Name: Meith Navlakha
SAP: 60004190068
POA Practical

EXPERIMENT 1

AIM: Write a program to implement Booths Algorithm

THEORY:

Booth's multiplication algorithm is a multiplication algorithm that multiplies two signed binary numbers in two's complement notation. It generates 2n bit product and treats both positive and negative unbiasedly.

Booth used desk calculators that were faster at shifting than adding and created the algorithm to increase their speed.

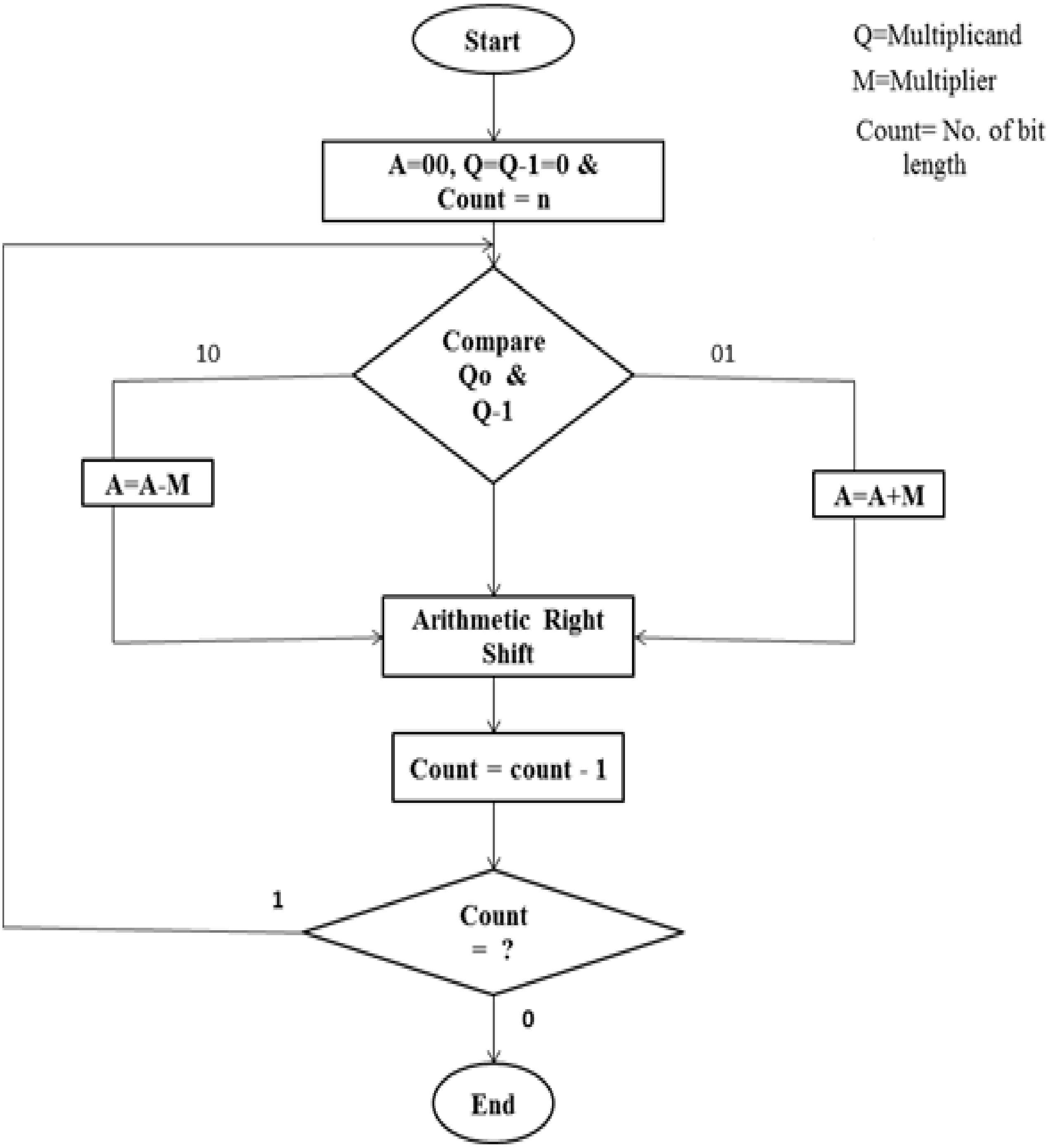
Basic flow of the Booths Algorithm:

1. Multiplier and multiplicand are placed in the Q and M register respectively.
2. Result for this will be stored in the A and Q registers.
3. Initially, A and Q_{-1} register will be 0.
4. Multiplication of a number is done in a cycle.
5. A 1-bit register Q_{-1} is placed right of the least significant bit Q_0 of the register Q.
6. In each of the cycle, Q_0 and Q_{-1} bits will be checked.
    i. If Q_0 and Q_{-1} are 11 or 00 then the bits of A, Q and Q_{-1} are shifted to the right by 1 bit.
    ii. If the value is shown 01 then multiplicand is added to AC. After addition, A, Q_0, Q_{-1} register are shifted to the right by 1 bit.
    iii. If the value is shown 10 then multiplicand is subtracted from AC. After subtraction A, Q_0, Q_{-1} register is shifted to the right by 1 bit.

Best Case : When there is a large block of consecutive 1's and 0's in the multipliers, so that there is minimum number of logical operations taking place, as in addition and subtraction.

Worst case : When there are pairs of alternate 0's and 1's, either 01 or 10 in the multipliers, so that maximum number of additions and subtractions are required.

FLOWCHART:


Start
A=00, Q=Q-1=0 &
Count = n
Q=Multiplicand
M=Multiplier
Count= No. of bit length
10
01
Compare Qo & Q-1
A=A-M
A=A+M
Arithmetic Right Shift
Count = count - 1
1
Count = ?
0
End

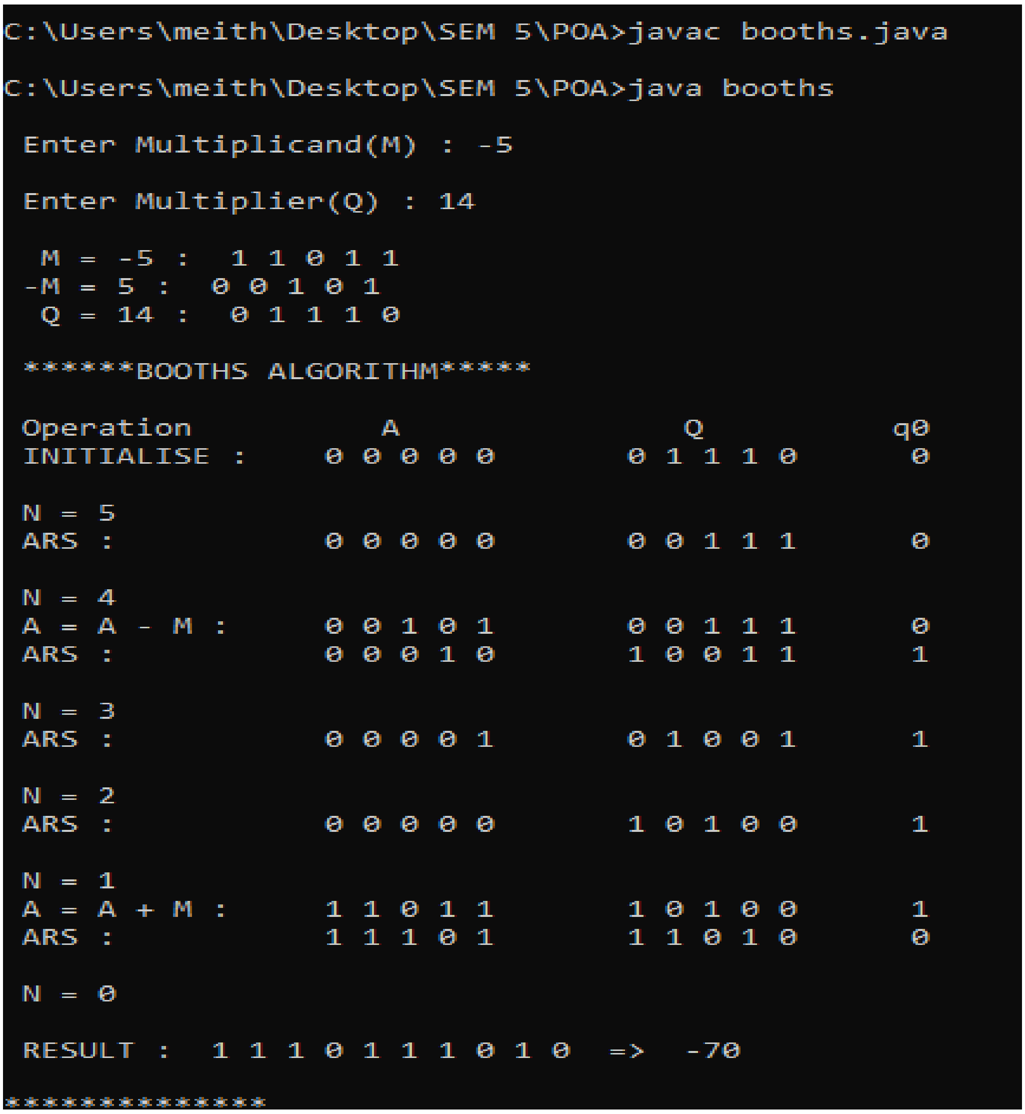
CODE:

```
import java.util.*;
import java.util.Arrays;
import java.lang.Integer;
import java.lang.Math;

public class booths {

  public static String displayArray(int[] arr) {

    String s = "";
    for (int i = 0; i < arr.length; i++) {
      s = s + " " + arr[i];
    }

    return s;
  }

  public static int[] Add(int[] A, int[] M) {

    int c = 0;
    for (int i = A.length - 1; i >= 0; i--) {
      A[i] = A[i] + M[i] + c;

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

    return A;
  }

  public static int[] twoCompliment(int[] arr, int len) {

    //1's Compliment
    for (int i = 0; i < len; i++) {
      arr[i] = (arr[i] + 1) % 2;
    }

    int[] plus1 = new int[len];
    plus1[len - 1] = 1;

    // Add 1
    arr = Add(arr, plus1);

    // System.out.print("\n 2's Compliment : " + displayArray(arr)) ;
    return arr;
  }

  public static int[] tobinary(int num) {

    int m = Math.abs(num);
    // int len =8  ;
    int[] arr = new int[10]; // Extra for sign bit
    int count = 0;

    while (m > 0) {

      arr[count] = m % 2;
      count++;
      m /= 2;
    }

    int[] a = new int[count + 1];

    for (int i = 1; i <= count; i++) {
      a[count + 1 - i] = arr[i - 1];
    }

    // if(num <0){
    //    a = twoCompliment(a , count+1) ;
    // }

    //  System.out.print("\n Binary of "+num+" : "+ displayArray(a)) ;

    return a;
  }

  public static int[][] rightShift(int[] A, int[] Q, int q0, int len) {
    int temp = A[len - 1];
    q0 = Q[len - 1];
    int[][] res = new int[3][];

    for (int k = len - 1; k > 0; k--) {
      A[k] = A[k - 1];
      Q[k] = Q[k - 1];

    }
    Q[0] = temp;

    res[0] = A;
    res[1] = Q;
    res[2] = new int[] {
      q0
    };

    System.out.print("\n ARS : \t\t" + displayArray(A) + "\t" + displayArray(Q) +
"\t" + q0);

    return res;
  }

  public static int toDecimal(int[] arr) {

    int num = 0;
    for (int i = 0; i < arr.length; i++) {
      num = num * 2 + arr[i];
    }

    return num;
  }

  public static void combine(int[] A, int[] Q) {

    if (A[0] == 1) {
      // Negative
      int[] result = new int[2 * A.length - 1];
      System.arraycopy(A, 1, result, 0, A.length - 1);
      System.arraycopy(Q, 0, result, A.length - 1, A.length);

      result = twoCompliment(result, result.length);
      System.out.print("\n\n RESULT : " + displayArray(A) + "" +
displayArray(Q) + "  =>  -" + toDecimal(result));

    } else {
      int[] result = new int[2 * A.length];
      System.arraycopy(A, 0, result, 0, A.length);
      System.arraycopy(Q, 0, result, A.length, A.length);
      System.out.print("\n\n RESULT : " + displayArray(result) + "  =>  " +
toDecimal(result));

    }

    System.out.print("\n\n**************");

  }

  public static int[] padding(int[] arr, int len) {

    int[] pad = new int[len];
    int k = arr.length;
    int i = 0;
    while (i < len && k > 0) {

      pad[len - arr.length + i] = arr[arr.length - k];
      i++;
      k--;
    }
    // System.out.print("\nPADDED :   \t"+ displayArray(pad) );
    return pad;
  }

  public static void boothsAlgo(int[] M, int[] minusM, int[] Q) {

    int[] A = new int[M.length];
    int q0 = 0;
    int N = M.length;

    System.out.print("\n Operation\t    A\t            Q \t       q0");
    System.out.print("\n INITIALISE :  \t" + displayArray(A) + "\t" +
displayArray(Q) + "\t" + q0);
    System.out.print("\n\n N = " + N);
```

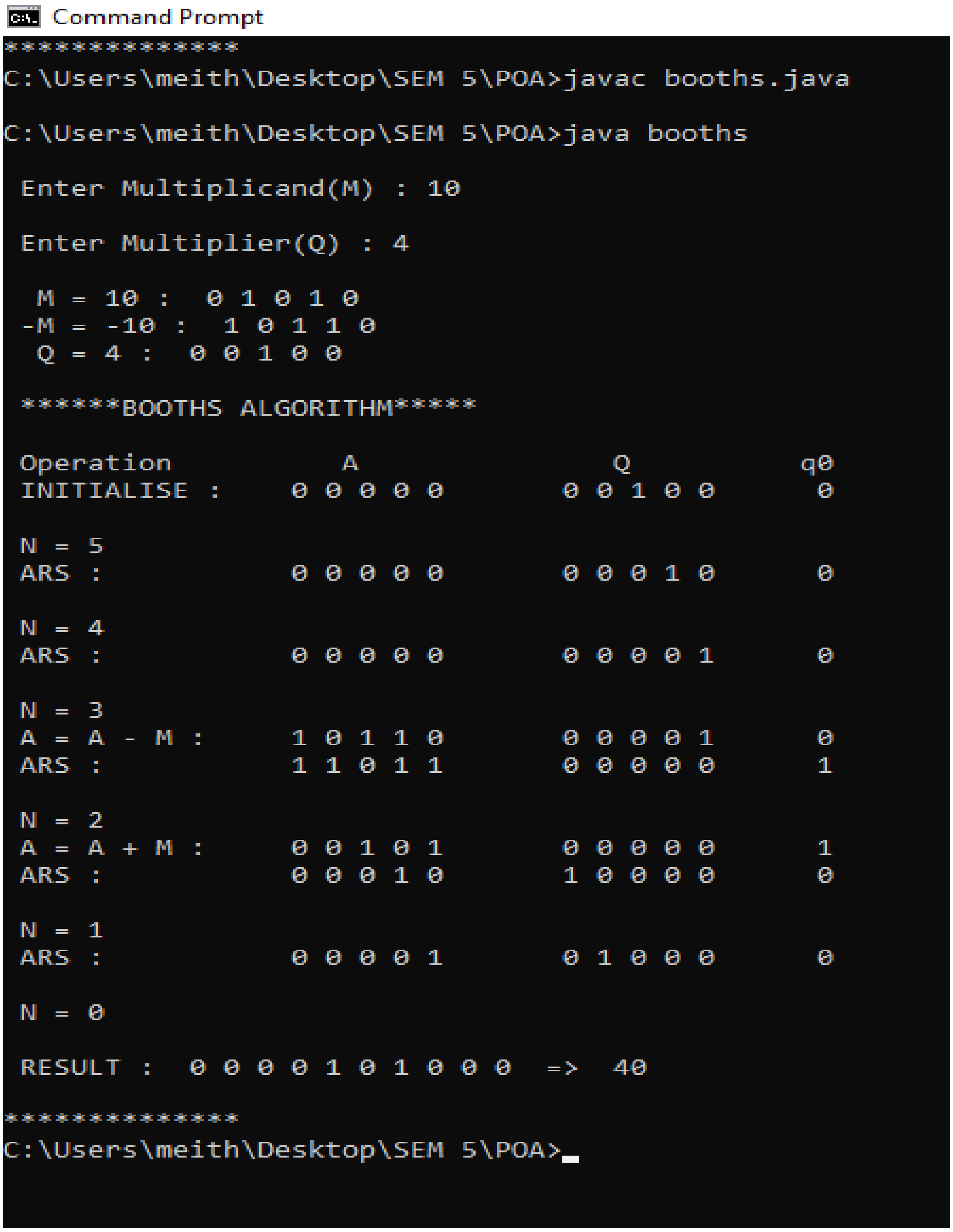
```
while (N > 0) {

      if (q0 == 0 && Q[Q.length - 1] == 1) {
        //10
        A = Add(A, minusM);
        System.out.print("\n A = A - M :  \t" + displayArray(A) + "\t" +
displayArray(Q) + "\t" + q0);

      } else if (q0 == 1 && Q[Q.length - 1] == 0) {
        // 01
        A = Add(A, M);
        System.out.print("\n A = A + M :  \t" + displayArray(A) + "\t" +
displayArray(Q) + "\t" + q0);

      }

      int[][] res = rightShift(A, Q, q0, A.length);
      A = res[0];
      Q = res[1];
      q0 = res[2][0];
      N--;
      System.out.print("\n\n N = " + N);

    }
    // System.out.print("\n\n FINAL : "+ displayArray(A)+""+ displayArray(Q));
    combine(A, Q);

  }

  public static void main(String args[]) {

    Scanner sc = new Scanner(System.in);

    System.out.print("\n Enter Multiplicand(M) : ");
    int m = sc.nextInt();

    System.out.print("\n Enter Multiplier(Q) : ");
    int q = sc.nextInt();

    // int m = -5 ;
    // int q = 14 ;





  int[] bin_Q = tobinary(q);
    int[] bin_M = tobinary(m);
    //int[] minusM = tobinary(-m) ;
    int[] M;
    int[] Q;

    if (Math.abs(q) > Math.abs(m)) {
      M = padding(bin_M, bin_Q.length);
      Q = bin_Q;
    } else {
      Q = padding(bin_Q, bin_M.length);
      M = bin_M;

    }

    int[] minusM = new int[M.length];
    System.arraycopy(M, 0, minusM, 0, minusM.length);

    if (q < 0) {
      Q = twoCompliment(Q, Q.length);
    }

    if (m < 0) {
      M = twoCompliment(M, M.length);

    } else {
      minusM = twoCompliment(minusM, minusM.length);;
    }

    System.out.print("\n  M = " + m + " : " + displayArray(M));
    System.out.print("\n -M = " + (-1 * m) + " : " + displayArray(minusM));
    System.out.print("\n  Q = " + q + " : " + displayArray(Q));

    System.out.print("\n\n ******BOOTHS ALGORITHM*****\n");
    boothsAlgo(M, minusM, Q);
```

```
 }
}
```

OUTPUT:



```
C:\Users\meith\Desktop\SEM 5\POA>javac booths.java

C:\Users\meith\Desktop\SEM 5\POA>java booths

 Enter Multiplicand(M) : -5

 Enter Multiplier(Q) : 14

  M = -5 :  1 1 0 1 1
 -M = 5 :  0 0 1 0 1
  Q = 14 :  0 1 1 1 0

 ******BOOTHS ALGORITHM*****

 Operation             A               Q            q0
 INITIALISE :     0 0 0 0 0       0 1 1 1 0       0

 N = 5
 ARS :            0 0 0 0 0       0 0 1 1 1       0

 N = 4
 A = A - M :      0 0 1 0 1       0 0 1 1 1       0
 ARS :            0 0 0 1 0       1 0 0 1 1       1

 N = 3
 ARS :            0 0 0 0 1       0 1 0 0 1       1

 N = 2
 ARS :            0 0 0 0 0       1 0 1 0 0       1

 N = 1
 A = A + M :      1 1 0 1 1       1 0 1 0 0       1
 ARS :            1 1 1 0 1       1 1 0 1 0       0

 N = 0

 RESULT :  1 1 1 0 1 1 1 0 1 0  =>  -70

***************
```



```
**************
C:\Users\meith\Desktop\SEM 5\POA>javac booths.java

C:\Users\meith\Desktop\SEM 5\POA>java booths

 Enter Multiplicand(M) : 10

 Enter Multiplier(Q) : 4

  M = 10 :   0 1 0 1 0
 -M = -10 :   1 0 1 1 0
  Q = 4 :   0 0 1 0 0

 ******BOOTHS ALGORITHM*****

 Operation            A               Q          q0
 INITIALISE :     0 0 0 0 0       0 0 1 0 0      0

 N = 5
 ARS :            0 0 0 0 0       0 0 0 1 0      0

 N = 4
 ARS :            0 0 0 0 0       0 0 0 0 1      0

 N = 3
 A = A - M :      1 0 1 1 0       0 0 0 0 1      0
 ARS :            1 1 0 1 1       0 0 0 0 0      1

 N = 2
 A = A + M :      0 0 1 0 1       0 0 0 0 0      1
 ARS :            0 0 0 1 0       1 0 0 0 0      0

 N = 1
 ARS :            0 0 0 0 1       0 1 0 0 0      0

 N = 0

 RESULT :  0 0 0 0 1 0 1 0 0 0  =>  40

**************
C:\Users\meith\Desktop\SEM 5\POA>_
```

CONCLUSION:

In this experiment I implemented Booths Algorithm in Java. Booths Algorithm is used for signed multiplication of integers. Here, we use 2’s compliment for performing subtraction as processor executes addition much faster than subtraction also, Arithmetic Right Shift is used after every step. At the end we get 2n bit result where n is the number of bits of the highest value.