# Assignment 6

Admn no. U20cs110
Name: Krishna Pandey

**Ques:** Write a C code to perform multiplication of two signed binary numbers using Booth's
multiplication algorithm.
Input: Two binary numbers
Output: Final Product

- In binary
- In equivalent decimal

**Ans:**

```c
// Program to multiply two signed binary numbers Boothe's Multiplication
Algorithm is used

#include <stdio.h>
#define N 5

// Prototype functions
void printArray(int arr[], int SIZE);
void binaryAddition(int arr1[], int arr2[], int SIZE);
int twoCompliment(int arr[], int SIZE);
void binaryToArray(int arr[], long int binary);
void ashr(int arrA[], int arrQ[], int *Qn);
int arrayToBinary(int arr[], int SIZE);
int binaryToDecimal(long long int binary);
void arrCombine(int arr1[], int arr2[], int Comb[]);

// MAIN function
int main()
{
    // take input for M & Q
    long int M, Q, Mcomp;
    printf("Enter in binary format\n");
    printf("Multiplicand : ");
    scanf("%ld", &M);
    printf("Multiplier : ");
    scanf("%ld", &Q);

    // if M & Q are 5 or more digits
    if (M > 1111 || Q > 1111)
```

```
    {
        printf("Multiplicand of Multiplier cannot be greater than 1111.\n\n");

        return 0;
    }

    // if M, Q are negative
    int Ms = 0; // sign bit
    int Qs = 0; // sign bit
    if (M < 0)
    {
        M *= -1;
        Ms = 1;
    }
    if (Q < 0)
    {
        Q *= -1;
        Qs = 1;
    }

    // define variables and arrays
    int arrM[N] = {0}; // multiplicand
    int arrQ[N] = {0}; // multiplier
    int arrA[N] = {0}; // accumulator
    int arrMcomp[N] = {0}; // 2's compliment of multiplicand
    int SC = N;
    int Qn = 0;

    // store M & Q in arrays
    binaryToArray(arrM, M);
    binaryToArray(arrQ, Q);

    // taking 2's compliment if M, Q are negative
    if (Ms == 1)
    {
        M = twoCompliment(arrM, N);
        binaryToArray(arrM, M);
    }
    if (Qs == 1)
    {
        Q = twoCompliment(arrQ, N);
        binaryToArray(arrQ, Q);
    }

    // 2's compliment of M
    Mcomp = twoCompliment(arrM, N);
    // store Mcomp in array
    binaryToArray(arrMcomp, Mcomp);
```

```
// Multiplication
while (SC > 0)
{
    if (arrQ[N-1] == 0)
    {
        if (Qn == 0)
        {
            ashr(arrA, arrQ, &Qn);
            SC--;
        }
        else // Qn == 1
        {
            binaryAddition(arrA, arrM, N);
            ashr(arrA, arrQ, &Qn);
            SC--;
        }
    }

    else // arrQ[N-1] == 1
    {
        if (Qn == 0)
        {
            binaryAddition(arrA, arrMcomp, N);
            ashr(arrA, arrQ, &Qn);
            SC--;
        }
        else // Qn == 1
        {
            ashr(arrA, arrQ, &Qn);
            SC--;
        }
    }
}

int arrP[2*N] = {0}; // Product array

// combine arrA[] and arrQ[] into arrP
arrCombine(arrA, arrQ, arrP);

// convert product into binary from array
long long int product = 0;
int decProduct = 0;

int Ps = arrP[0]; // sign bit

if (Ps == 1) // product is negative
    product = twoCompliment(arrP, 2*N);
```

```
    else // product is positive
        product = arrayToBinary(arrP, 2*N);

    decProduct = binaryToDecimal(product);

    if (Ps == 1) // product has to be negative
    {
        product *= -1;
        decProduct *= -1;
    }

    // Printing the product
    printf("\nProduct in binary : %lld\n", product);
    printf("Product in decimal : %d\n\n", decProduct);

    return 0;
}

// print an array
void printArray(int arr[], int SIZE)
{
    for (int i = 0; i < SIZE; i++)
    {
        printf("%3d", arr[i]);
    }
}

// add two binary numbers stored in an array
void binaryAddition(int arr1[], int arr2[], int SIZE)
{
    int c = 0; // carry
    for (int i = SIZE-1; i >= 0; i--)
    {
        int a = arr1[i];
        int b = arr2[i];

        switch(a)
        {
            case 1 :
                switch(b)
                {
                    case 1 :
                        switch(c)
                        {
                            case 1:
                                arr1[i] = 1;
                                c = 1;
```

```
                        break;

                    case 0:
                        arr1[i] = 0;
                        c = 1;
                        break;
                }
                break;

            case 0 :
                switch(c)
                {
                    case 1:
                        arr1[i] = 0;
                        c = 1;
                        break;

                    case 0:
                        arr1[i] = 1;
                        c = 0;
                        break;
                }
                break;
        }
        break;

    case 0 :
        switch(b)
        {
            case 1 :
                switch(c)
                {
                    case 1:
                        arr1[i] = 0;
                        c = 1;
                        break;

                    case 0:
                        arr1[i] = 1;
                        c = 0;
                        break;
                }
                break;

            case 0 :
                switch(c)
                {
                    case 1:
```

```
                                arr1[i] = 1;
                                c = 0;
                                break;

                            case 0:
                                arr1[i] = 0;
                                c = 0;
                                break;
                        }
                        break;
                }
                break;
        }
    }
}

// convert a binary number into its two's compliment
int twoCompliment(int arr[], int SIZE)
{
    int arrComp[SIZE];
    int arrTemp[SIZE];
    for (int i = 0; i < SIZE; i++)
    {
        arrComp[i] = 0;
        arrTemp[i] = 0;
    }

    for (int i = 0; i < SIZE; i++)
    {
        if (arr[i] == 0)
            arrComp[i] = 1;

        else
            arrComp[i] = 0;
    }

    arrTemp[SIZE-1] = 1;

    binaryAddition(arrComp, arrTemp, SIZE);

    long long int result = arrayToBinary(arrComp, SIZE);

    return result;
}

// store a binary number into an array
void binaryToArray(int arr[], long int binary)
{
```

```
    for (int i = N-1; i >= 0; i--)
    {
        arr[i] = binary % 10;
        binary /= 10;
    }
}

// shift right a binary number
void ashr(int arrA[], int arrQ[], int *Qn)
{
    *Qn = arrQ[N-1];
    int t = arrA[N-1];

    for (int i = N-1; i > 0; i--)
    {
        arrQ[i] = arrQ[i-1];
        arrA[i] = arrA[i-1];
    }

    arrQ[0] = t;
}

// convert array of number into a number
int arrayToBinary(int arr[], int SIZE)
{
    long long int num = 0;
    int mul = 1;
    for (int i = SIZE-1; i >= 0; i--)
    {
        num += (arr[i] * mul);
        mul *= 10;
    }

    return num;
}

// convert binary to decimal
int binaryToDecimal(long long int binary)
{
    int dec = 0;
    int mul = 1;

    while (binary > 0)
    {
        dec += mul * (binary % 10);
        mul *= 2;
        binary /= 10;
    }
```

```
    return dec;
}

// combine two arrays
void arrCombine(int arr1[], int arr2[], int Comb[])
{
    for (int i = 0; i < N; i++)
    {
        Comb[i] = arr1[i];
    }

    for (int i = N; i < 2*N; i++)
    {
        Comb[i] = arr2[i-N];
    }
}
```

## Output Screenshot