(b[j]==3)
                            {
                                b[j-1]+=1;
                                b[j]=1;
                            }
                }

                    if(b[0]==2)
                        b[0]=0;

                    if(b[0]==3)
                        b[0]=1;
                }

                for(i=0;i<2*n -1;i++)
            {
                if(i==n)
                    printf("\t");



                printf("%d",b[i]);
            }printf("_");

            printf("\tA+M\n");

}




     if(b[0]==0)
     {
         b[2*n-1]=1;
         for(i=0;i<2*n ;i++)
            {
                if(i==n)
                    printf("\t");
```

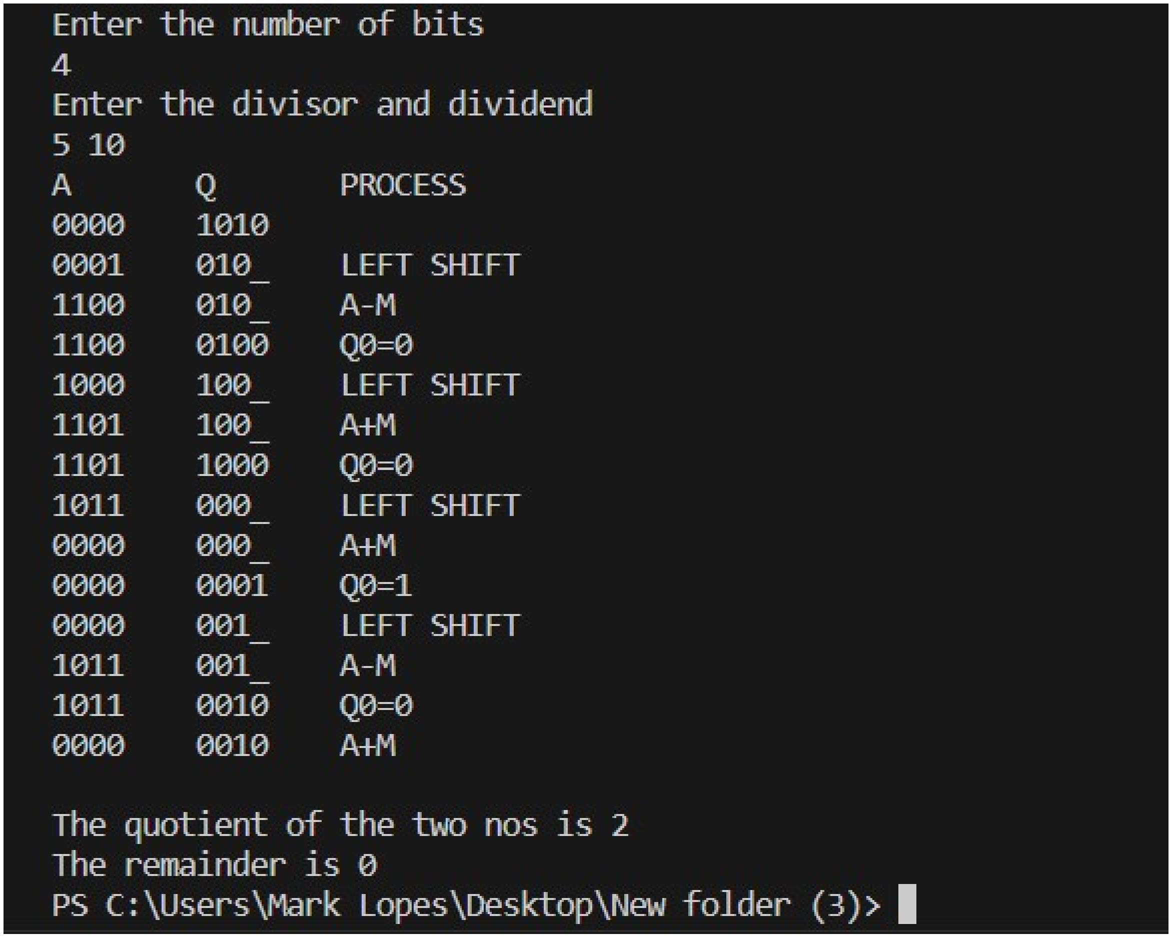


```
                            printf("%d",b[i]);
                        }

                        printf("\tQ0=1\n");
                 }



                 if(b[0]==1)
                 {
                     b[2*n-1]=0;
                     for(i=0;i<2*n ;i++)
                        {
                            if(i==n)
                                printf("\t");



                            printf("%d",b[i]);
                        }

                        printf("\tQ0=0\n");

                 }

  }

if(b[0]==1)
{
                    for(j=n-1;j>=0;j--)
                            {
                                b[j]+=a[j];

                                if(j!=0)
                           {
                                if(b[j]==2)
                                        {
                                            b[j-1]+=1;
                                            b[j]=0;
                                        }
                                if(b[j]==3)
                                        {
```



```
                                             b[j-1]+=1;
                                             b[j]=1;
                                         }
                             }

                                 if(b[0]==2)
                                     b[0]=0;

                                 if(b[0]==3)
                                     b[0]=1;
                             }

                             for(i=0;i<2*n;i++)
                         {
                             if(i==n)
                                 printf("\t");



                             printf("%d",b[i]);
                         }

                         printf("\tA+M\n");
}
 printf("\n");
for(i=n;i<2*n;i++)
{
    quo+= b[i]*pow(2,2*n-1-i);
}
for(i=0;i<n;i++)
{
    rem+= b[i]*pow(2,n-1-i);
}
printf("The quotient of the two nos is %d\nThe remainder is %d",quo,rem);

 printf("\n");
  return 0;
}
```

Output:-



```
Enter the number of bits
4
Enter the divisor and dividend
5 10
A       Q       PROCESS
0000    1010
0001    010_    LEFT SHIFT
1100    010_    A-M
1100    0100    Q0=0
1000    100_    LEFT SHIFT
1101    100_    A+M
1101    1000    Q0=0
1011    000_    LEFT SHIFT
0000    000_    A+M
0000    0001    Q0=1
0000    001_    LEFT SHIFT
1011    001_    A-M
1011    0010    Q0=0
0000    0010    A+M

The quotient of the two nos is 2
The remainder is 0
PS C:\Users\Mark Lopes\Desktop\New folder (3)>
```

6. Conclusion:

Hence we can implement the restring and Non-restoring algorithm in C language for unsigned integers.