K. J. Somaiya College of Engineering, Mumbai-77
(A Constituent College of Somaiya Vidyavihar University)
Department of Computer Engineering

Batch: A2 Roll No.: 16010122041

Experiment / assignment / tutorial No: 3

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

TITLE : To study and implement Restoring method of division

AIM : The basis of algorithm is based on paper and pencil approach and the operation involves repetitive shifting with addition and subtraction. So the main aim is to depict the usual process in the form of an algorithm.

Expected OUTCOME of Experiment: (Mention CO /CO's attained here)

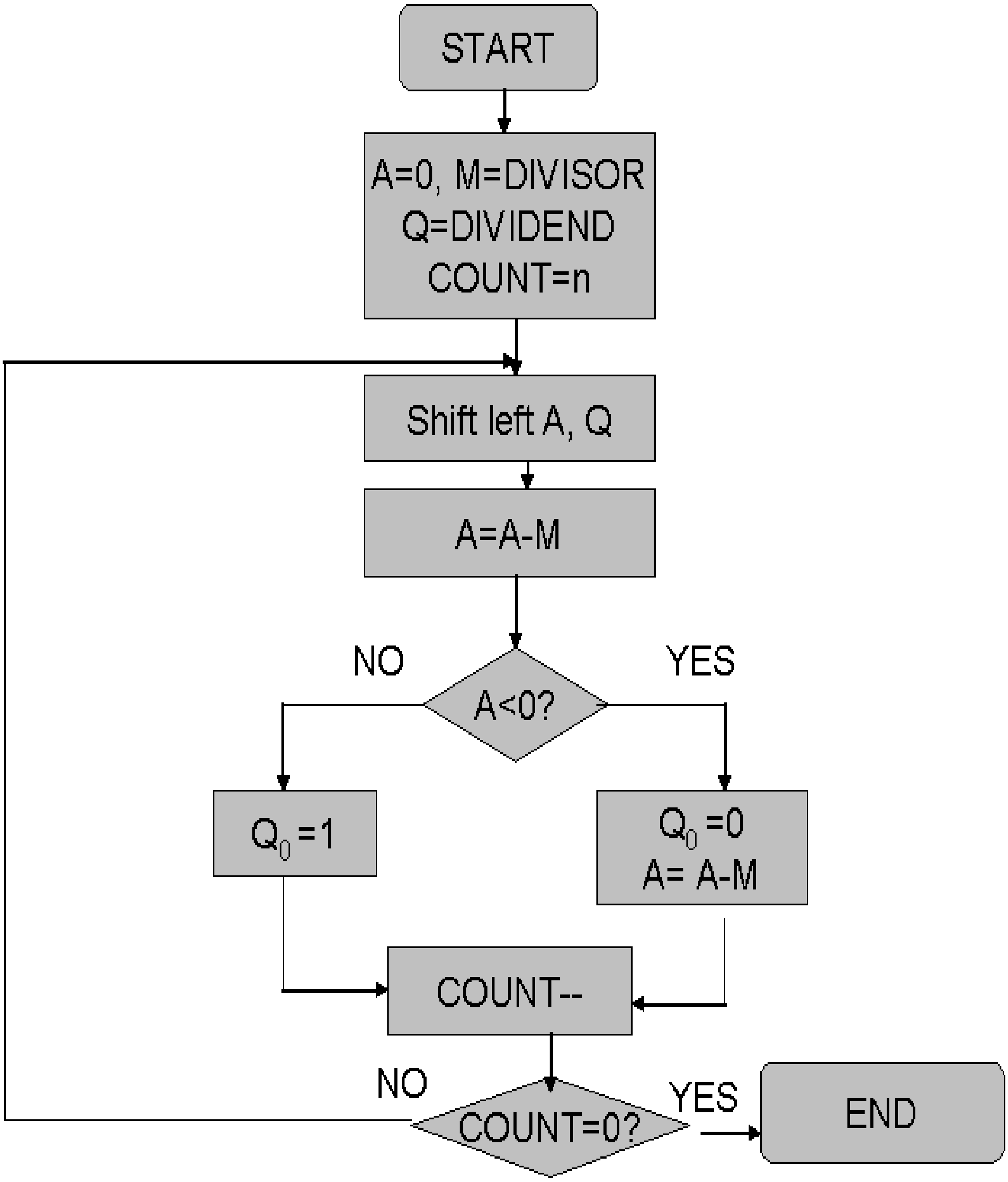
Books/ Journals/ Websites referred:

1. Carl Hamacher, Zvonko Vranesic and Safwat Zaky, "Computer Organization", Fifth Edition, TataMcGraw-Hill.
2. William Stallings, "Computer Organization and Architecture: Designing for Performance", Eighth Edition, Pearson.
3. Dr. M. Usha, T. S. Srikanth, "Computer System Architecture and Organization", First Edition, Wiley-India.

Pre Lab/ Prior Concepts:

The Restoring algorithm works with any combination of positive and negative numbers.

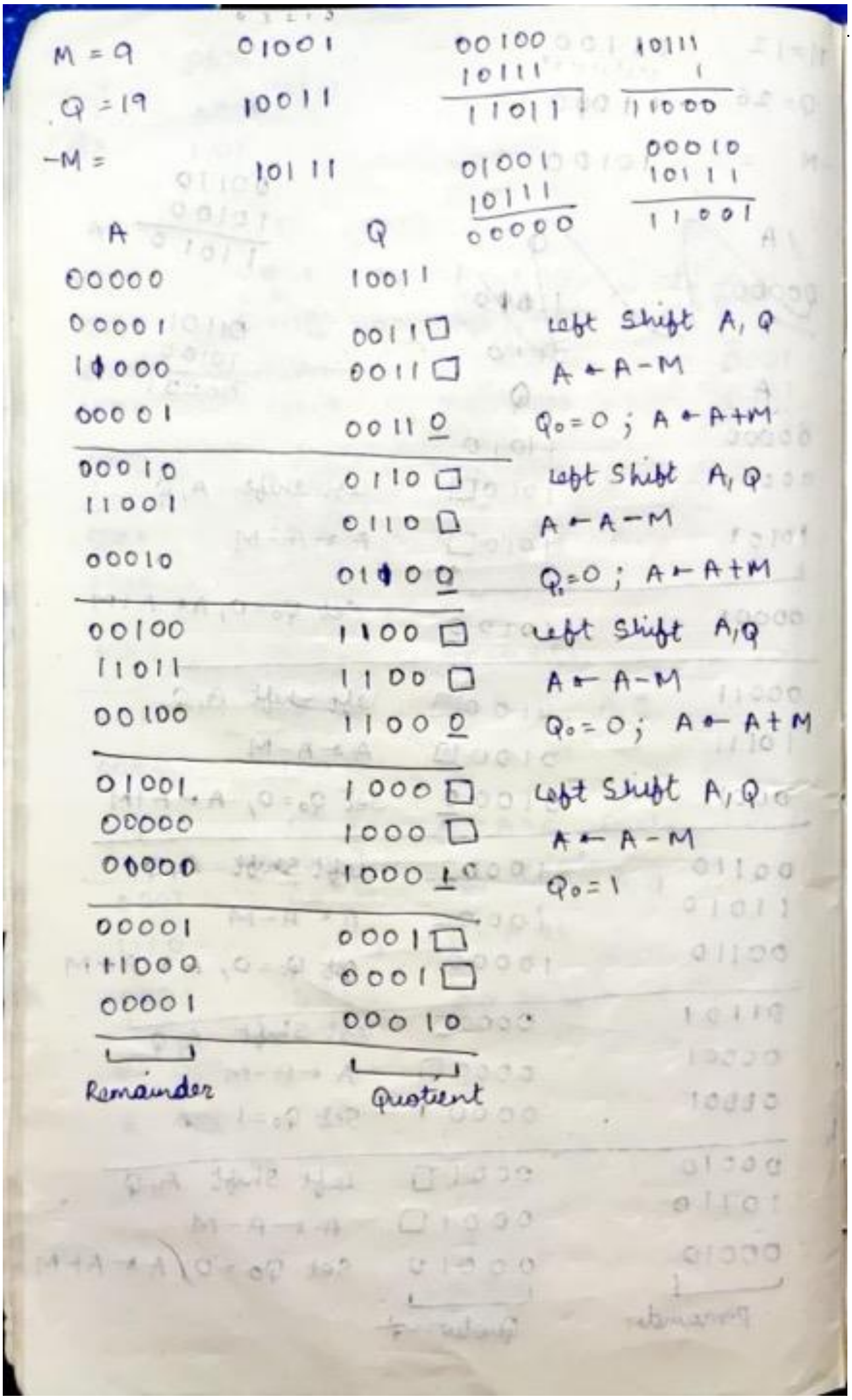
Flowchart for Restoring of Division:



Design Steps:

1. Start
2. Initialize A=0, M=Divisor, Q=Dividend and count=n (no of bits)
3. Left shift A, Q
4. If MSB of A and M are same
5. Then A=A-M
6. Else A=A+M
7. If MSB of previous A and present A are same
8. Q_0=0 & store present A
9. Else Q_0=0 & restore previous A
10. Decrement count.
11. If count=0 go to 11
12. Else go to 3
13. STOP

Example:- (Handwritten solved problems needs to be uploaded)



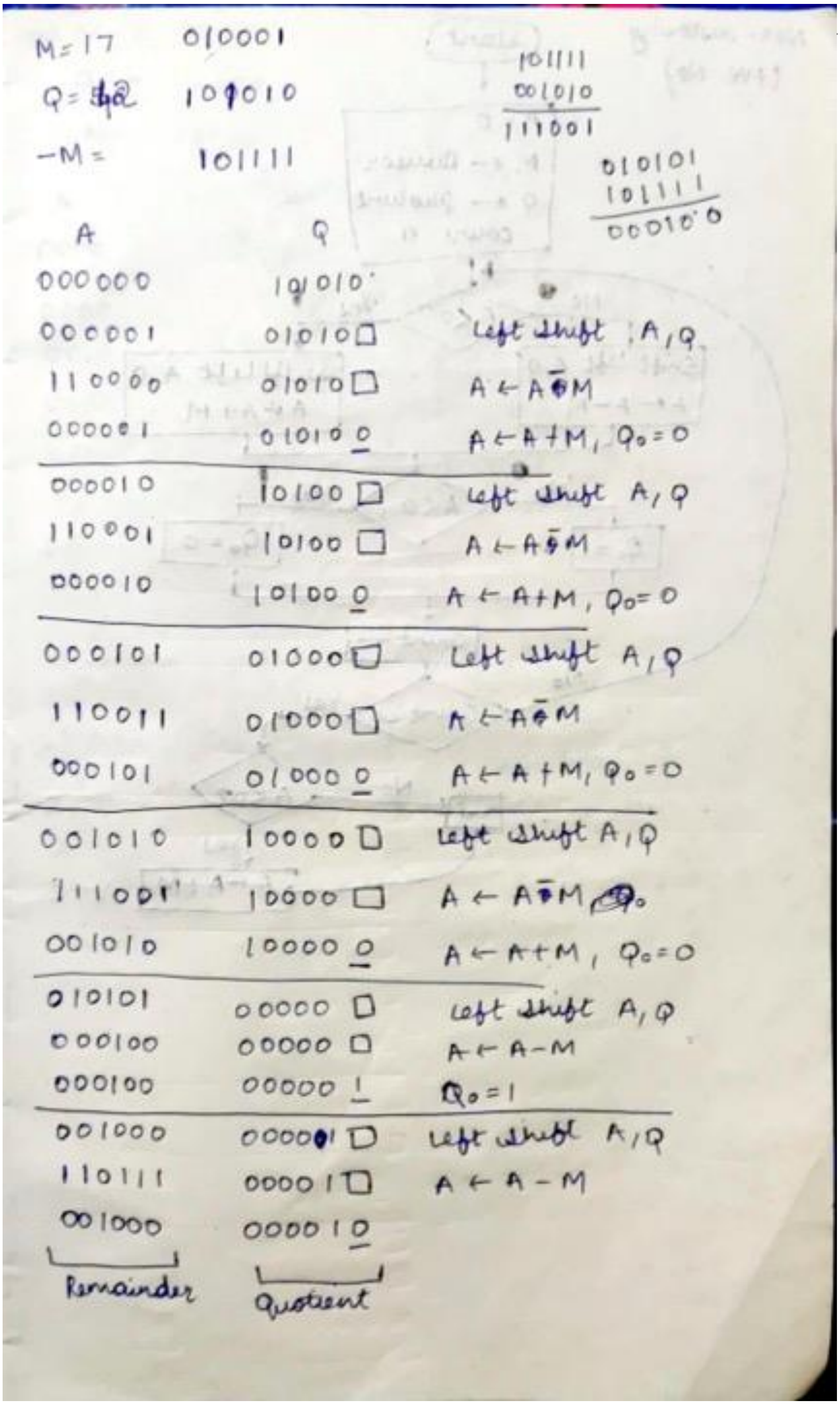
M = 9 01001

Q = 19 10011

−M = 10111

```
 00100      10111
 10111          1
 -----      -----
 11011      11000

 01001      00010
 10111      10111
 -----      -----
 00000      11001
```

A	Q	
00000	10011	
00001	0011□	Left Shift A, Q
1ɸ000	0011□	A ← A − M
00001	0011$\underline{0}$	$Q_0 = 0$; A ← A+M
00010	0110□	Left Shift A, Q
11001	0110□	A ← A − M
00010	0110$\underline{0}$	$Q_0 = 0$; A ← A+M
00100	1100□	Left Shift A, Q
11011	1100□	A ← A − M
00100	1100$\underline{0}$	$Q_0 = 0$; A ← A+M
01001	1000□	Left Shift A, Q
00000	1000□	A ← A − M
00000	1000$\underline{1}$	$Q_0 = 1$
00001	0001□	
11000	0001□	
00001	00010	
Remainder	Quotient	



M = 17 010001

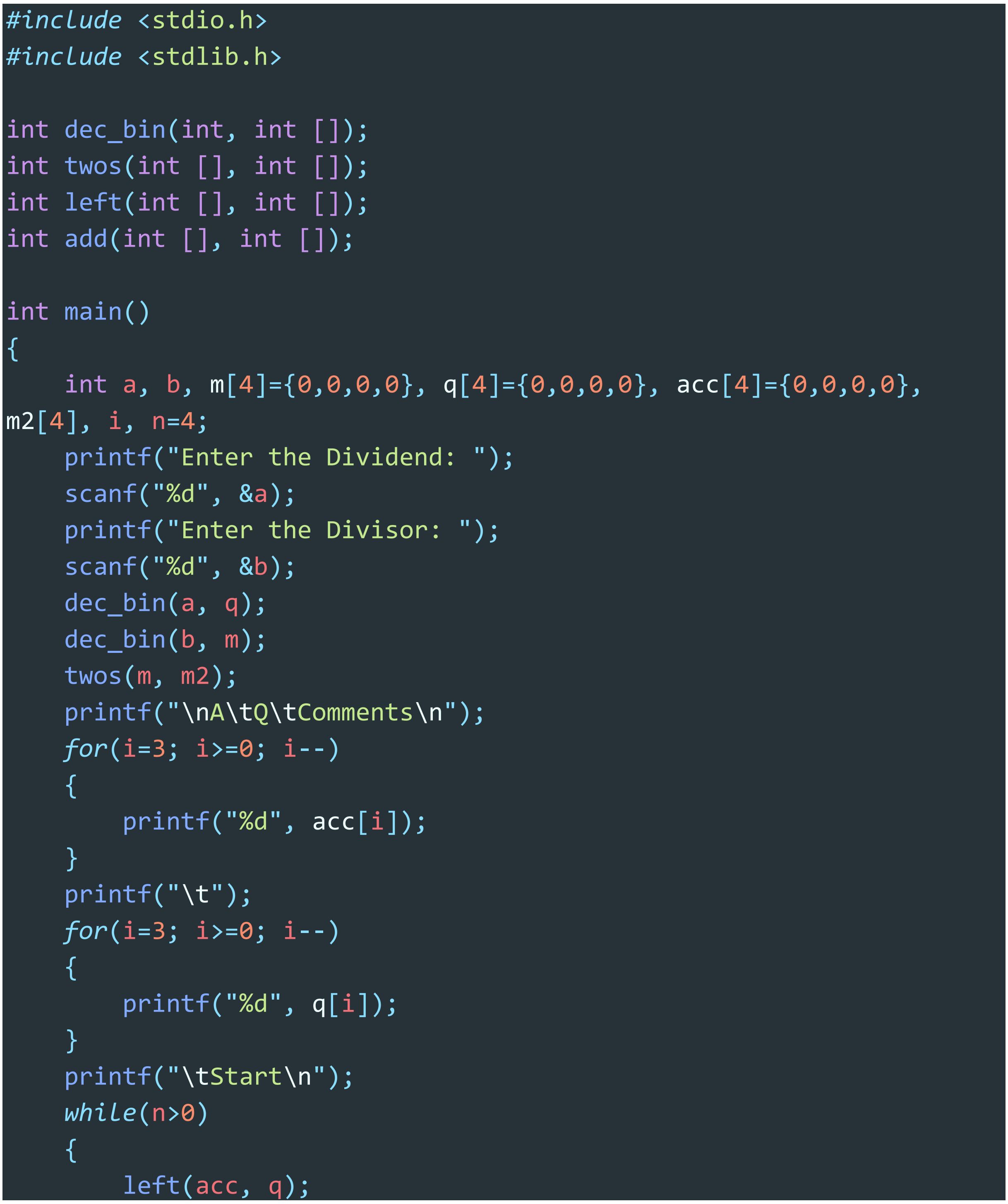
Q = 42 101010

−M = 101111

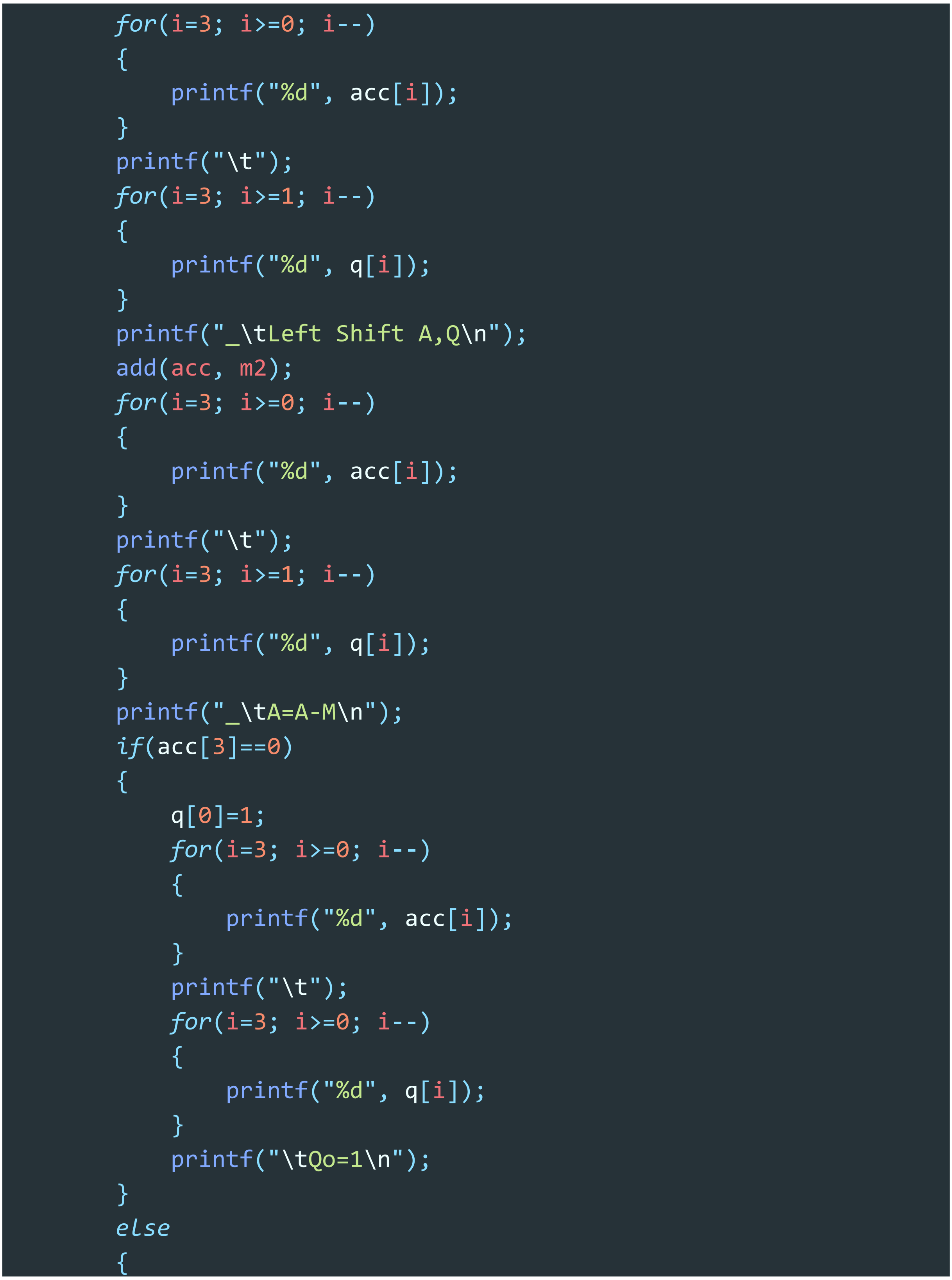
```
  101111
+ 001010
  ------
  111001

  010101
  101111
  ------
  000100
```

A	Q	
000000	101010	
000001	01010□	Left shift A, Q
110000	01010□	A ← A−M
000001	010100	A ← A+M, $Q_0 = 0$
000010	10100□	Left shift A, Q
110001	10100□	A ← A−M
000010	101000	A ← A+M, $Q_0 = 0$
000101	01000□	Left shift A, Q
110011	01000□	A ← A−M
000101	010000	A ← A+M, $Q_0 = 0$
001010	10000□	Left shift A, Q
111001	10000□	A ← A−M, ~~Q_0~~
001010	100000	A ← A+M, $Q_0 = 0$
010101	00000□	Left shift A, Q
000100	00000□	A ← A−M
000100	000001	$Q_0 = 1$
001000	00001□	Left shift A, Q
110111	00001□	A ← A−M
001000	000010	
Remainder	Quotient	

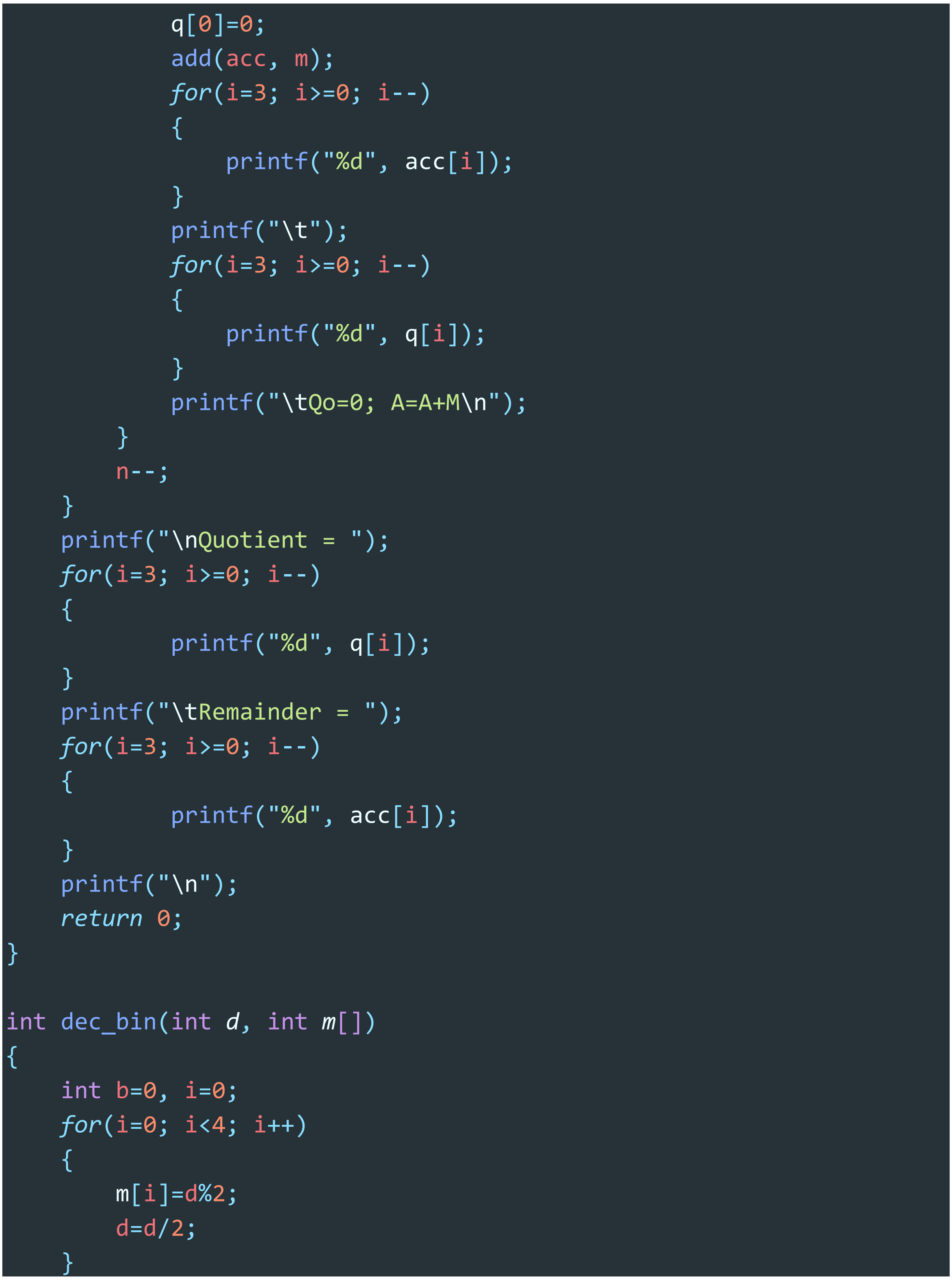
Code:



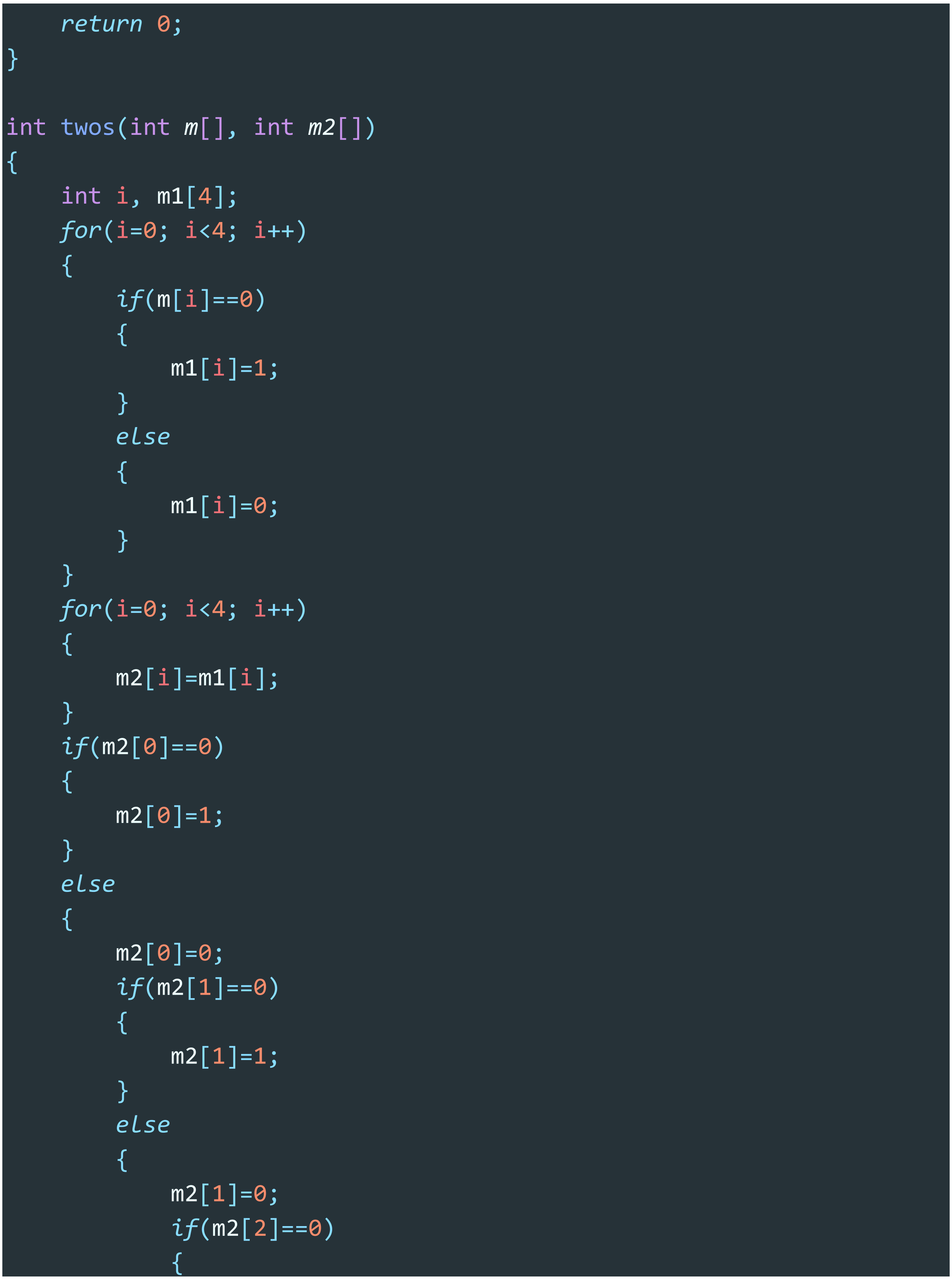
```
#include <stdio.h>
#include <stdlib.h>

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);
```



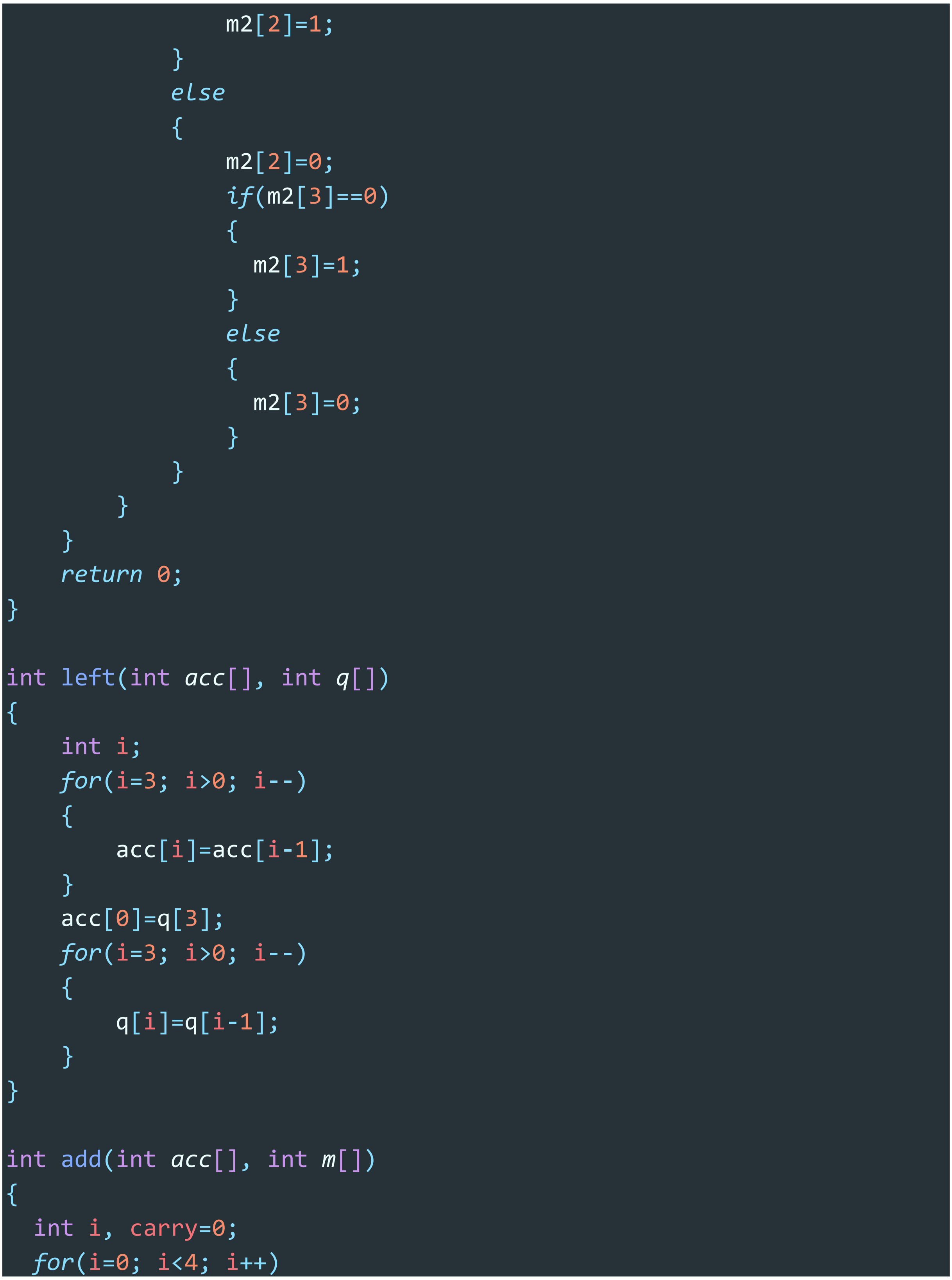
```
        for(i=3; i>=0; i--)
        {
            printf("%d", acc[i]);
        }
        printf("\t");
        for(i=3; i>=1; i--)
        {
            printf("%d", q[i]);
        }
        printf("_\tLeft Shift A,Q\n");
        add(acc, m2);
        for(i=3; i>=0; i--)
        {
            printf("%d",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



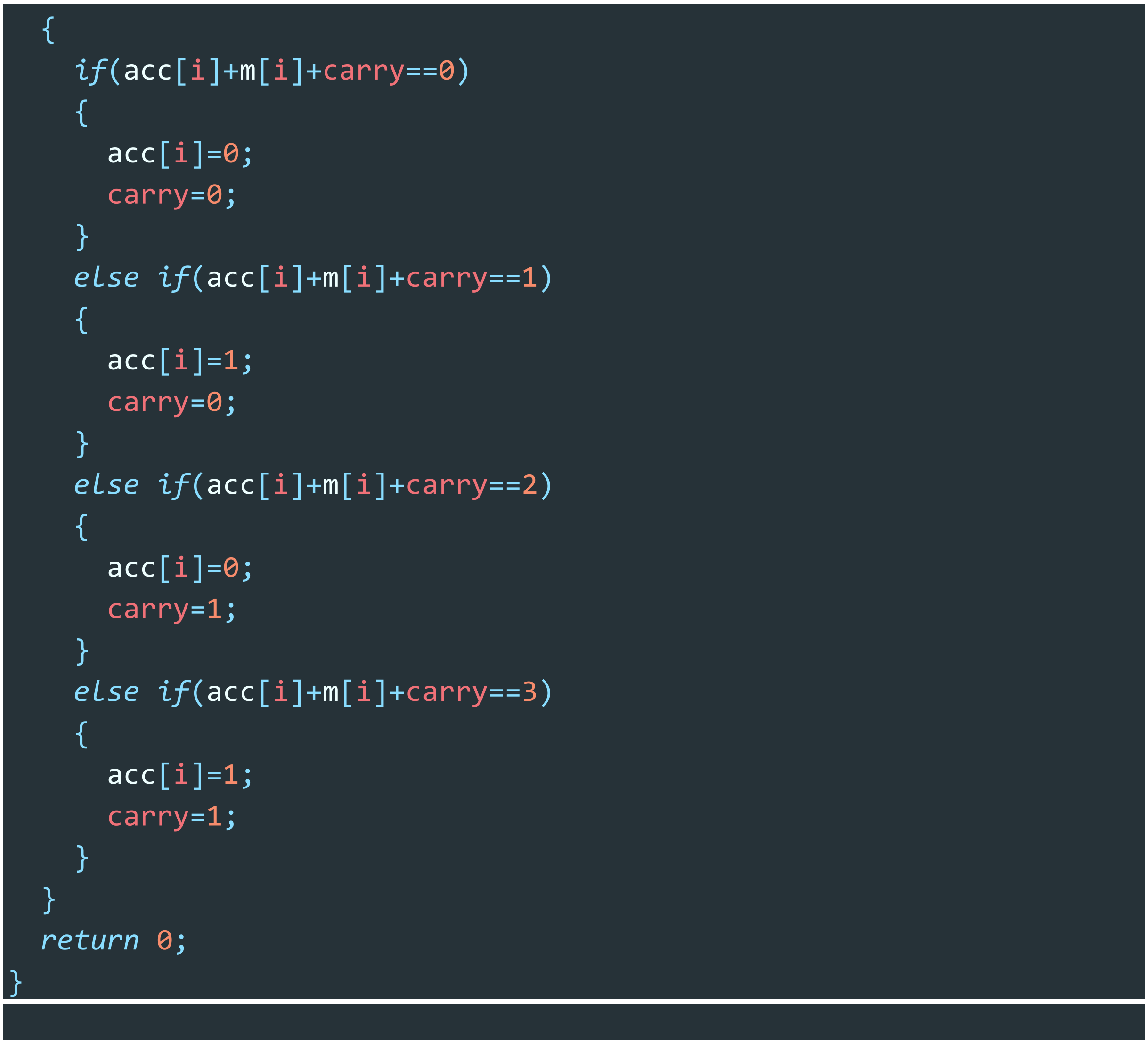
```
            q[0]=0;
            add(acc, m);
            for(i=3; i>=0; i--)
            {
                printf("%d", acc[i]);
            }
            printf("\t");
            for(i=3; i>=0; i--)
            {
                printf("%d", q[i]);
            }
            printf("\tQo=0; A=A+M\n");
        }
        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)
            {
```

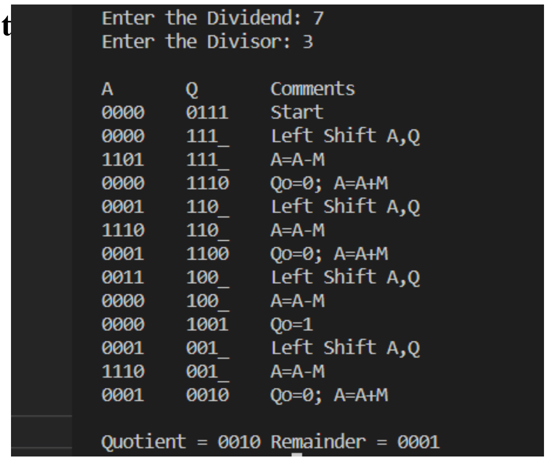


```
                m2[2]=1;
            }
            else
            {
                m2[2]=0;
                if(m2[3]==0)
                {
                  m2[3]=1;
                }
                else
                {
                  m2[3]=0;
                }
            }
        }
    }
    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: 7
Enter the Divisor: 3

A       Q       Comments
0000    0111    Start
0000    111_    Left Shift A,Q
1101    111_    A=A-M
0000    1110    Qo=0; A=A+M
0001    110_    Left Shift A,Q
1110    110_    A=A-M
0001    1100    Qo=0; A=A+M
0011    100_    Left Shift A,Q
0000    100_    A=A-M
0000    1001    Qo=1
0001    001_    Left Shift A,Q
1110    001_    A=A-M
0001    0010    Qo=0; A=A+M

Quotient = 0010 Remainder = 0001
```

Conclusion

The Restoring method of division has been studied and its implementation has been conducted successfully.

Post Lab Descriptive Questions

1. **What are the advantages of restoring division over non restoring division?**

In each step of your division calculation the result of the step is either 1 or 0,
depending if the dividend is less than or larger than the divisor.
You generally do a test subtraction for each digit step; if the result is positive or zero,
you note down a 1 as next digit of your quotient.
If the result is negative, you proceed with one of two strategies:
• restoring method: you add the divisor back, and put 0 as your next
quotient digit
• non-restoring method: you don't do that - you keep negative
remainder and a digit 1, and basically correct things by a
supplementary addition afterwards.

Date: _____________ **Signature of faculty in-charge**