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

Batch: D2 **Roll No.:16010122323**

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.

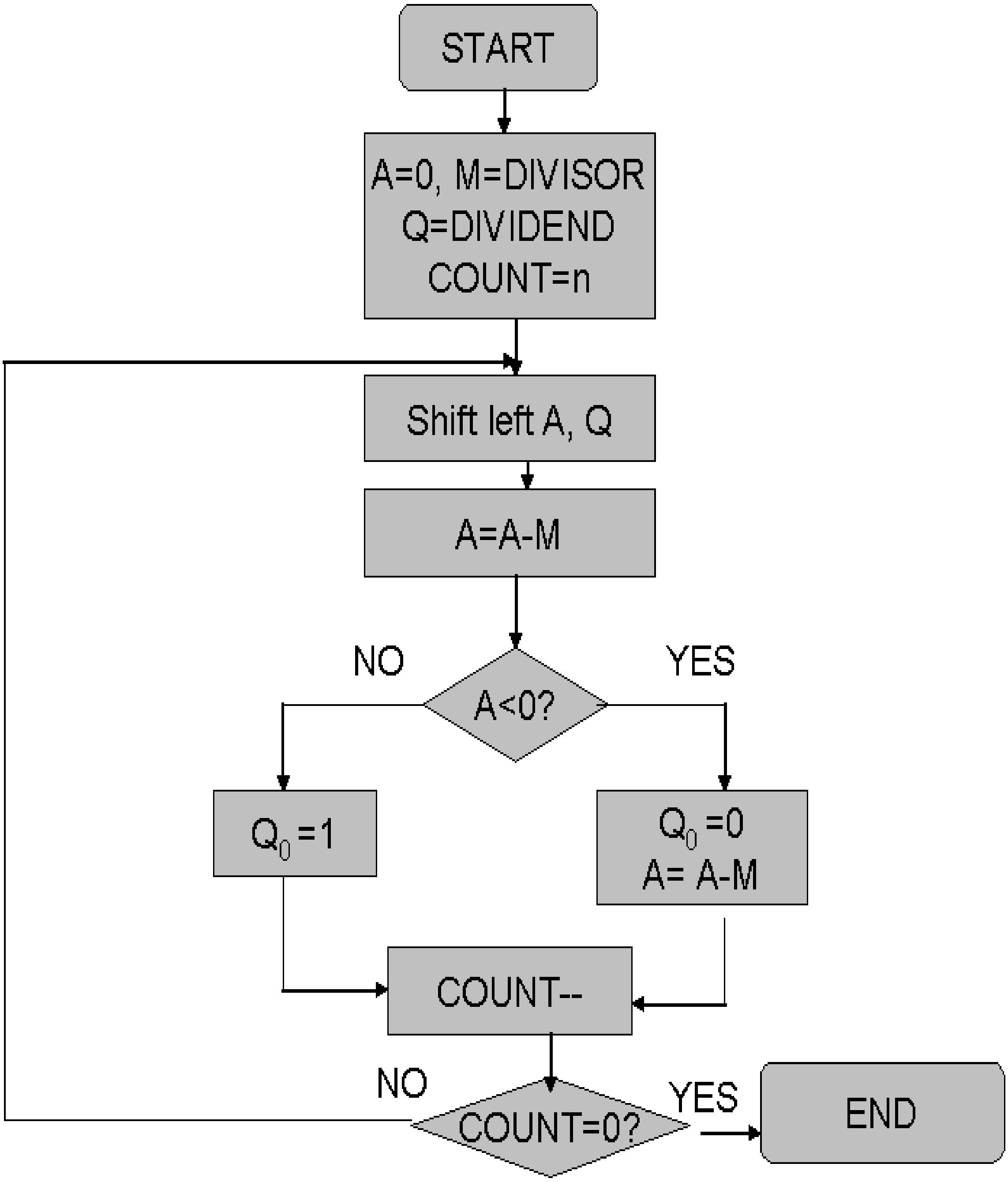
2. William Stallings, "Computer Organization and Architecture: Designing for Performance", Eighth Edition, Pearson.

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.

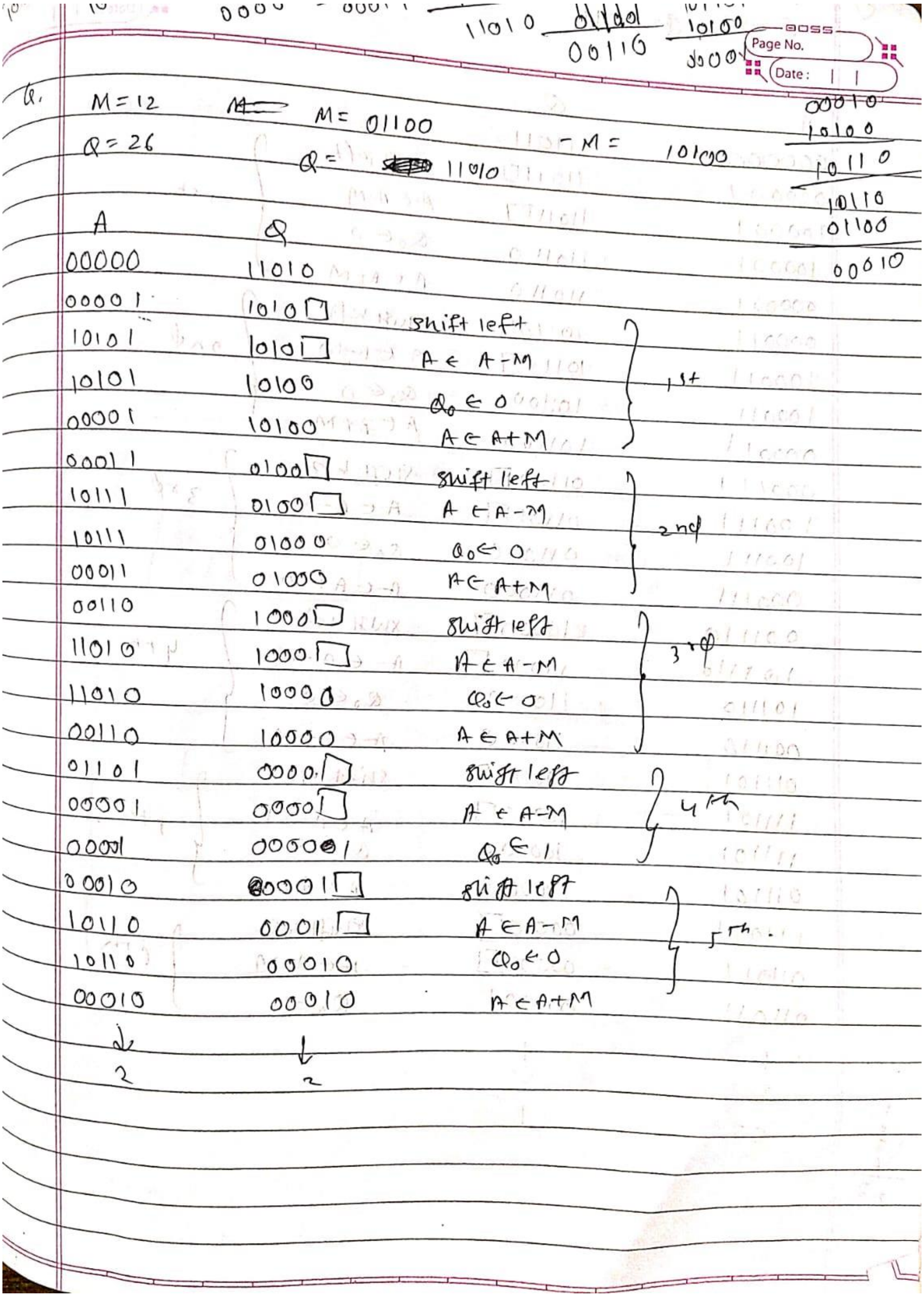
Flowchart for Restoring of Division:



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:-



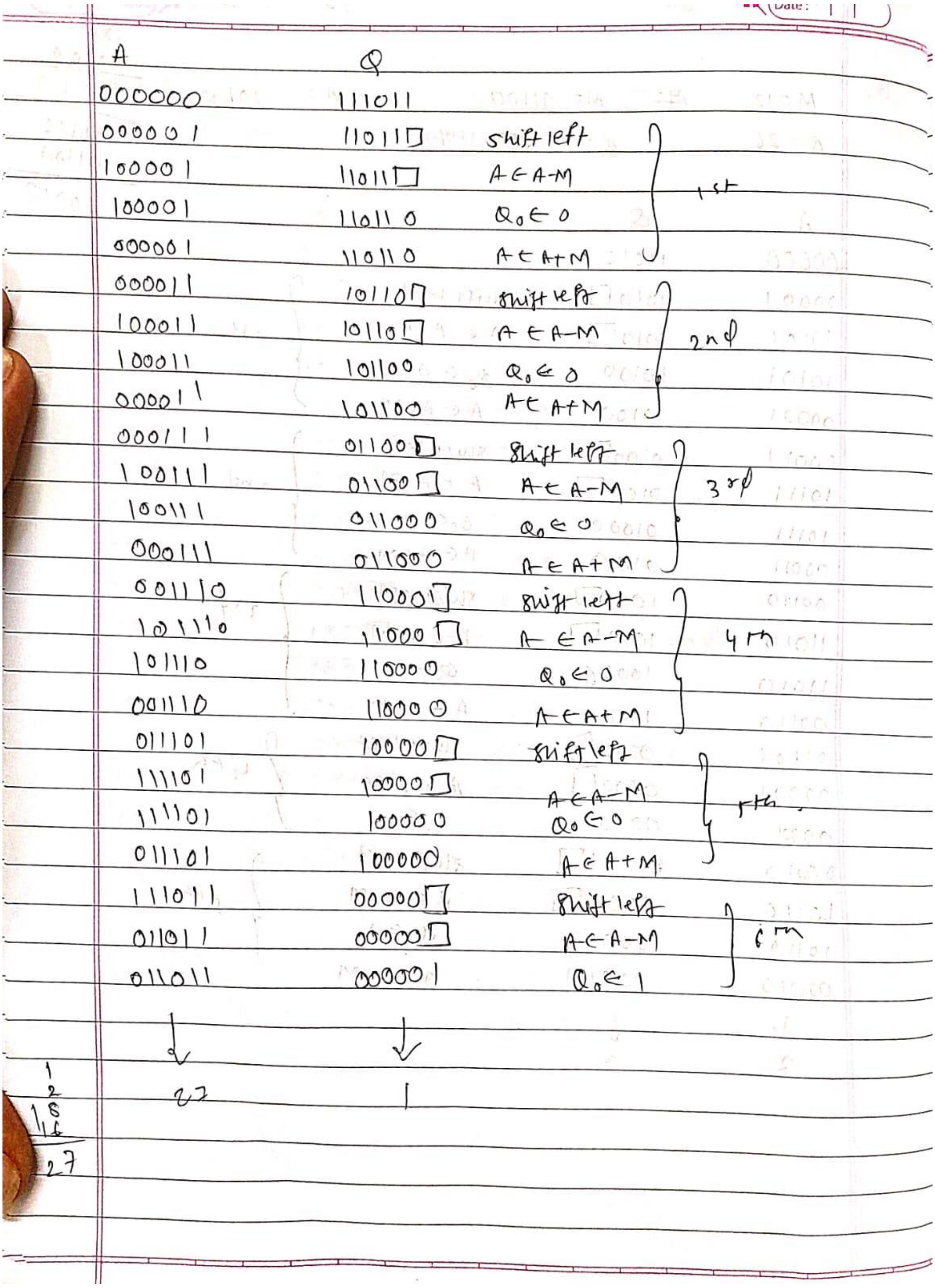
11010 ~~01100~~
00110

10100
00000

Q. M = 12 M = 01100 −M = 10100
Q = 26 Q = 11010

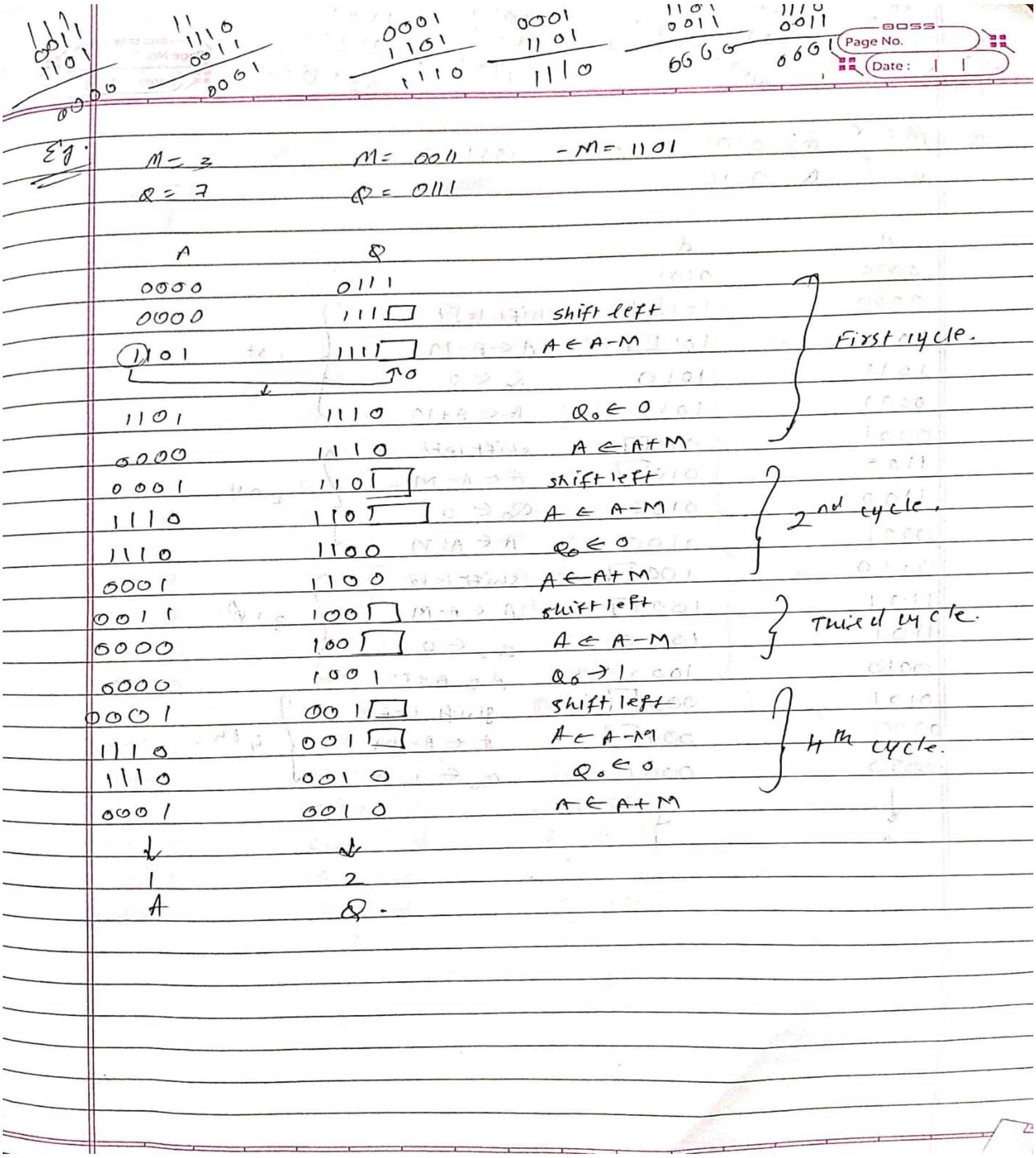
00010
10100
10110
10110
01100
00010

A	Q		
00000	11010		
00001	1010□	shift left	1st
10101	1010□	A ← A − M	
10101	10100	Q_0 ← 0	
00001	10100	A ← A + M	
00011	0100□	shift left	2nd
10111	0100□	A ← A − M	
10111	01000	Q_0 ← 0	
00011	01000	A ← A + M	
00110	1000□	shift left	3rd
11010	1000□	A ← A − M	
11010	10000	Q_0 ← 0	
00110	10000	A ← A + M	
01101	0000□	shift left	4th
00001	0000□	A ← A − M	
00001	00001	Q_0 ← 1	
00010	0001□	shift left	5th
10110	0001□	A ← A − M	
10110	00010	Q_0 ← 0	
00010	00010	A ← A + M	
↓ 2	↓ 2		



A	Q		
000000	111011		
000001	11011□	shift left	1st
100001	11011□	A←A-M	
100001	110110	Q_0←0	
000001	110110	A←A+M	
000011	10110□	shift left	2nd
100011	10110□	A←A-M	
100011	101100	Q_0←0	
000011	101100	A←A+M	
000111	01100□	shift left	3rd
100111	01100□	A←A-M	
100111	011000	Q_0←0	
000111	011000	A←A+M	
001110	11000□	shift left	4th
101110	11000□	A←A-M	
101110	110000	Q_0←0	
001110	110000	A←A+M	
011101	10000□	shift left	5th
111101	10000□	A←A-M	
111101	100000	Q_0←0	
011101	100000	A←A+M	
111011	00000□	shift left	6th
011011	00000□	A←A-M	
011011	000001	Q_0←1	
↓	↓		
27	1		

1
2
8
16
27



```
1111     11       0001    0001    1101    1110
0011     1110     1101    1101    0011    0011
1101     0011     ----    ----    ----    ----
1101     ----     1110    1110    0000    0001
----     0001
0000
```

Eg. M = 3 M = 0011 -M = 1101

Q = 7 Q = 0111

A	Q		
0000	0111		
0000	111□	shift left	
1101	111□	A ← A-M	First cycle.
	↑0		
1101	1110	$Q_0 \leftarrow 0$	
0000	1110	A ← A+M	
0001	110□	shift left	
1110	110□	A ← A-M	2nd cycle.
1110	1100	$Q_0 \leftarrow 0$	
0001	1100	A ← A+M	
0011	100□	shift left	
0000	100□	A ← A-M	Third cycle.
0000	1001	$Q_0 \rightarrow 1$	
0001	001□	shift left	
1110	001□	A ← A-M	4th cycle.
1110	0010	$Q_0 \leftarrow 0$	
0001	0010	A ← A+M	
↓	↓		
1	2		
A	Q.		

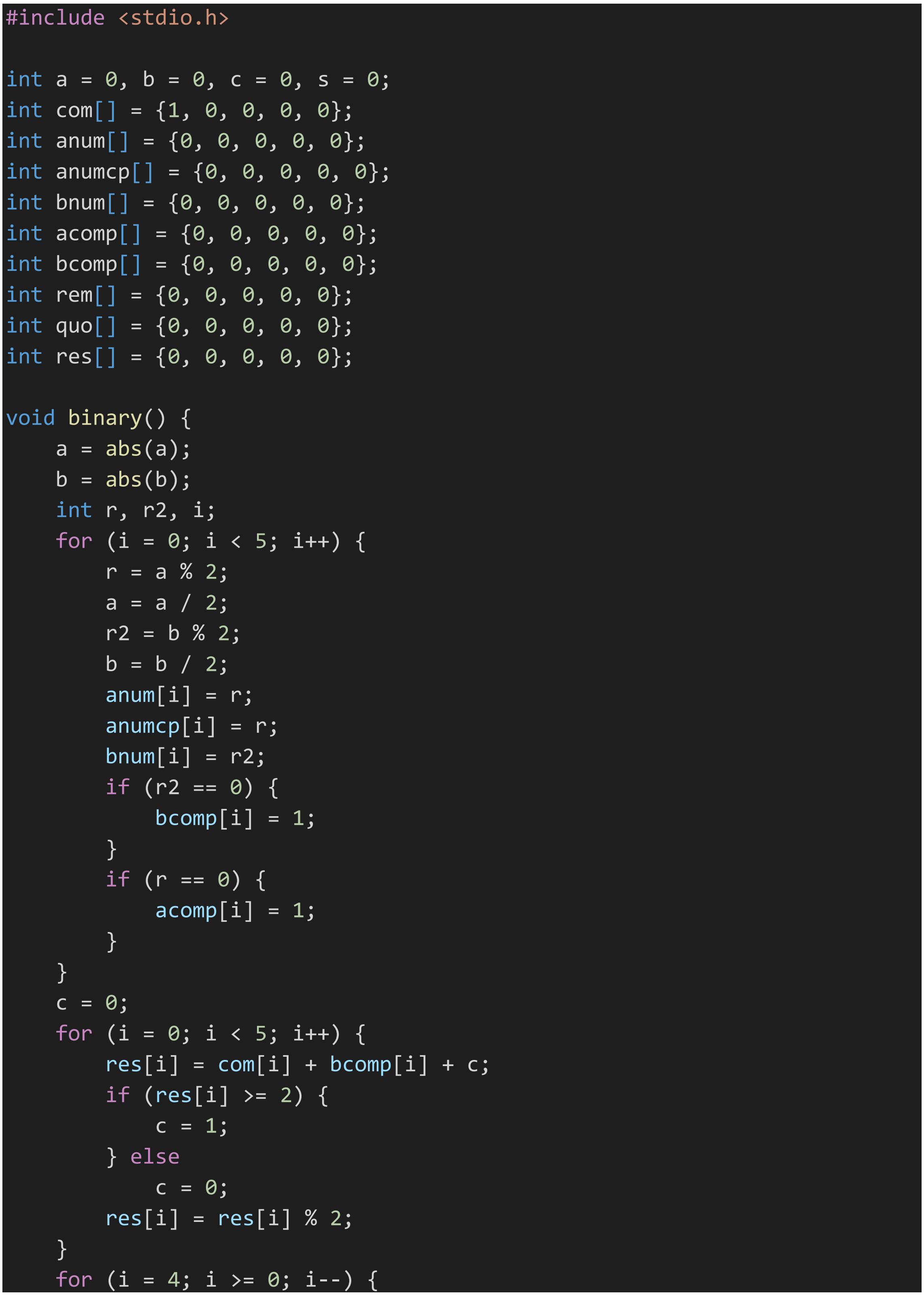
1100 0101 001 | 0101 0000 | 010 1011 0000 | 1011 1100 | 1011 1101 | 0101 0010

Q. M = 5   M = 0101   −M = 1011

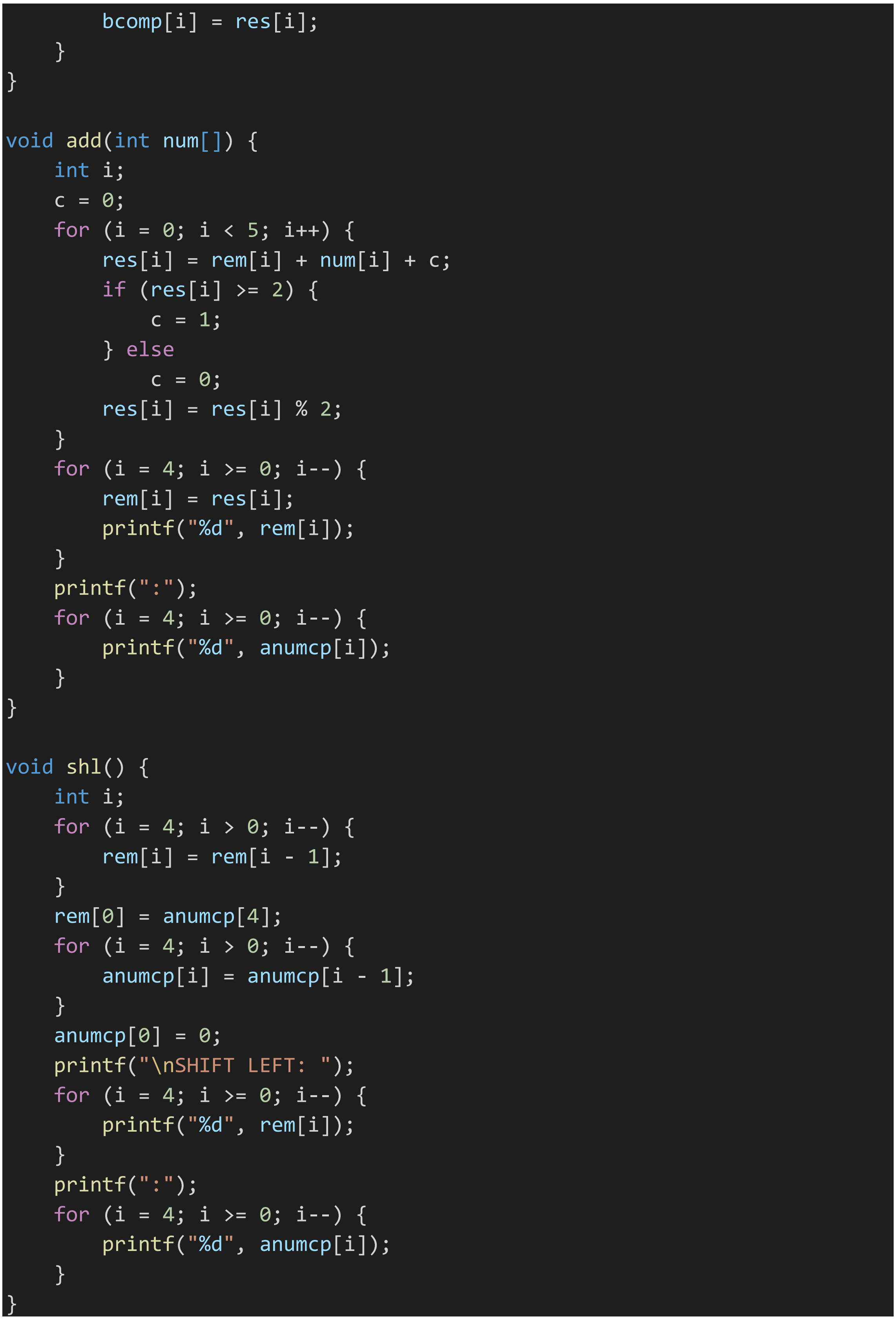
Q = 5   Q = 0101

A	Q		
0000	0101		
0000	101□	shift left	1st
1011	101□	A←A−M	
1011	1010	$Q_0 \leftarrow 0$	
0000	1010	A←A+M	
0001	010□	shift left	2nd
1100	010□	A←A−M	
1100	0100	$Q_0 \leftarrow 0$	
0001	0100	A←A+M	
0010	100□	shift left	3rd
1101	100□	A←A−M	
1101	1000	$Q_0 \leftarrow 0$	
0010	1000	A←A+M	
0101	000□	shift left	4th.
0000	000□	A←A−M	
0000	0001	$Q_0 \leftarrow 1$	
↓ 0	↓ 1		

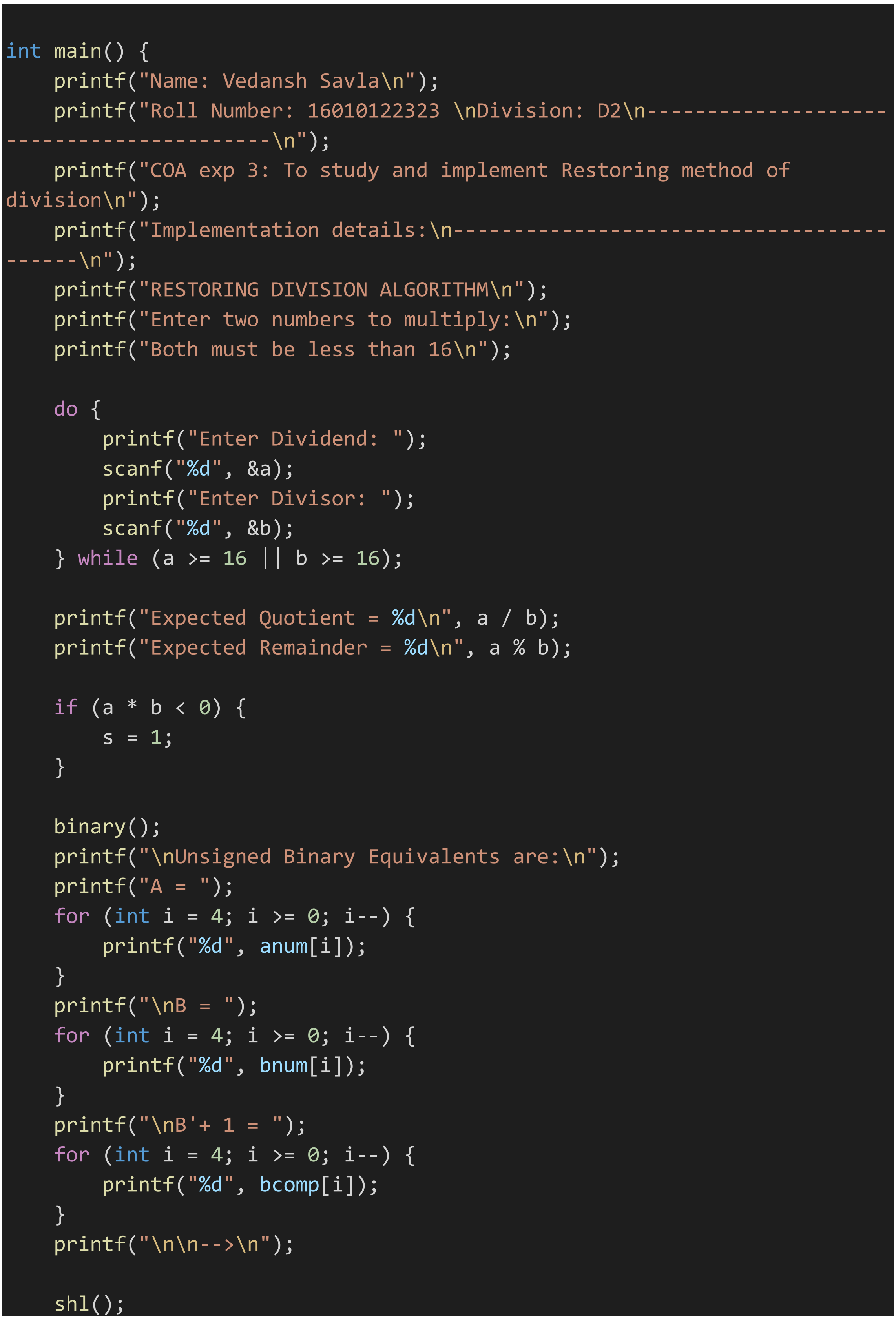
Implementation:



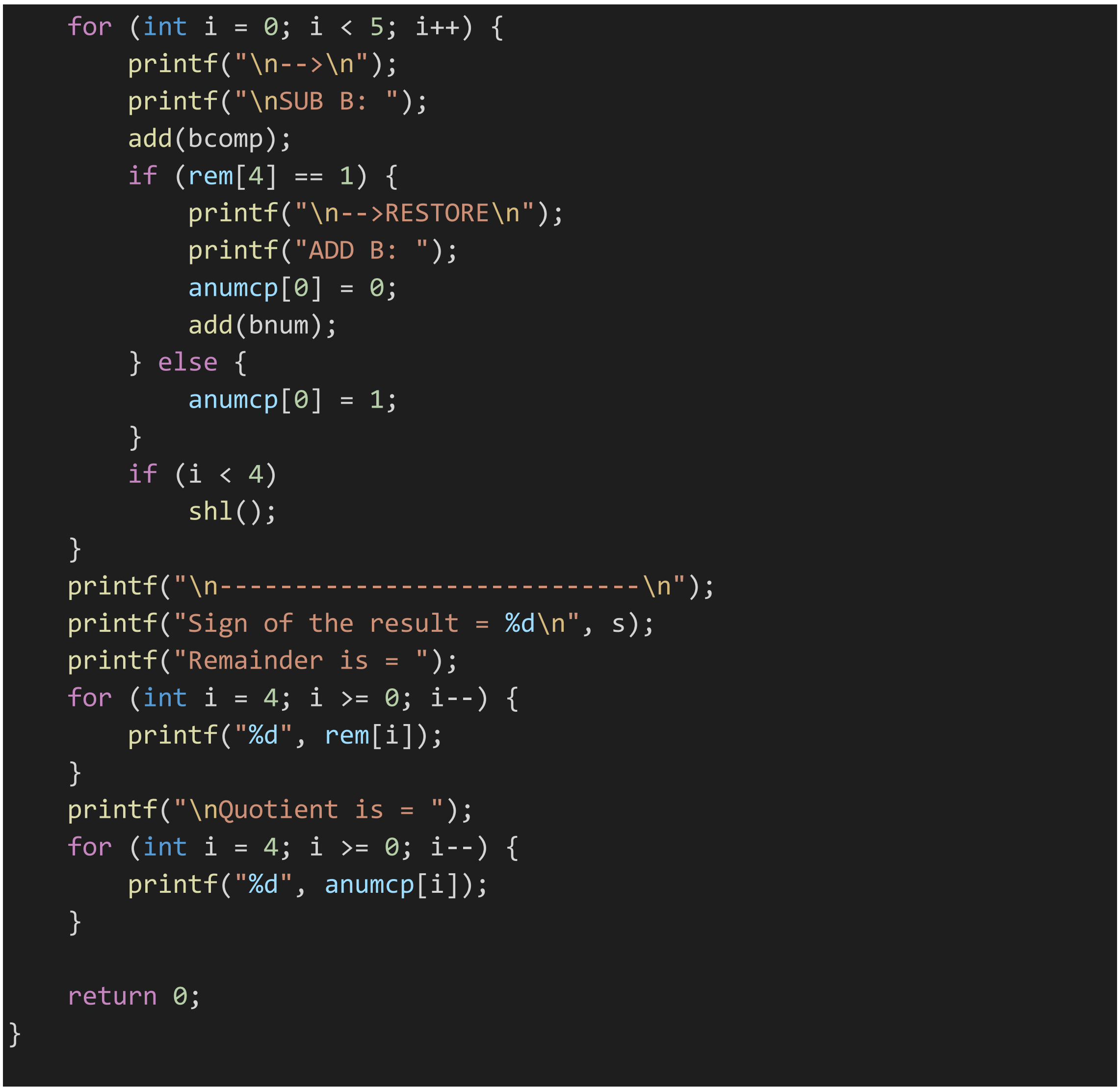
```
#include <stdio.h>

int a = 0, b = 0, c = 0, s = 0;
int com[] = {1, 0, 0, 0, 0};
int anum[] = {0, 0, 0, 0, 0};
int anumcp[] = {0, 0, 0, 0, 0};
int bnum[] = {0, 0, 0, 0, 0};
int acomp[] = {0, 0, 0, 0, 0};
int bcomp[] = {0, 0, 0, 0, 0};
int rem[] = {0, 0, 0, 0, 0};
int quo[] = {0, 0, 0, 0, 0};
int res[] = {0, 0, 0, 0, 0};

void binary() {
    a = abs(a);
    b = abs(b);
    int r, r2, i;
    for (i = 0; i < 5; i++) {
        r = a % 2;
        a = a / 2;
        r2 = b % 2;
        b = b / 2;
        anum[i] = r;
        anumcp[i] = r;
        bnum[i] = r2;
        if (r2 == 0) {
            bcomp[i] = 1;
        }
        if (r == 0) {
            acomp[i] = 1;
        }
    }
    c = 0;
    for (i = 0; i < 5; i++) {
        res[i] = com[i] + bcomp[i] + c;
        if (res[i] >= 2) {
            c = 1;
        } else
            c = 0;
        res[i] = res[i] % 2;
    }
    for (i = 4; i >= 0; i--) {
```



```
        bcomp[i] = res[i];
    }
}

void add(int num[]) {
    int i;
    c = 0;
    for (i = 0; i < 5; i++) {
        res[i] = rem[i] + num[i] + c;
        if (res[i] >= 2) {
            c = 1;
        } else
            c = 0;
        res[i] = res[i] % 2;
    }
    for (i = 4; i >= 0; i--) {
        rem[i] = res[i];
        printf("%d", rem[i]);
    }
    printf(":");
    for (i = 4; i >= 0; i--) {
        printf("%d", anumcp[i]);
    }
}

void shl() {
    int i;
    for (i = 4; i > 0; i--) {
        rem[i] = rem[i - 1];
    }
    rem[0] = anumcp[4];
    for (i = 4; i > 0; i--) {
        anumcp[i] = anumcp[i - 1];
    }
    anumcp[0] = 0;
    printf("\nSHIFT LEFT: ");
    for (i = 4; i >= 0; i--) {
        printf("%d", rem[i]);
    }
    printf(":");
    for (i = 4; i >= 0; i--) {
        printf("%d", anumcp[i]);
    }
}
```

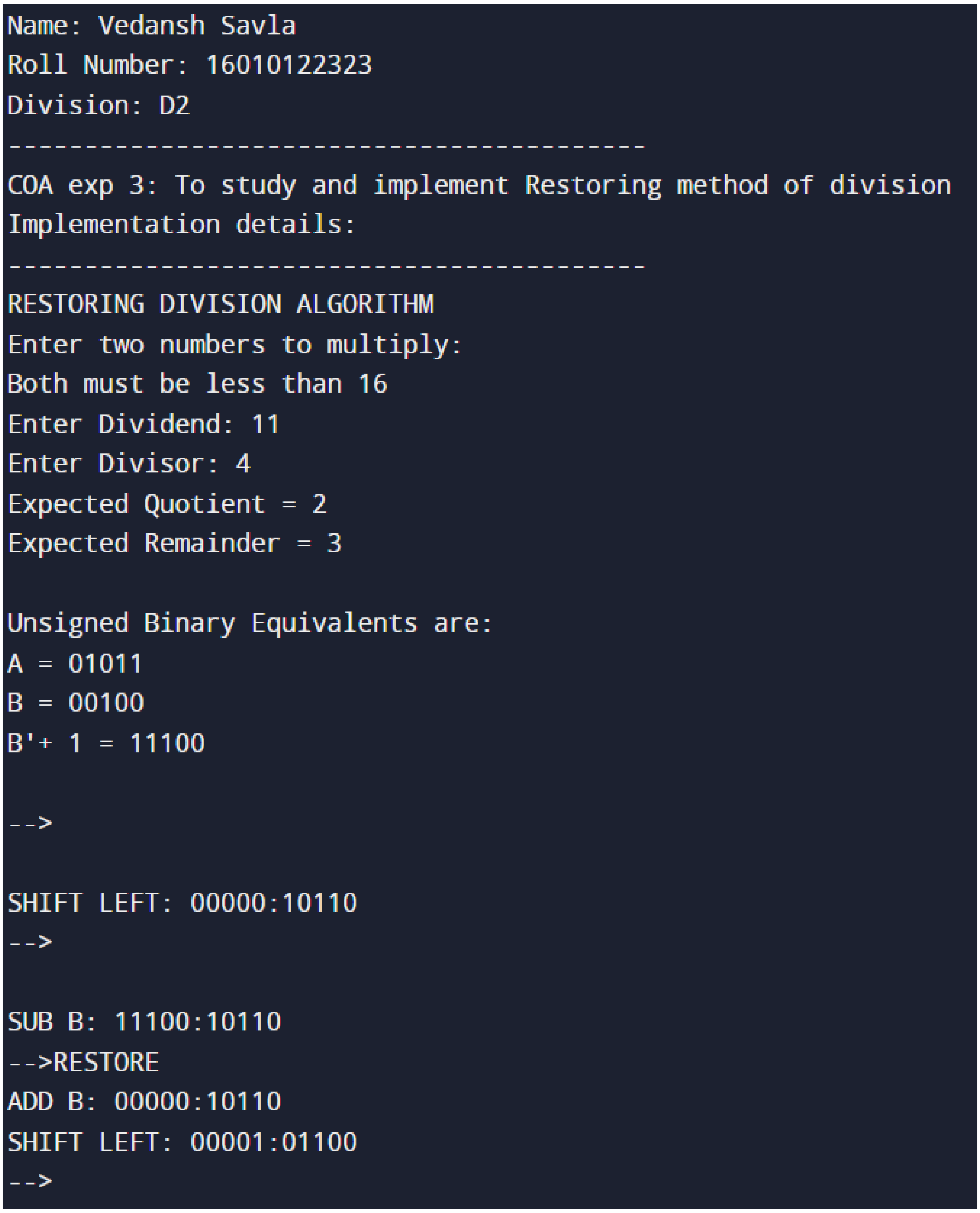


```
int main() {
    printf("Name: Vedansh Savla\n");
    printf("Roll Number: 16010122323 \nDivision: D2\n------------------------------------------\n");
    printf("COA exp 3: To study and implement Restoring method of division\n");
    printf("Implementation details:\n------------------------------------------\n");
    printf("RESTORING DIVISION ALGORITHM\n");
    printf("Enter two numbers to multiply:\n");
    printf("Both must be less than 16\n");

    do {
        printf("Enter Dividend: ");
        scanf("%d", &a);
        printf("Enter Divisor: ");
        scanf("%d", &b);
    } while (a >= 16 || b >= 16);

    printf("Expected Quotient = %d\n", a / b);
    printf("Expected Remainder = %d\n", a % b);

    if (a * b < 0) {
        s = 1;
    }

    binary();
    printf("\nUnsigned Binary Equivalents are:\n");
    printf("A = ");
    for (int i = 4; i >= 0; i--) {
        printf("%d", anum[i]);
    }
    printf("\nB = ");
    for (int i = 4; i >= 0; i--) {
        printf("%d", bnum[i]);
    }
    printf("\nB'+ 1 = ");
    for (int i = 4; i >= 0; i--) {
        printf("%d", bcomp[i]);
    }
    printf("\n\n-->\n");

    shl();
```

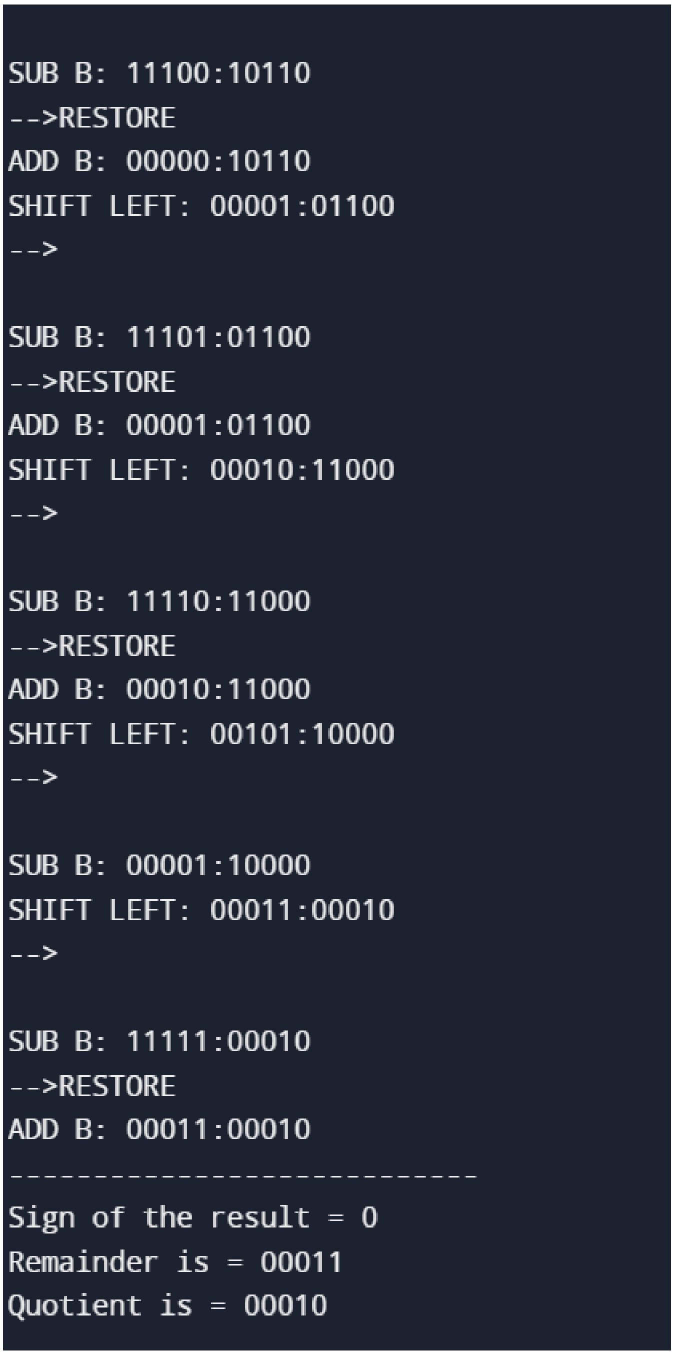


```
    for (int i = 0; i < 5; i++) {
        printf("\n-->\n");
        printf("\nSUB B: ");
        add(bcomp);
        if (rem[4] == 1) {
            printf("\n-->RESTORE\n");
            printf("ADD B: ");
            anumcp[0] = 0;
            add(bnum);
        } else {
            anumcp[0] = 1;
        }
        if (i < 4)
            shl();
    }
    printf("\n----------------------------\n");
    printf("Sign of the result = %d\n", s);
    printf("Remainder is = ");
    for (int i = 4; i >= 0; i--) {
        printf("%d", rem[i]);
    }
    printf("\nQuotient is = ");
    for (int i = 4; i >= 0; i--) {
        printf("%d", anumcp[i]);
    }

    return 0;
}
```

Output:



```
Name: Vedansh Savla
Roll Number: 16010122323
Division: D2
-------------------------------------------
COA exp 3: To study and implement Restoring method of division
Implementation details:
-------------------------------------------
RESTORING DIVISION ALGORITHM
Enter two numbers to multiply:
Both must be less than 16
Enter Dividend: 11
Enter Divisor: 4
Expected Quotient = 2
Expected Remainder = 3

Unsigned Binary Equivalents are:
A = 01011
B = 00100
B'+ 1 = 11100

-->

SHIFT LEFT: 00000:10110
-->

SUB B: 11100:10110
-->RESTORE
ADD B: 00000:10110
SHIFT LEFT: 00001:01100
-->
```



```
SUB B: 11100:10110
-->RESTORE
ADD B: 00000:10110
SHIFT LEFT: 00001:01100
-->

SUB B: 11101:01100
-->RESTORE
ADD B: 00001:01100
SHIFT LEFT: 00010:11000
-->

SUB B: 11110:11000
-->RESTORE
ADD B: 00010:11000
SHIFT LEFT: 00101:10000
-->

SUB B: 00001:10000
SHIFT LEFT: 00011:00010
-->

SUB B: 11111:00010
-->RESTORE
ADD B: 00011:00010
----------------------------
Sign of the result = 0
Remainder is = 00011
Quotient is = 00010
```

Conclusion:

Booths restoring division was implemented successfully.

Post Lab Descriptive Questions

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

The advantage of using non - restoring arithmetic over the standard restoring division is that a test subtraction is not required; the sign bit determines whether an addition or subtraction is used. The disadvantage, though, is that an extra bit must be maintained in the partial remainder to keep track of the sign.

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