ASSIGNMENT 4: LOGISIM

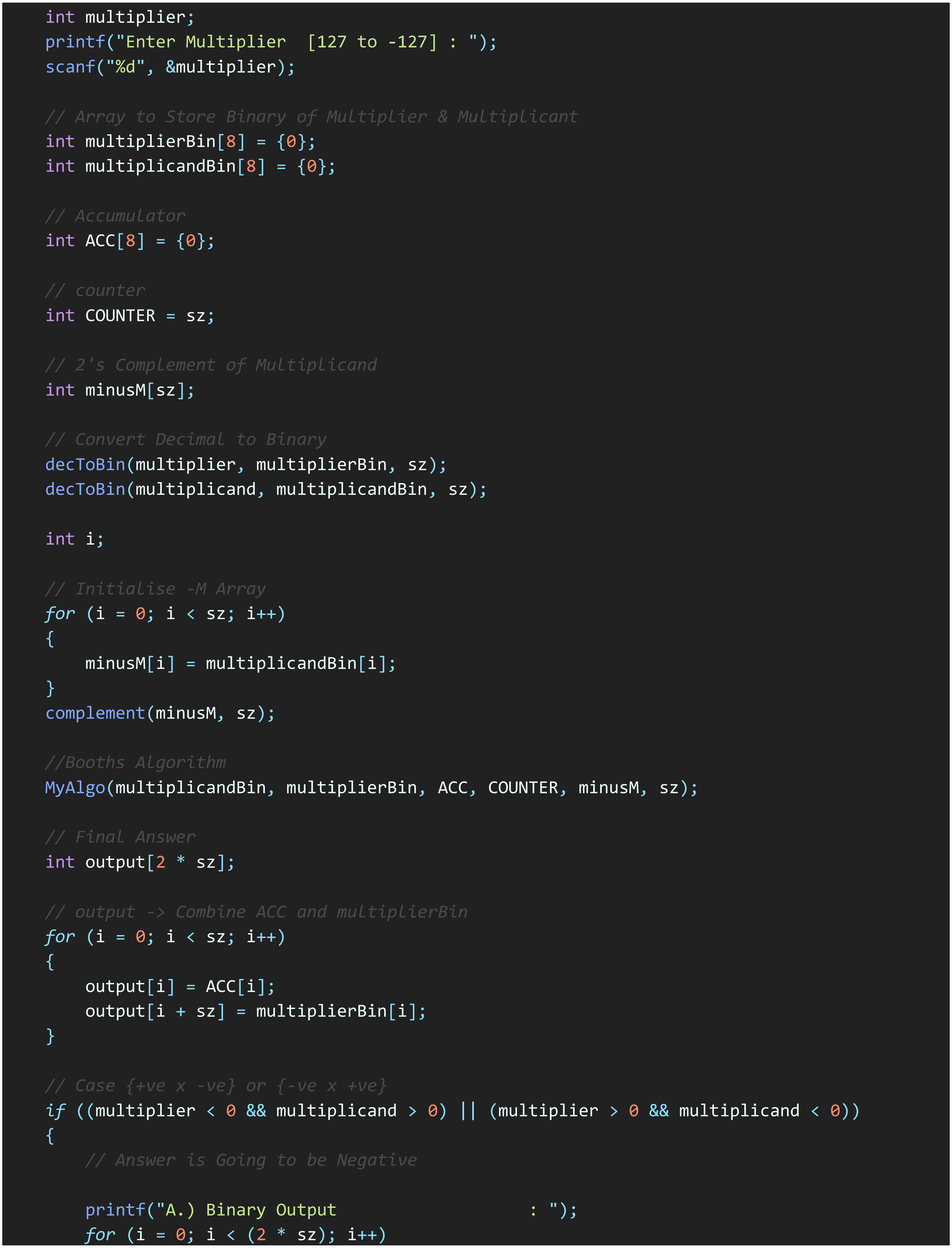
U19CS012 [D-12]

Q1.) Implement Booth's Algorithm in C. Show the output of multiplication of your implementation for the following cases:

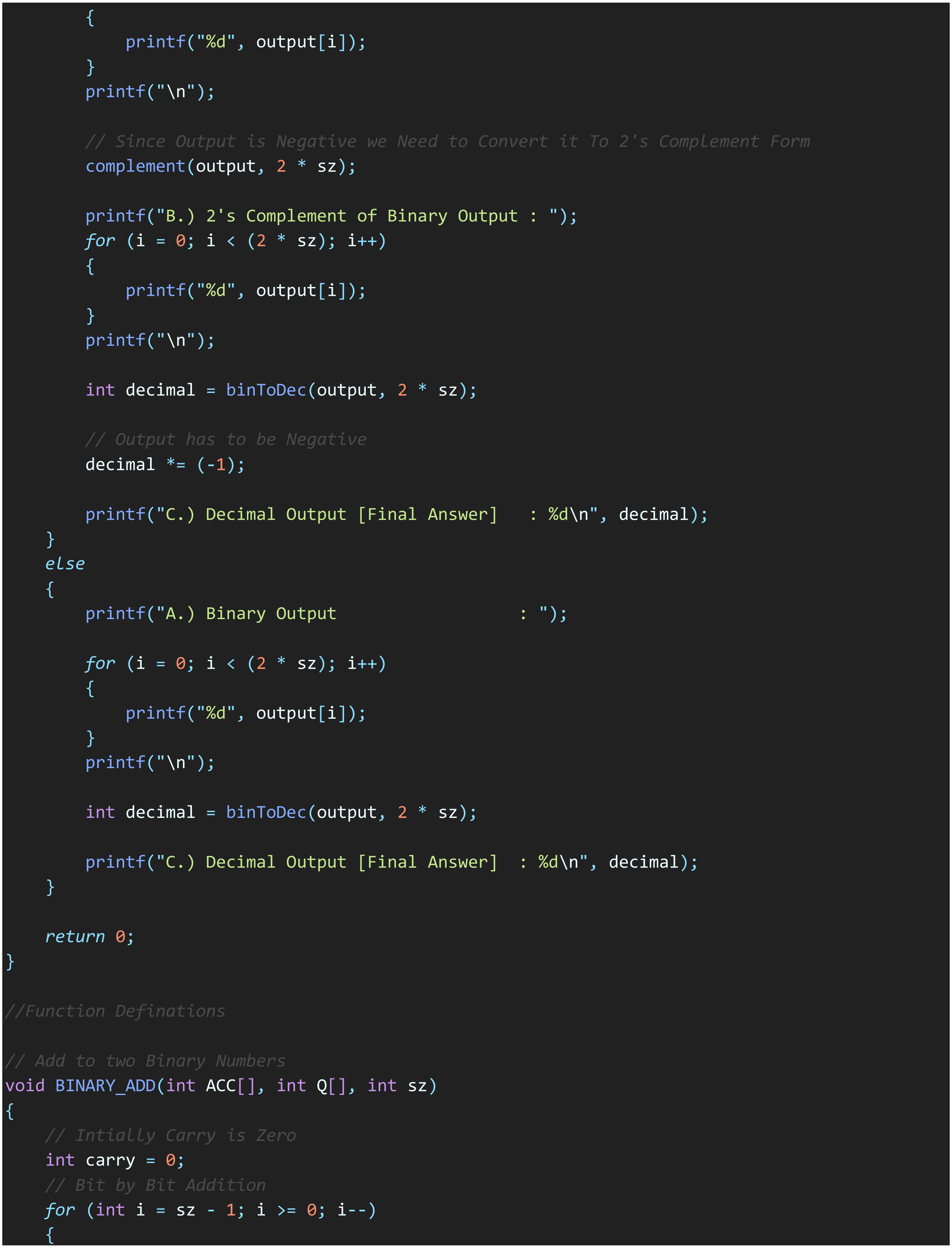
(i) Both positive numbers
(ii) Positive multiplier and Negative multiplicand
(iii) Negative multiplier and Positive multiplicand
(iv) Both numbers negative

Code:

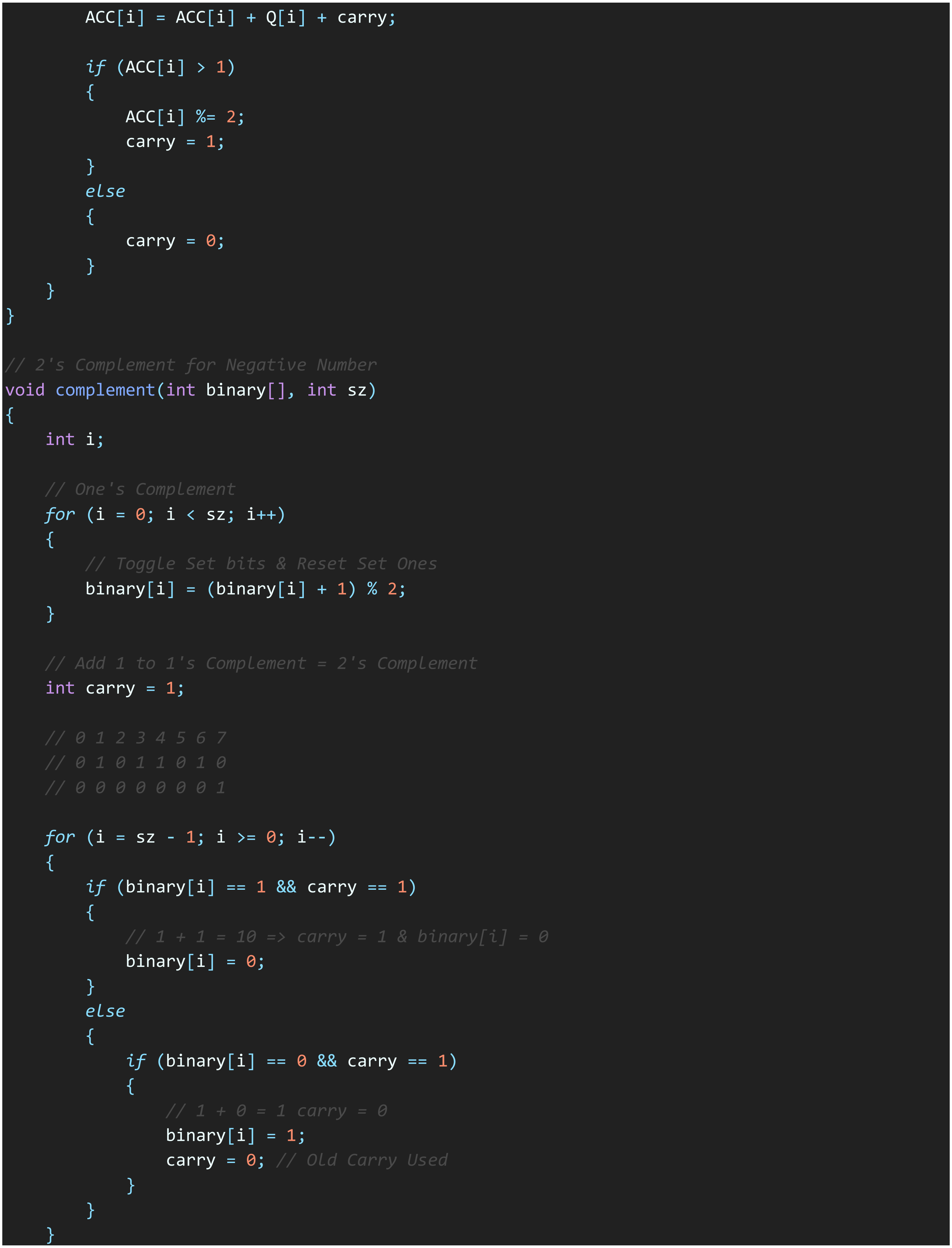
```c
#include <stdio.h>

//Function Declarations

// Booths Algorithm for Multiplication
void MyAlgo(int M[], int Q[], int ACC[], int cnt, int minusM[], int sz);

// Add to two Binary Numbers
void BINARY_ADD(int ACC[], int Q[], int sz);

// 2's Complement for Negative Number
void complement(int binary[], int sz);

// Arithmetic Right Shift
void ARShift(int ACC[], int Q[], int *Qn, int sz);

// Decimal to Binary conversion
void decToBin(int n, int binary[], int sz);

// Binary to decimal conversion
int binToDec(int bin[], int sz);

// Modular Approach Of Programming Followed
int main()
{
    int sz = 8;

    // Input Two Numbers
    int multiplicand;
    printf("\nEnter Multiplicand[127 to -127] : ");
    scanf("%d", &multiplicand);
```



```c
    int multiplier;
    printf("Enter Multiplier  [127 to -127] : ");
    scanf("%d", &multiplier);

    // Array to Store Binary of Multiplier & Multiplicant
    int multiplierBin[8] = {0};
    int multiplicandBin[8] = {0};

    // Accumulator
    int ACC[8] = {0};

    // counter
    int COUNTER = sz;

    // 2's Complement of Multiplicand
    int minusM[sz];

    // Convert Decimal to Binary
    decToBin(multiplier, multiplierBin, sz);
    decToBin(multiplicand, multiplicandBin, sz);

    int i;

    // Initialise -M Array
    for (i = 0; i < sz; i++)
    {
        minusM[i] = multiplicandBin[i];
    }
    complement(minusM, sz);

    //Booths Algorithm
    MyAlgo(multiplicandBin, multiplierBin, ACC, COUNTER, minusM, sz);

    // Final Answer
    int output[2 * sz];

    // output -> Combine ACC and multiplierBin
    for (i = 0; i < sz; i++)
    {
        output[i] = ACC[i];
        output[i + sz] = multiplierBin[i];
    }

    // Case {+ve x -ve} or {-ve x +ve}
    if ((multiplier < 0 && multiplicand > 0) || (multiplier > 0 && multiplicand < 0))
    {
        // Answer is Going to be Negative

        printf("A.) Binary Output                   : ");
        for (i = 0; i < (2 * sz); i++)
```

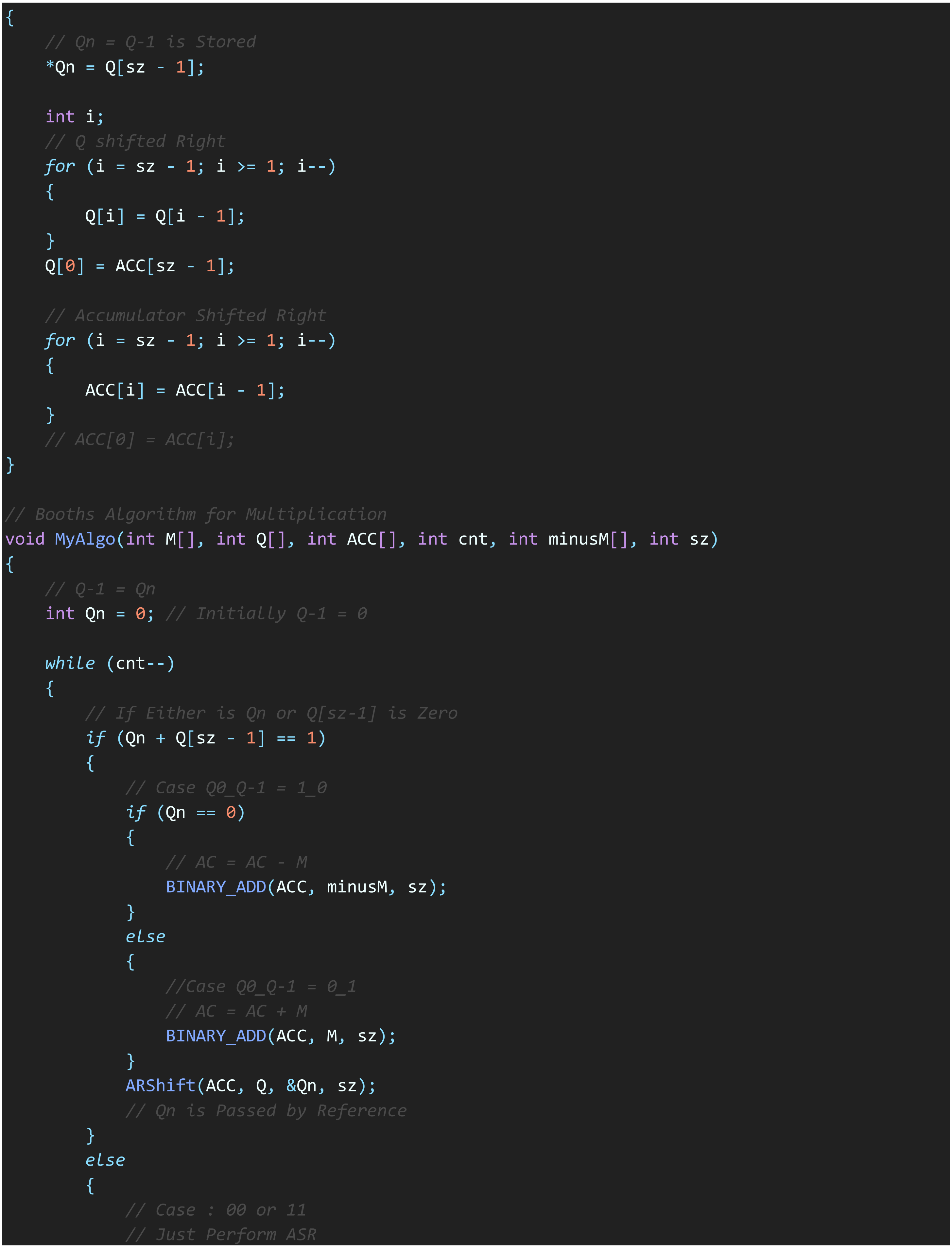


```
        {
            printf("%d", output[i]);
        }
        printf("\n");

        // Since Output is Negative we Need to Convert it To 2's Complement Form
        complement(output, 2 * sz);

        printf("B.) 2's Complement of Binary Output : ");
        for (i = 0; i < (2 * sz); i++)
        {
            printf("%d", output[i]);
        }
        printf("\n");

        int decimal = binToDec(output, 2 * sz);

        // Output has to be Negative
        decimal *= (-1);

        printf("C.) Decimal Output [Final Answer]   : %d\n", decimal);
    }
    else
    {
        printf("A.) Binary Output                  : ");

        for (i = 0; i < (2 * sz); i++)
        {
            printf("%d", output[i]);
        }
        printf("\n");

        int decimal = binToDec(output, 2 * sz);

        printf("C.) Decimal Output [Final Answer]  : %d\n", decimal);
    }

    return 0;
}

//Function Definations

// Add to two Binary Numbers
void BINARY_ADD(int ACC[], int Q[], int sz)
{
    // Intially Carry is Zero
    int carry = 0;
    // Bit by Bit Addition
    for (int i = sz - 1; i >= 0; i--)
    {
```

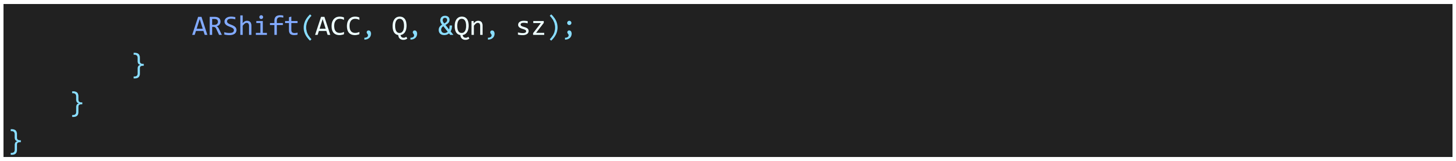


```c
        ACC[i] = ACC[i] + Q[i] + carry;

        if (ACC[i] > 1)
        {
            ACC[i] %= 2;
            carry = 1;
        }
        else
        {
            carry = 0;
        }
    }
}

// 2's Complement for Negative Number
void complement(int binary[], int sz)
{
    int i;

    // One's Complement
    for (i = 0; i < sz; i++)
    {
        // Toggle Set bits & Reset Set Ones
        binary[i] = (binary[i] + 1) % 2;
    }

    // Add 1 to 1's Complement = 2's Complement
    int carry = 1;

    // 0 1 2 3 4 5 6 7
    // 0 1 0 1 1 0 1 0
    // 0 0 0 0 0 0 0 1

    for (i = sz - 1; i >= 0; i--)
    {
        if (binary[i] == 1 && carry == 1)
        {
            // 1 + 1 = 10 => carry = 1 & binary[i] = 0
            binary[i] = 0;
        }
        else
        {
            if (binary[i] == 0 && carry == 1)
            {
                // 1 + 0 = 1 carry = 0
                binary[i] = 1;
                carry = 0; // Old Carry Used
            }
        }
    }
```

```
}

// Decimal to binary conversion
void decToBin(int n, int bin[], int sz)
{
    int idx = sz - 1;
    if (n < 0)
    {
        // Make n Positive
        n *= (-1);

        while (n > 0)
        {
            bin[idx] = n % 2;
            n = n / 2;
            idx--;
        }

        // 2's Complement of "n"
        complement(bin, sz);
    }
    else
    {
        while (n > 0)
        {
            bin[idx] = n % 2;
            n = n / 2;
            idx--;
        }
    }
}

// Binary to decimal conversion
int binToDec(int bin[], int sz)
{
    int decimal = 0;

    // Initializing base value to 1, i.e 2^0
    int base = 1;

    for (int i = sz - 1; i >= 0; i--)
    {
        decimal += bin[i] * base;
        base *= 2;
    }
    return decimal;
}

// Arithmetic Right Shift
void ARShift(int ACC[], int Q[], int *Qn, int sz)
```



```
{
    // Qn = Q-1 is Stored
    *Qn = Q[sz - 1];

    int i;
    // Q shifted Right
    for (i = sz - 1; i >= 1; i--)
    {
        Q[i] = Q[i - 1];
    }
    Q[0] = ACC[sz - 1];

    // Accumulator Shifted Right
    for (i = sz - 1; i >= 1; i--)
    {
        ACC[i] = ACC[i - 1];
    }
    // ACC[0] = ACC[i];
}

// Booths Algorithm for Multiplication
void MyAlgo(int M[], int Q[], int ACC[], int cnt, int minusM[], int sz)
{
    // Q-1 = Qn
    int Qn = 0; // Initially Q-1 = 0

    while (cnt--)
    {
        // If Either is Qn or Q[sz-1] is Zero
        if (Qn + Q[sz - 1] == 1)
        {
            // Case Q0_Q-1 = 1_0
            if (Qn == 0)
            {
                // AC = AC - M
                BINARY_ADD(ACC, minusM, sz);
            }
            else
            {
                //Case Q0_Q-1 = 0_1
                // AC = AC + M
                BINARY_ADD(ACC, M, sz);
            }
            ARShift(ACC, Q, &Qn, sz);
            // Qn is Passed by Reference
        }
        else
        {
            // Case : 00 or 11
            // Just Perform ASR
```



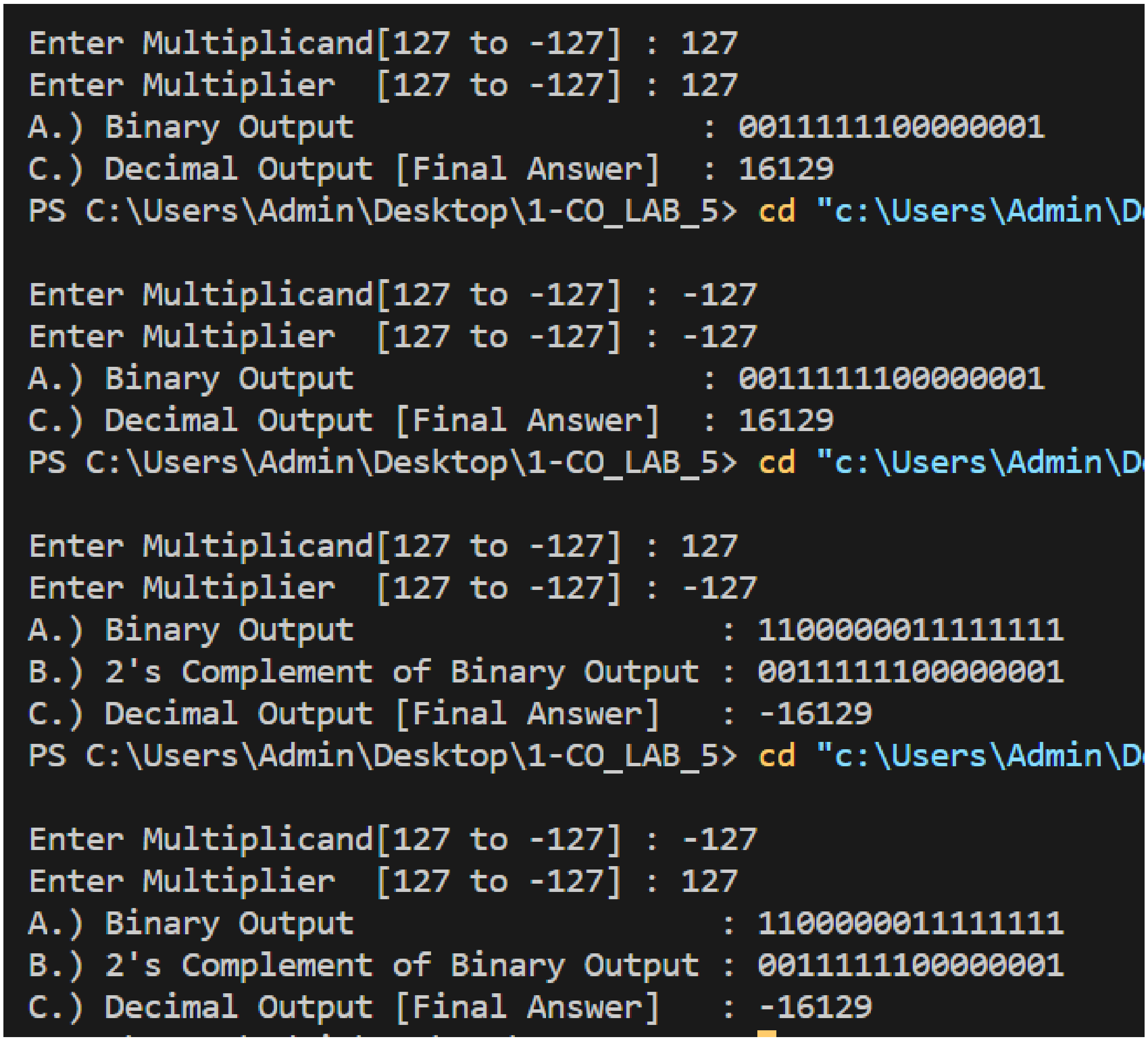
```
            ARShift(ACC, Q, &Qn, sz);
        }
    }
}
```

Test Cases:

All the Four Cases are covered in this Two Test Set.

1.) Test Set 1

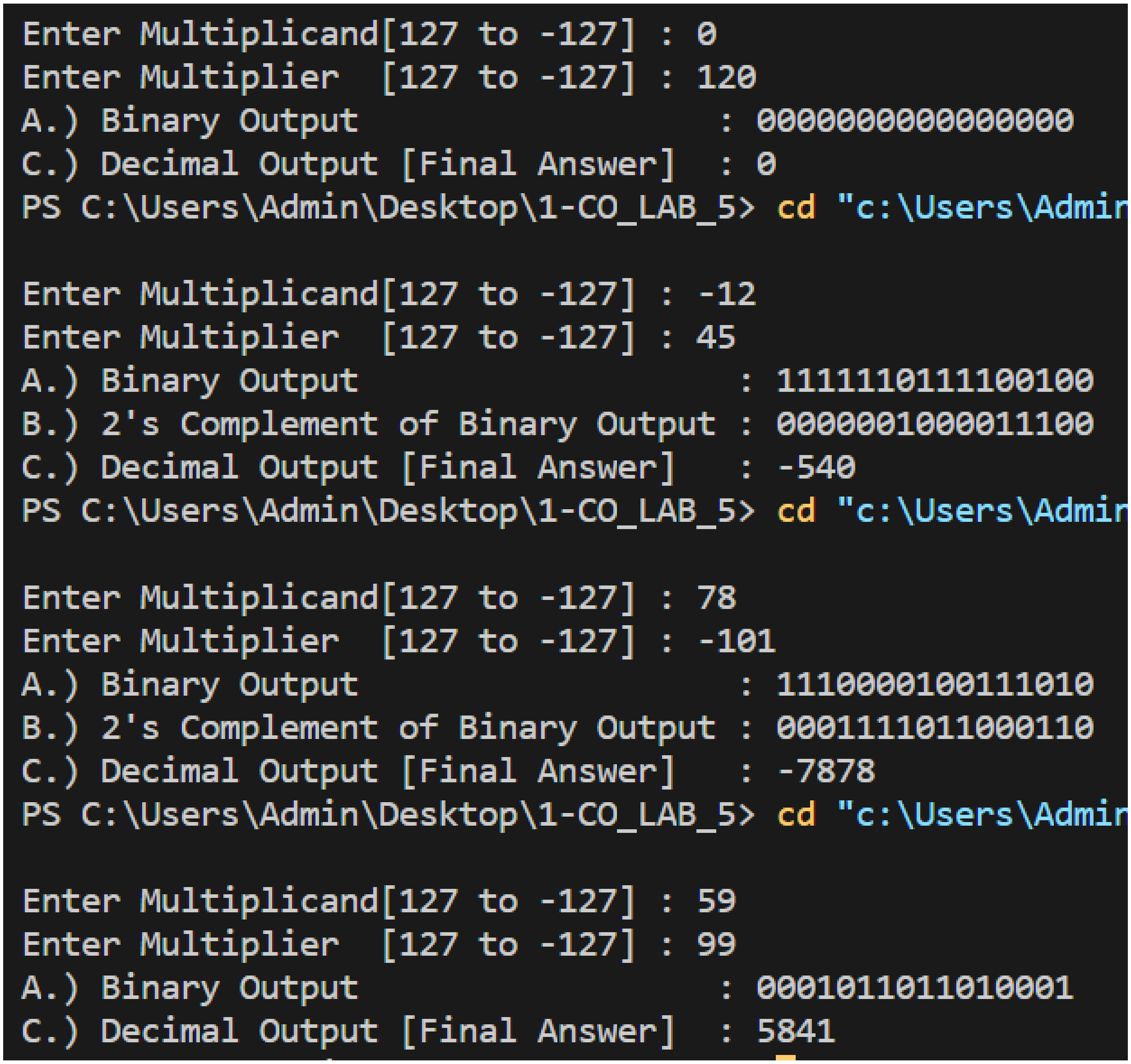
[{127,127}, {-127,-127}, {127,-127}, {-127,127}]



```
Enter Multiplicand[127 to -127] : 127
Enter Multiplier  [127 to -127] : 127
A.) Binary Output                 : 0011111100000001
C.) Decimal Output [Final Answer]  : 16129
PS C:\Users\Admin\Desktop\1-CO_LAB_5> cd "c:\Users\Admin\D

Enter Multiplicand[127 to -127] : -127
Enter Multiplier  [127 to -127] : -127
A.) Binary Output                 : 0011111100000001
C.) Decimal Output [Final Answer]  : 16129
PS C:\Users\Admin\Desktop\1-CO_LAB_5> cd "c:\Users\Admin\D

Enter Multiplicand[127 to -127] : 127
Enter Multiplier  [127 to -127] : -127
A.) Binary Output                  : 1100000011111111
B.) 2's Complement of Binary Output : 0011111100000001
C.) Decimal Output [Final Answer]   : -16129
PS C:\Users\Admin\Desktop\1-CO_LAB_5> cd "c:\Users\Admin\D

Enter Multiplicand[127 to -127] : -127
Enter Multiplier  [127 to -127] : 127
A.) Binary Output                  : 1100000011111111
B.) 2's Complement of Binary Output : 0011111100000001
C.) Decimal Output [Final Answer]   : -16129
```

2.) Test Set 2

[{0,120}, {-12,45}, {78,-101}, {59,99}]



```
Enter Multiplicand[127 to -127] : 0
Enter Multiplier  [127 to -127] : 120
A.) Binary Output                  : 0000000000000000
C.) Decimal Output [Final Answer]  : 0
PS C:\Users\Admin\Desktop\1-CO_LAB_5> cd "c:\Users\Admin

Enter Multiplicand[127 to -127] : -12
Enter Multiplier  [127 to -127] : 45
A.) Binary Output                    : 1111110111100100
B.) 2's Complement of Binary Output : 0000001000011100
C.) Decimal Output [Final Answer]    : -540
PS C:\Users\Admin\Desktop\1-CO_LAB_5> cd "c:\Users\Admin

Enter Multiplicand[127 to -127] : 78
Enter Multiplier  [127 to -127] : -101
A.) Binary Output                    : 1110000100111010
B.) 2's Complement of Binary Output : 0001111011000110
C.) Decimal Output [Final Answer]    : -7878
PS C:\Users\Admin\Desktop\1-CO_LAB_5> cd "c:\Users\Admin

Enter Multiplicand[127 to -127] : 59
Enter Multiplier  [127 to -127] : 99
A.) Binary Output                  : 0001011011010001
C.) Decimal Output [Final Answer]  : 5841
```

Submitted By:

Roll Number: *U19CS012 (D-12)*

Name: **Bhagya Rana**