Batch: D2 Roll No.: 16010122323

Experiment / assignment / tutorial No.4

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

Signature of the Staff In-charge with date

TITLE : To study and implement Non Restoring method of division

AIM : The basis of algorithm is based on paper and pencil approach and the operation involve 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/COs attained here)
To better understand the non-restoring algorithm and executing it using a programming language. To find the advantage of non-restoring over restoring division.

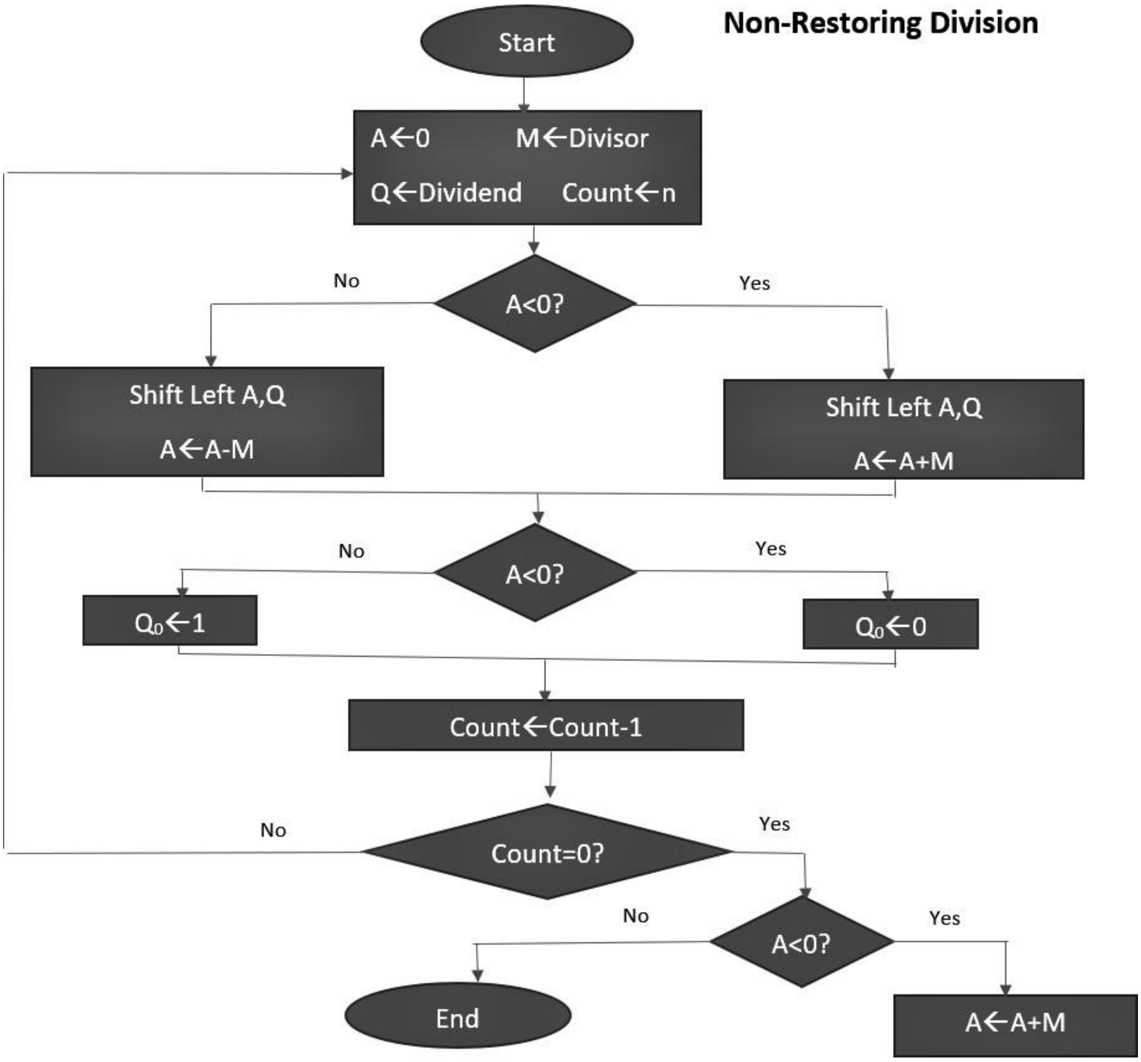
Books/ Journals/ Websites referred:

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.

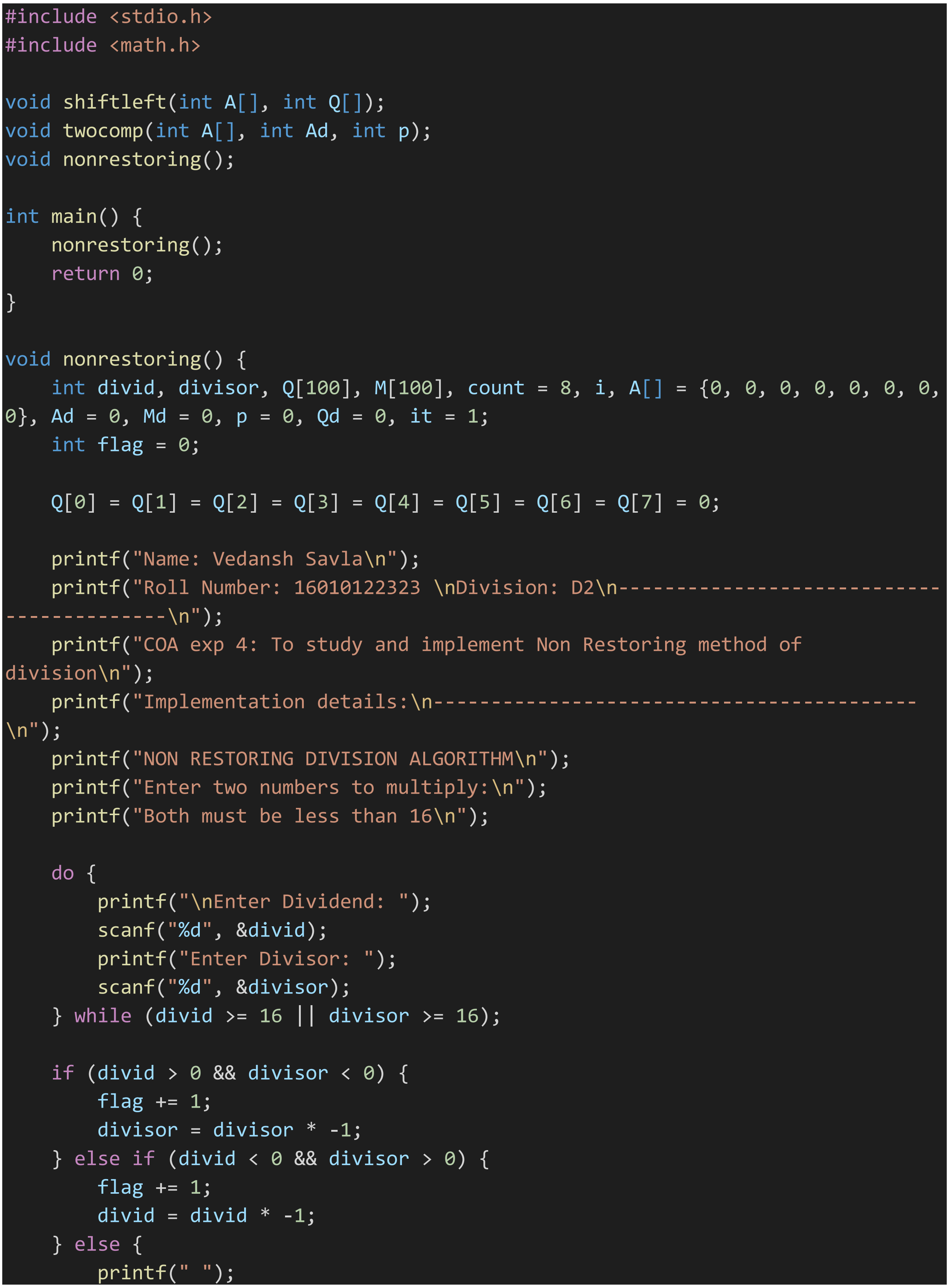
Pre Lab/ Prior Concepts:

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

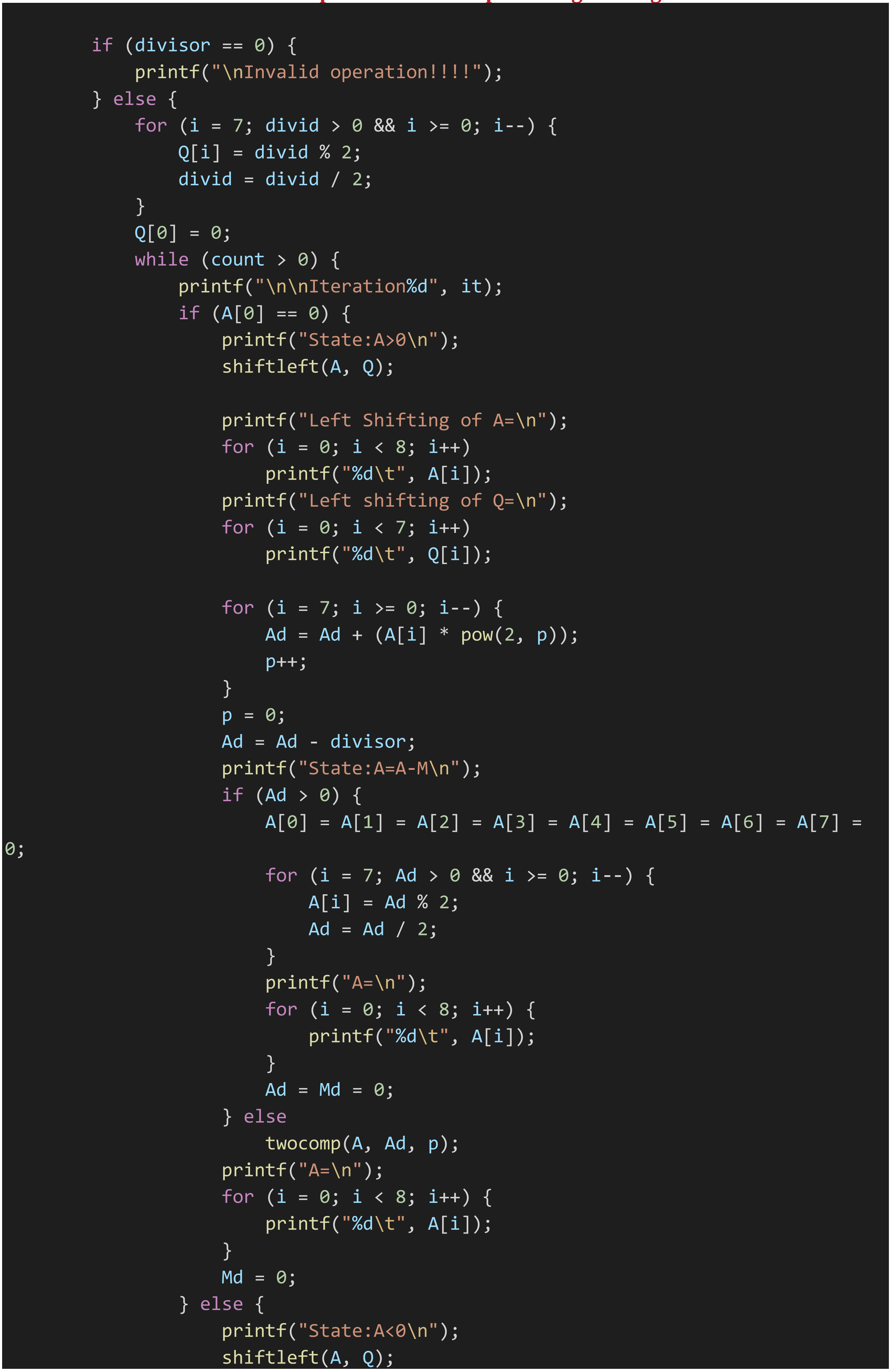
Flowchart for Non-Restoring of Division:



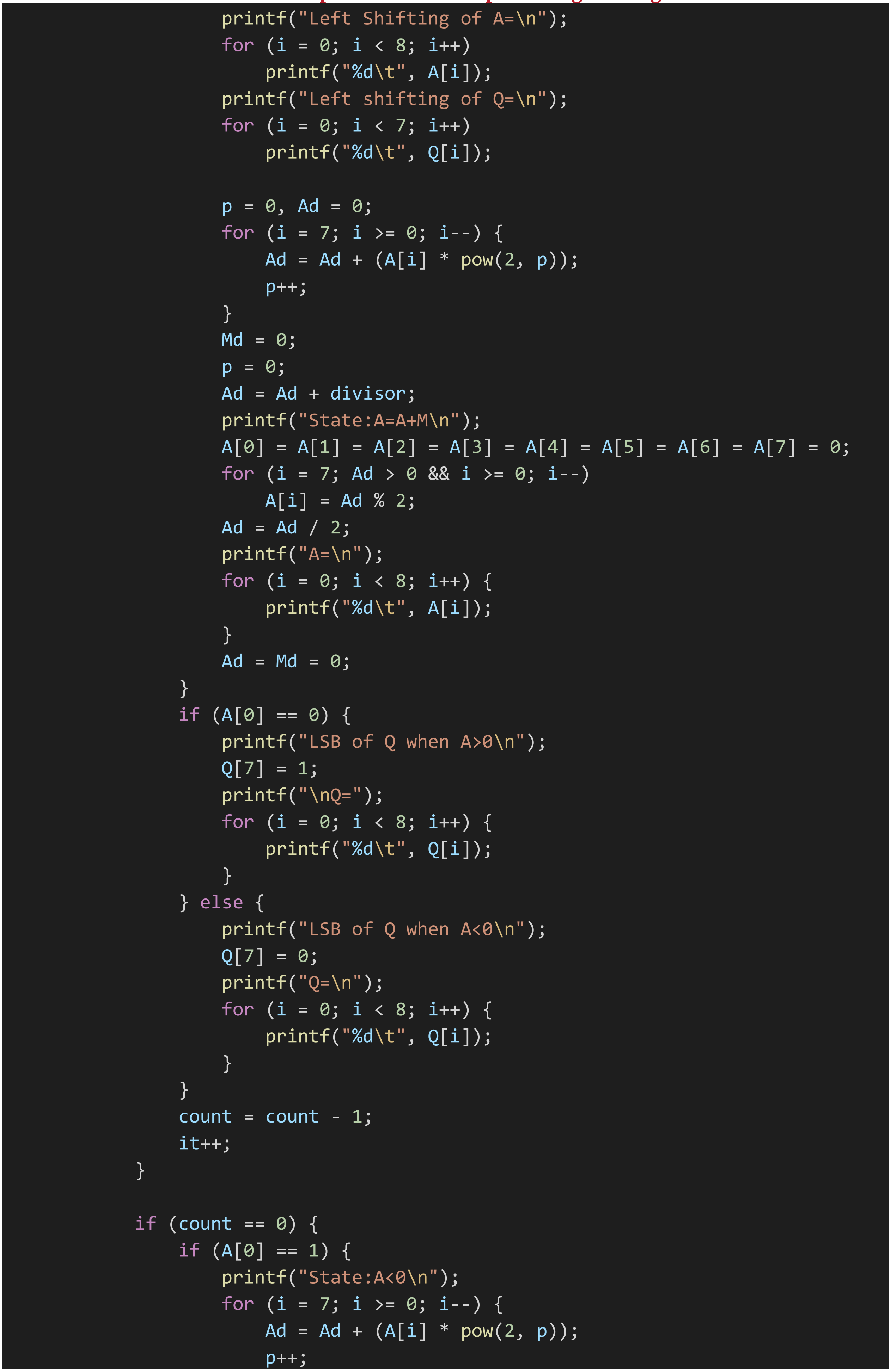
Implementation details:



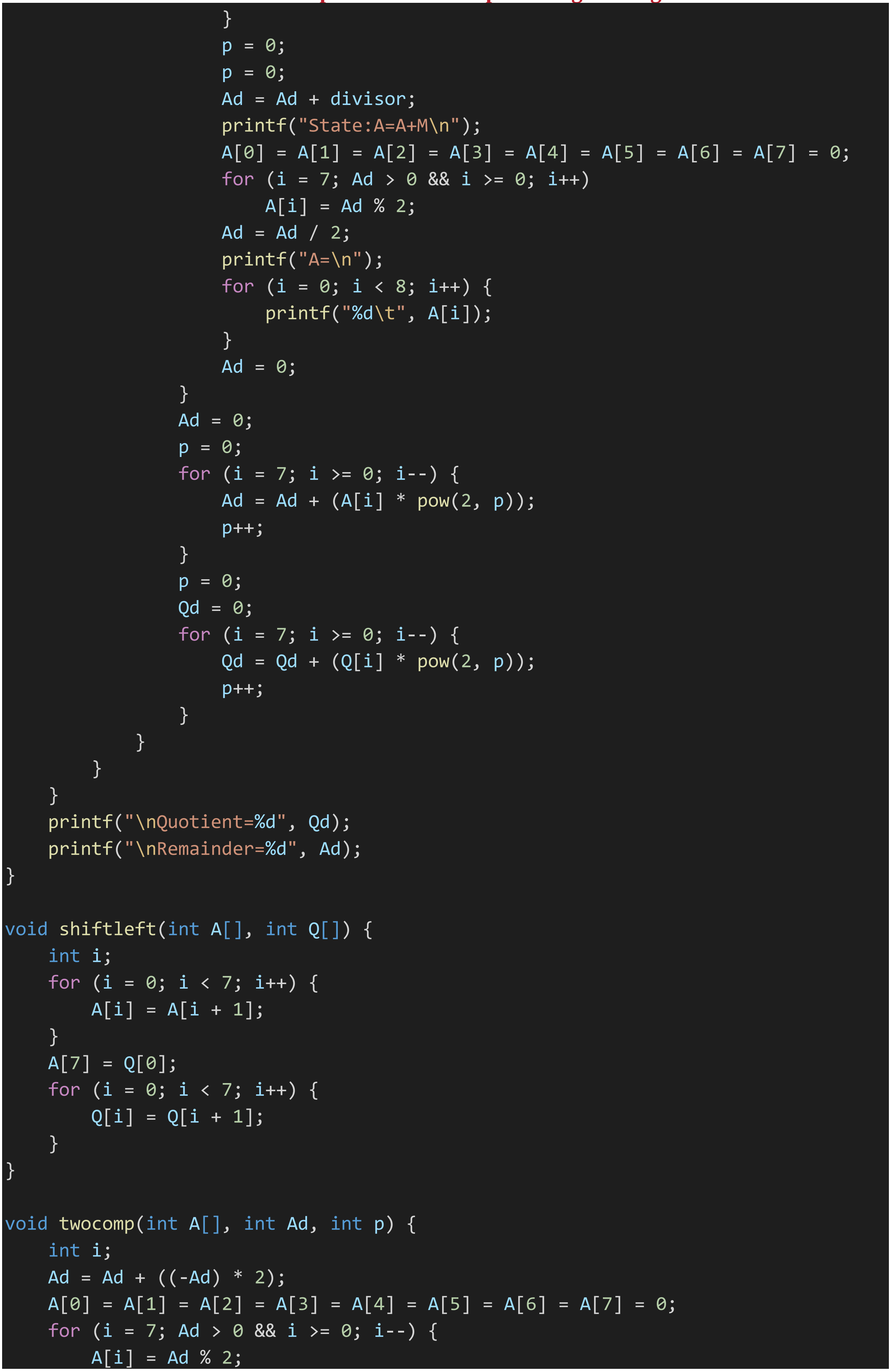
```
#include <stdio.h>
#include <math.h>

void shiftleft(int A[], int Q[]);
void twocomp(int A[], int Ad, int p);
void nonrestoring();

int main() {
    nonrestoring();
    return 0;
}

void nonrestoring() {
    int divid, divisor, Q[100], M[100], count = 8, i, A[] = {0, 0, 0, 0, 0, 0, 0,
0}, Ad = 0, Md = 0, p = 0, Qd = 0, it = 1;
    int flag = 0;

    Q[0] = Q[1] = Q[2] = Q[3] = Q[4] = Q[5] = Q[6] = Q[7] = 0;

    printf("Name: Vedansh Savla\n");
    printf("Roll Number: 16010122323 \nDivision: D2\n----------------------------
--------------\n");
    printf("COA exp 4: To study and implement Non Restoring method of
division\n");
    printf("Implementation details:\n------------------------------------------
\n");
    printf("NON RESTORING DIVISION ALGORITHM\n");
    printf("Enter two numbers to multiply:\n");
    printf("Both must be less than 16\n");

    do {
        printf("\nEnter Dividend: ");
        scanf("%d", &divid);
        printf("Enter Divisor: ");
        scanf("%d", &divisor);
    } while (divid >= 16 || divisor >= 16);

    if (divid > 0 && divisor < 0) {
        flag += 1;
        divisor = divisor * -1;
    } else if (divid < 0 && divisor > 0) {
        flag += 1;
        divid = divid * -1;
    } else {
        printf(" ");
```



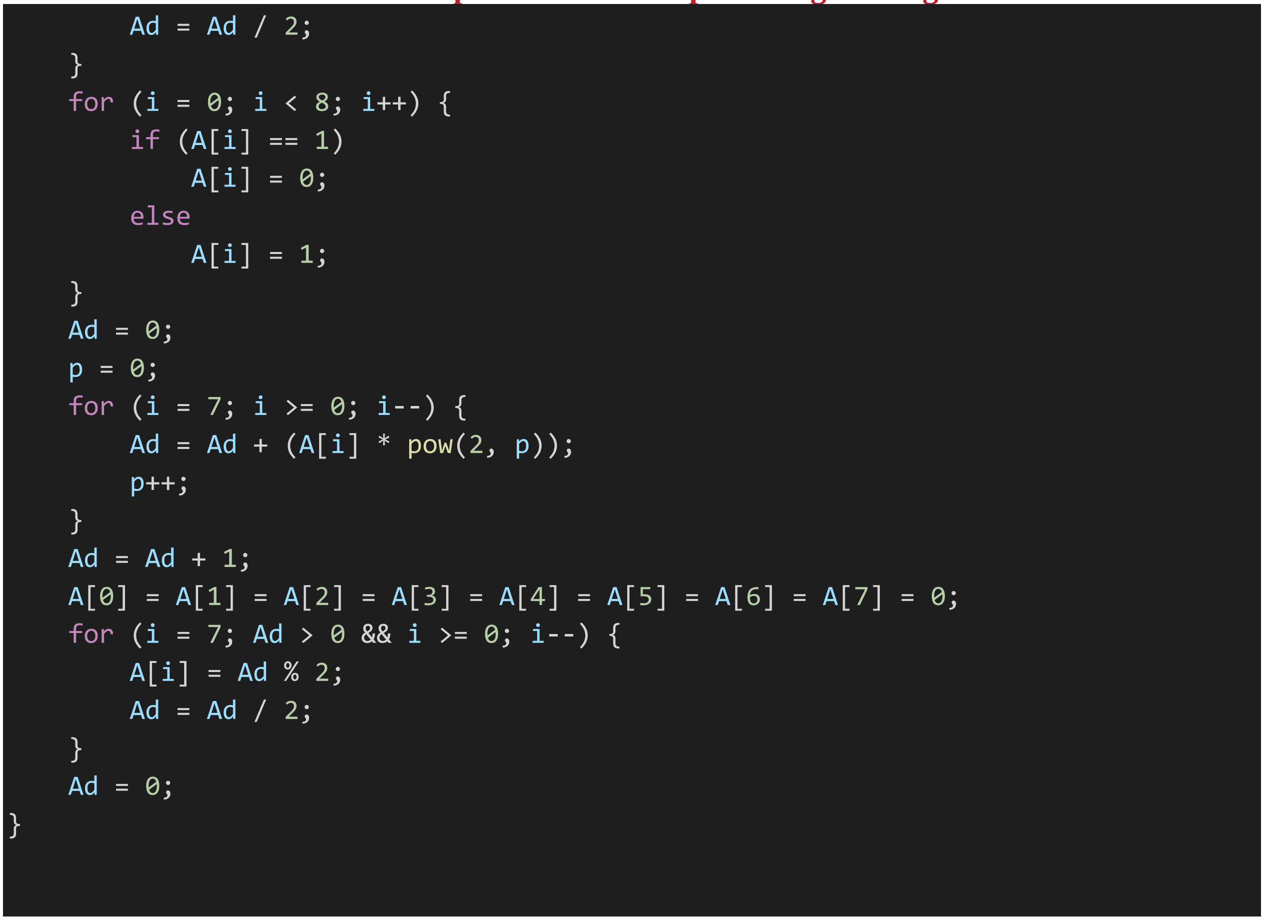
```
        if (divisor == 0) {
            printf("\nInvalid operation!!!!");
        } else {
            for (i = 7; divid > 0 && i >= 0; i--) {
                Q[i] = divid % 2;
                divid = divid / 2;
            }
            Q[0] = 0;
            while (count > 0) {
                printf("\n\nIteration%d", it);
                if (A[0] == 0) {
                    printf("State:A>0\n");
                    shiftleft(A, Q);

                    printf("Left Shifting of A=\n");
                    for (i = 0; i < 8; i++)
                        printf("%d\t", A[i]);
                    printf("Left shifting of Q=\n");
                    for (i = 0; i < 7; i++)
                        printf("%d\t", Q[i]);

                    for (i = 7; i >= 0; i--) {
                        Ad = Ad + (A[i] * pow(2, p));
                        p++;
                    }
                    p = 0;
                    Ad = Ad - divisor;
                    printf("State:A=A-M\n");
                    if (Ad > 0) {
                        A[0] = A[1] = A[2] = A[3] = A[4] = A[5] = A[6] = A[7] =
0;

                        for (i = 7; Ad > 0 && i >= 0; i--) {
                            A[i] = Ad % 2;
                            Ad = Ad / 2;
                        }
                        printf("A=\n");
                        for (i = 0; i < 8; i++) {
                            printf("%d\t", A[i]);
                        }
                        Ad = Md = 0;
                    } else
                        twocomp(A, Ad, p);
                    printf("A=\n");
                    for (i = 0; i < 8; i++) {
                        printf("%d\t", A[i]);
                    }
                    Md = 0;
                } else {
                    printf("State:A<0\n");
                    shiftleft(A, Q);
```



```
            printf("Left Shifting of A=\n");
            for (i = 0; i < 8; i++)
                printf("%d\t", A[i]);
            printf("Left shifting of Q=\n");
            for (i = 0; i < 7; i++)
                printf("%d\t", Q[i]);

            p = 0, Ad = 0;
            for (i = 7; i >= 0; i--) {
                Ad = Ad + (A[i] * pow(2, p));
                p++;
            }
            Md = 0;
            p = 0;
            Ad = Ad + divisor;
            printf("State:A=A+M\n");
            A[0] = A[1] = A[2] = A[3] = A[4] = A[5] = A[6] = A[7] = 0;
            for (i = 7; Ad > 0 && i >= 0; i--)
                A[i] = Ad % 2;
            Ad = Ad / 2;
            printf("A=\n");
            for (i = 0; i < 8; i++) {
                printf("%d\t", A[i]);
            }
            Ad = Md = 0;
        }
        if (A[0] == 0) {
            printf("LSB of Q when A>0\n");
            Q[7] = 1;
            printf("\nQ=");
            for (i = 0; i < 8; i++) {
                printf("%d\t", Q[i]);
            }
        } else {
            printf("LSB of Q when A<0\n");
            Q[7] = 0;
            printf("Q=\n");
            for (i = 0; i < 8; i++) {
                printf("%d\t", Q[i]);
            }
        }
        count = count - 1;
        it++;
    }

    if (count == 0) {
        if (A[0] == 1) {
            printf("State:A<0\n");
            for (i = 7; i >= 0; i--) {
                Ad = Ad + (A[i] * pow(2, p));
                p++;
```

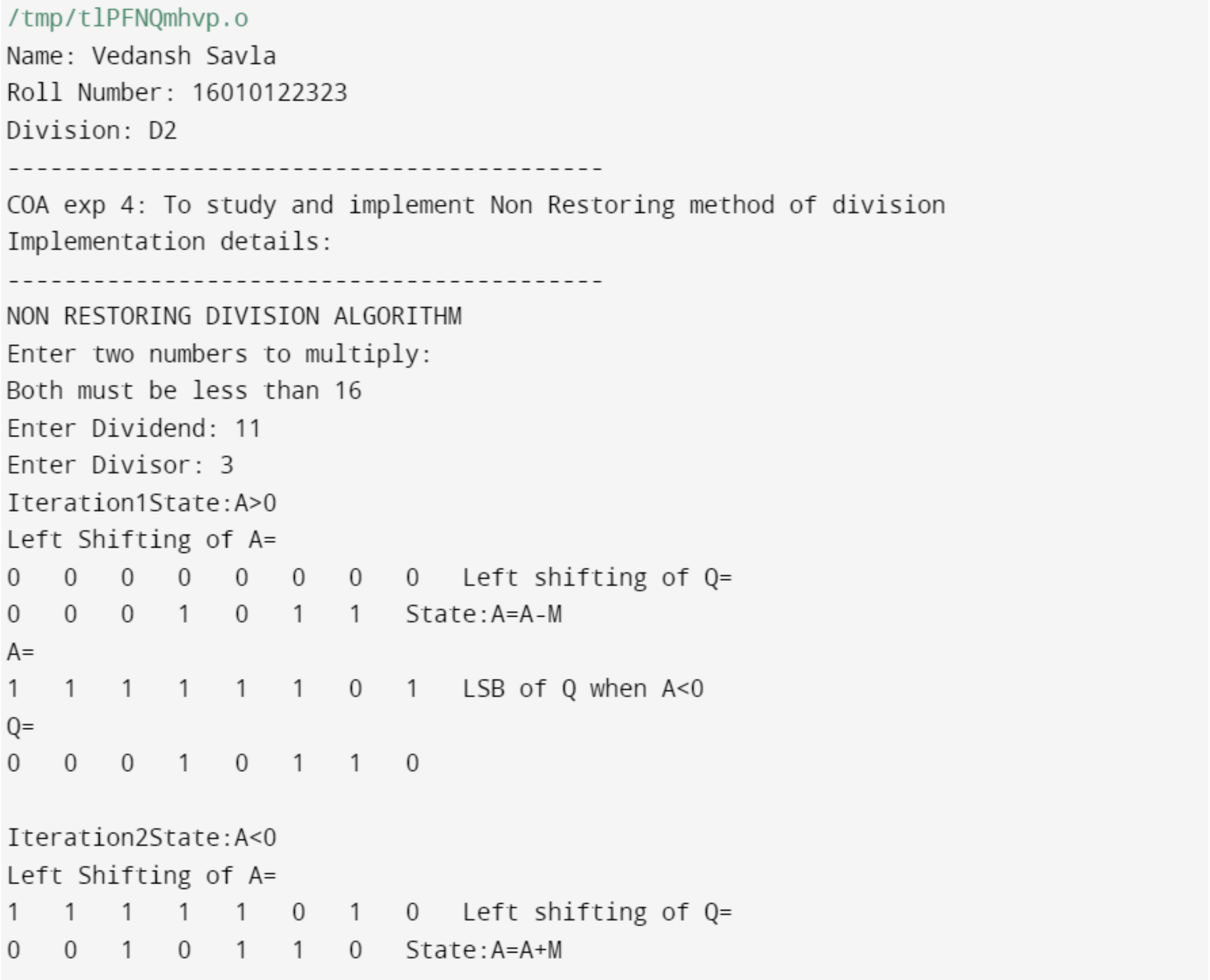


```
                    }
                    p = 0;
                    p = 0;
                    Ad = Ad + divisor;
                    printf("State:A=A+M\n");
                    A[0] = A[1] = A[2] = A[3] = A[4] = A[5] = A[6] = A[7] = 0;
                    for (i = 7; Ad > 0 && i >= 0; i++)
                        A[i] = Ad % 2;
                    Ad = Ad / 2;
                    printf("A=\n");
                    for (i = 0; i < 8; i++) {
                        printf("%d\t", A[i]);
                    }
                    Ad = 0;
                }
                Ad = 0;
                p = 0;
                for (i = 7; i >= 0; i--) {
                    Ad = Ad + (A[i] * pow(2, p));
                    p++;
                }
                p = 0;
                Qd = 0;
                for (i = 7; i >= 0; i--) {
                    Qd = Qd + (Q[i] * pow(2, p));
                    p++;
                }
            }
        }
    }
    printf("\nQuotient=%d", Qd);
    printf("\nRemainder=%d", Ad);
}

void shiftleft(int A[], int Q[]) {
    int i;
    for (i = 0; i < 7; i++) {
        A[i] = A[i + 1];
    }
    A[7] = Q[0];
    for (i = 0; i < 7; i++) {
        Q[i] = Q[i + 1];
    }
}

void twocomp(int A[], int Ad, int p) {
    int i;
    Ad = Ad + ((-Ad) * 2);
    A[0] = A[1] = A[2] = A[3] = A[4] = A[5] = A[6] = A[7] = 0;
    for (i = 7; Ad > 0 && i >= 0; i--) {
        A[i] = Ad % 2;
```

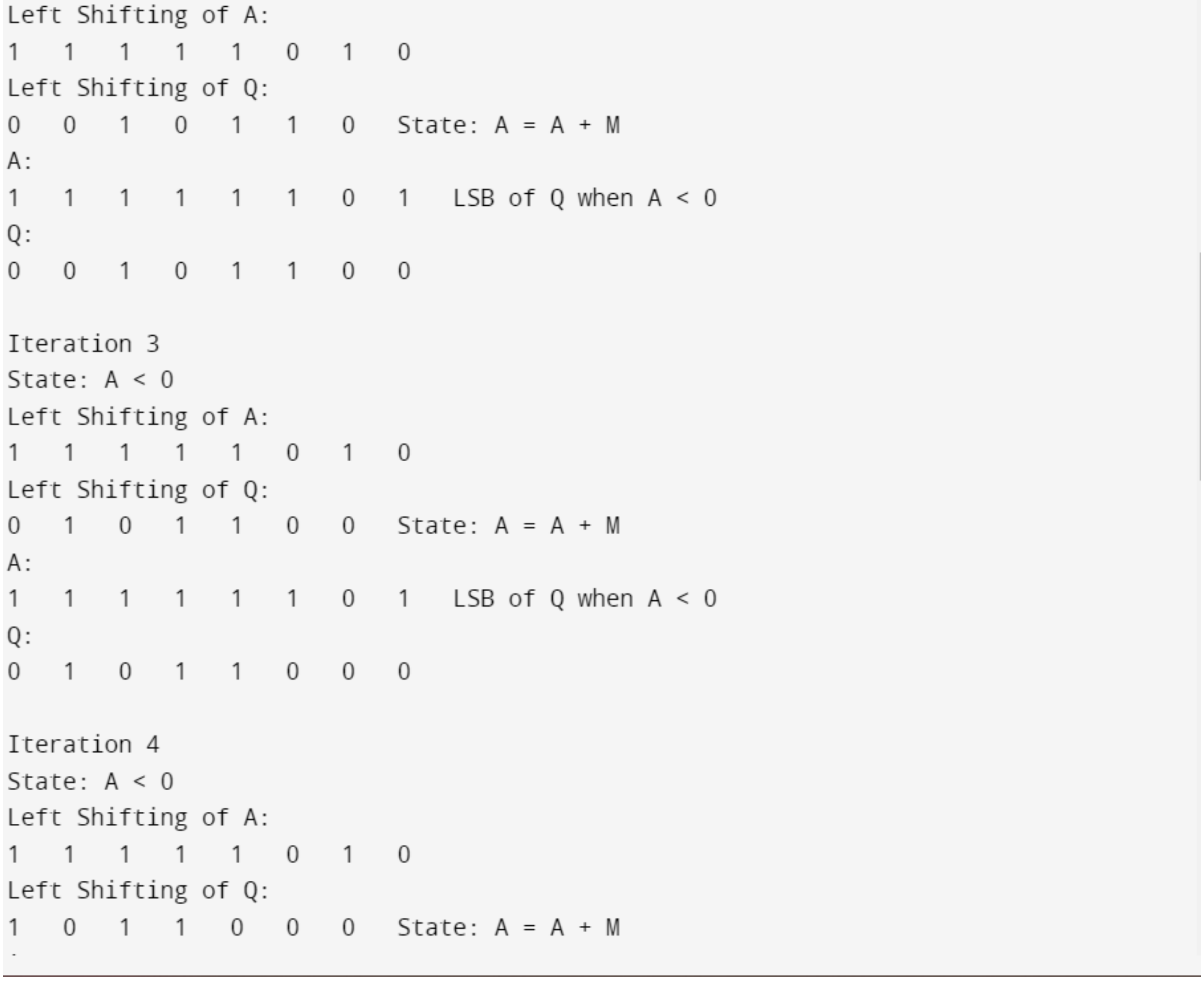


```
        Ad = Ad / 2;
    }
    for (i = 0; i < 8; i++) {
        if (A[i] == 1)
            A[i] = 0;
        else
            A[i] = 1;
    }
    Ad = 0;
    p = 0;
    for (i = 7; i >= 0; i--) {
        Ad = Ad + (A[i] * pow(2, p));
        p++;
    }
    Ad = Ad + 1;
    A[0] = A[1] = A[2] = A[3] = A[4] = A[5] = A[6] = A[7] = 0;
    for (i = 7; Ad > 0 && i >= 0; i--) {
        A[i] = Ad % 2;
        Ad = Ad / 2;
    }
    Ad = 0;
}
```

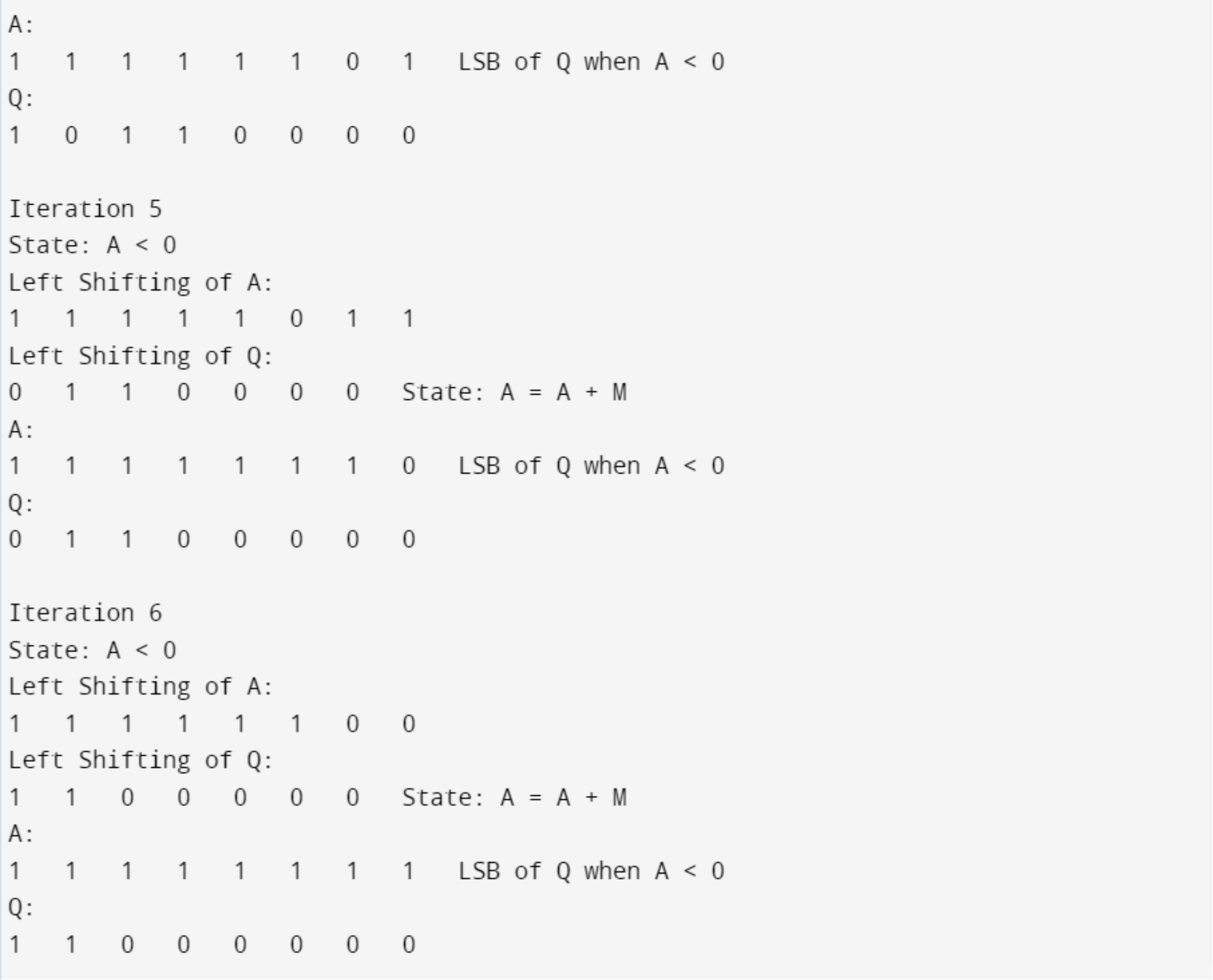
Output:



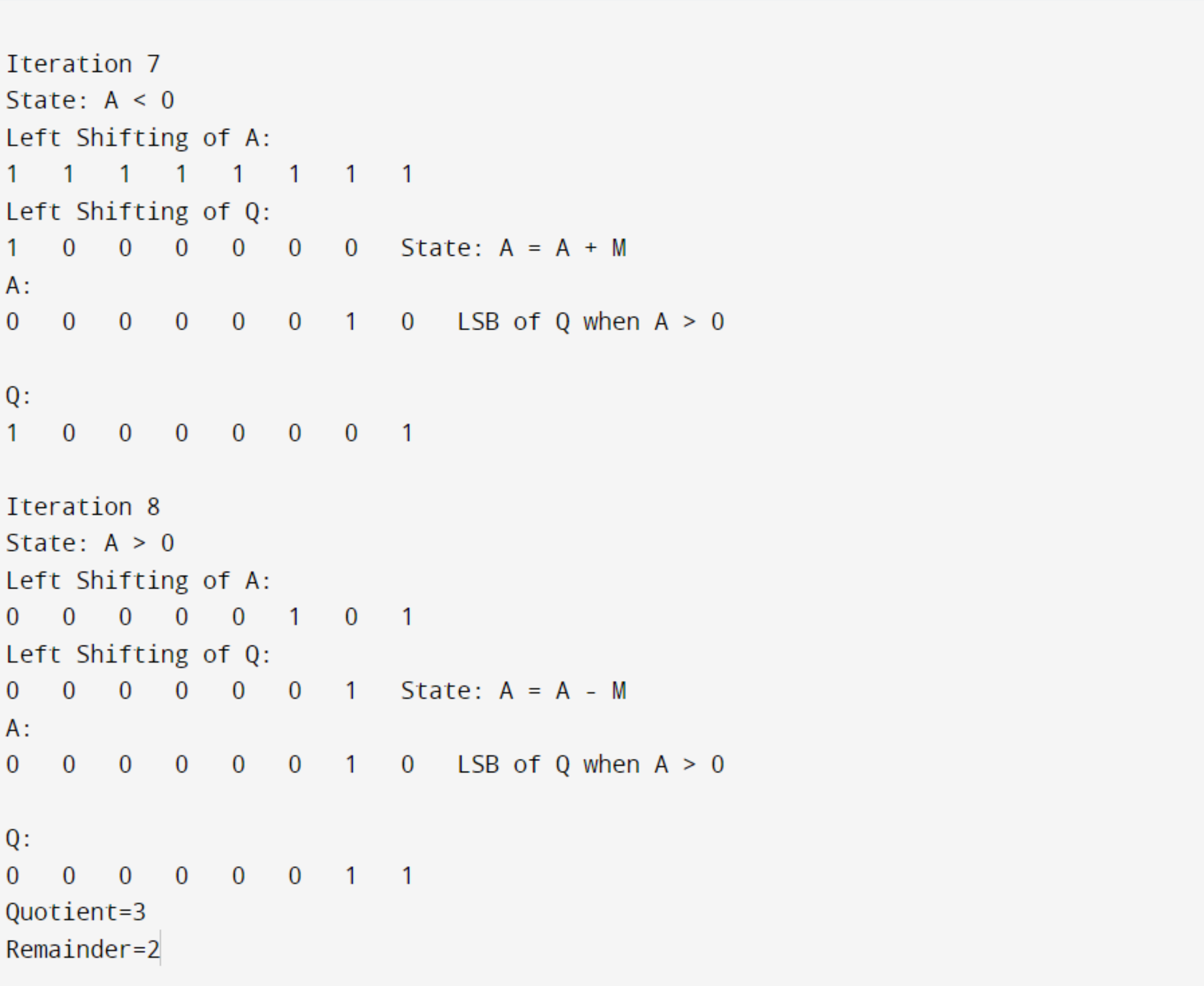
```
/tmp/tlPFNQmhvp.o
Name: Vedansh Savla
Roll Number: 16010122323
Division: D2
------------------------------------------
COA exp 4: To study and implement Non Restoring method of division
Implementation details:
------------------------------------------
NON RESTORING DIVISION ALGORITHM
Enter two numbers to multiply:
Both must be less than 16
Enter Dividend: 11
Enter Divisor: 3
Iteration1State:A>0
Left Shifting of A=
0   0   0   0   0   0   0   0   Left shifting of Q=
0   0   0   1   0   1   1   State:A=A-M
A=
1   1   1   1   1   1   0   1   LSB of Q when A<0
Q=
0   0   0   1   0   1   1   0

Iteration2State:A<0
Left Shifting of A=
1   1   1   1   1   0   1   0   Left shifting of Q=
0   0   1   0   1   1   0   State:A=A+M
```



```
Left Shifting of A:
1   1   1   1   1   0   1   0
Left Shifting of Q:
0   0   1   0   1   1   0   State: A = A + M
A:
1   1   1   1   1   1   0   1   LSB of Q when A < 0
Q:
0   0   1   0   1   1   0   0

Iteration 3
State: A < 0
Left Shifting of A:
1   1   1   1   1   0   1   0
Left Shifting of Q:
0   1   0   1   1   0   0   State: A = A + M
A:
1   1   1   1   1   1   0   1   LSB of Q when A < 0
Q:
0   1   0   1   1   0   0   0

Iteration 4
State: A < 0
Left Shifting of A:
1   1   1   1   1   0   1   0
Left Shifting of Q:
1   0   1   1   0   0   0   State: A = A + M
```

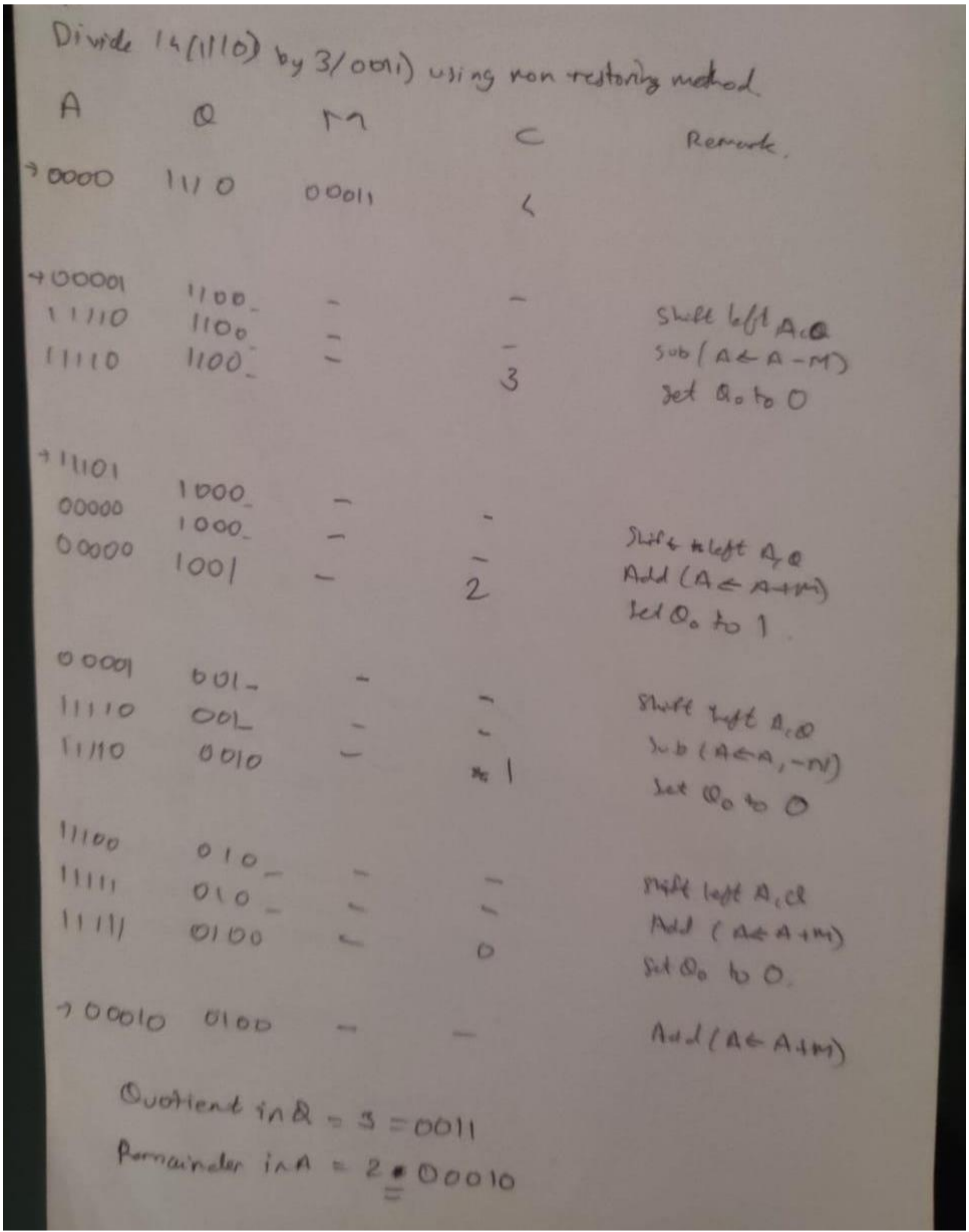


```
A:
1   1   1   1   1   1   0   1   LSB of Q when A < 0
Q:
1   0   1   1   0   0   0   0

Iteration 5
State: A < 0
Left Shifting of A:
1   1   1   1   1   0   1   1
Left Shifting of Q:
0   1   1   0   0   0   0   State: A = A + M
A:
1   1   1   1   1   1   1   0   LSB of Q when A < 0
Q:
0   1   1   0   0   0   0   0

Iteration 6
State: A < 0
Left Shifting of A:
1   1   1   1   1   1   0   0
Left Shifting of Q:
1   1   0   0   0   0   0   State: A = A + M
A:
1   1   1   1   1   1   1   1   LSB of Q when A < 0
Q:
1   1   0   0   0   0   0   0
```



```
Iteration 7
State: A < 0
Left Shifting of A:
1   1   1   1   1   1   1   1
Left Shifting of Q:
1   0   0   0   0   0   0   State: A = A + M
A:
0   0   0   0   0   0   1   0   LSB of Q when A > 0

Q:
1   0   0   0   0   0   0   1

Iteration 8
State: A > 0
Left Shifting of A:
0   0   0   0   0   1   0   1
Left Shifting of Q:
0   0   0   0   0   0   1   State: A = A - M
A:
0   0   0   0   0   0   1   0   LSB of Q when A > 0

Q:
0   0   0   0   0   0   1   1
Quotient=3
Remainder=2
```

Example: (Handwritten solved problem needs to uploaded):



Divide 14(1110) by 3/0011) using non restoring method

A	Q	M	C	Remark.
→ 0000	1110	00011	4	
→ 00001	1100_	–	–	
11110	1100	–	–	Shift left A,Q
11110	1100_	–	–	Sub (A ← A – M)
			3	Set Q₀ to 0
→ 11101	1000_	–		
00000	1000_	–	–	Shift ~~to~~ left A,Q
00000	1001	–	–	Add (A ← A+M)
			2	Set Q₀ to 1
00001	001_	–		
11110	001_	–	–	Shift left A,Q
11110	0010	–	–	Sub (A ← A, –M)
			1	Set Q₀ to 0
11100	010_	–		
11111	010_	–	–	Shift left A,Q
11111	0100	–	–	Add (A ← A+M)
			0	Set Q₀ to 0.
→ 00010	0100	–	–	Add (A ← A+M)

Quotient in Q = 3 = 0011

Remainder in A = 2 = 00010

Conclusion:

In this experiment, Non-Restoring Division Algorithm is executed with the help of C programming.

The advantage of Non-Restoring Division over Restoring Division is better understood.

Post Lab Descriptive Questions

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

Non-restoring division uses the digit set {−1, 1} for the quotient digits instead of {0, 1}. Non-Restoring Division when implemented in hardware, there is only one decision and addition/subtraction per quotient bit; there is no restoring step after the subtraction, which potentially cuts down the numbers of operations by up to half and lets it be executed faster**.**

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.