BOOTH'S MULTIPLICATION ALGORITHM
EXP NO: 34
AIM: To implement Booth's multiplication algorithm in C for multiplying two numbers using binary arithmetic.

ALGORITHM:
Step 1: Start.
Step 2: Declare and initialize necessary variables and arrays:

- a, b, c for input numbers and carry.
- a1, b1 for absolute values of inputs.
- Arrays com[], anum[], anumcp[], bnum[], acomp[], bcomp[], pro[], and res[] for binary representations and intermediate results.

Step 3: Convert the numbers a and b into binary form and store them in anum[] and bnum[]. Compute the two's complement of b and store it in bcomp[].

Step 4: Perform two's complement conversion for the negative input if necessary and adjust the binary representations accordingly.

Step 5: Apply Booth's multiplication algorithm: a. Initialize quotient (q) and product (pro[]). b. Perform arithmetic right shift on the product and update the quotient and the product based on the current bit of the multiplicand. c. If the multiplicand bit and the previous quotient bit are 10, subtract B (using two's complement). d. If the multiplicand bit and the previous quotient bit are 01, add B. e. Shift the accumulator and quotient as needed.

Step 6: Print the binary equivalents and the product.
Step 7: End.

PROGRAM:

```
#include <stdio.h>

#include <math.h>


int a = 0, b = 0, c = 0, a1 = 0, b1 = 0, com[5] = {1, 0, 0, 0, 0};

int anum[5] = {0}, anumcp[5] = {0}, bnum[5] = {0};

int acomp[5] = {0}, bcomp[5] = {0}, pro[5] = {0}, res[5] = {0};


void binary() {

   a1 = fabs(a);

   b1 = fabs(b);

   int r, r2, i, temp;


   // Convert numbers to binary

   for (i = 0; i < 5; i++) {

      r = a1 % 2;

      a1 = a1 / 2;

      r2 = b1 % 2;
```

```
    b1 = b1 / 2;
    anum[i] = r;
    anumcp[i] = r;
    bnum[i] = r2;
    bcomp[i] = (r2 == 0) ? 1 : 0;
    acomp[i] = (r == 0) ? 1 : 0;
}

// Part for two's complement
c = 0;
for (i = 0; i < 5; i++) {
    res[i] = com[i] + bcomp[i] + c;
    c = (res[i] >= 2) ? 1 : 0;
    res[i] = res[i] % 2;
}
for (i = 4; i >= 0; i--) {
    bcomp[i] = res[i];
}

// In case of negative inputs
if (a < 0) {
    c = 0;
    for (i = 4; i >= 0; i--) {
        res[i] = 0;
    }
    for (i = 0; i < 5; i++) {
        res[i] = com[i] + acomp[i] + c;
        c = (res[i] >= 2) ? 1 : 0;
        res[i] = res[i] % 2;
    }
    for (i = 4; i >= 0; i--) {
        anum[i] = res[i];
        anumcp[i] = res[i];
    }
```

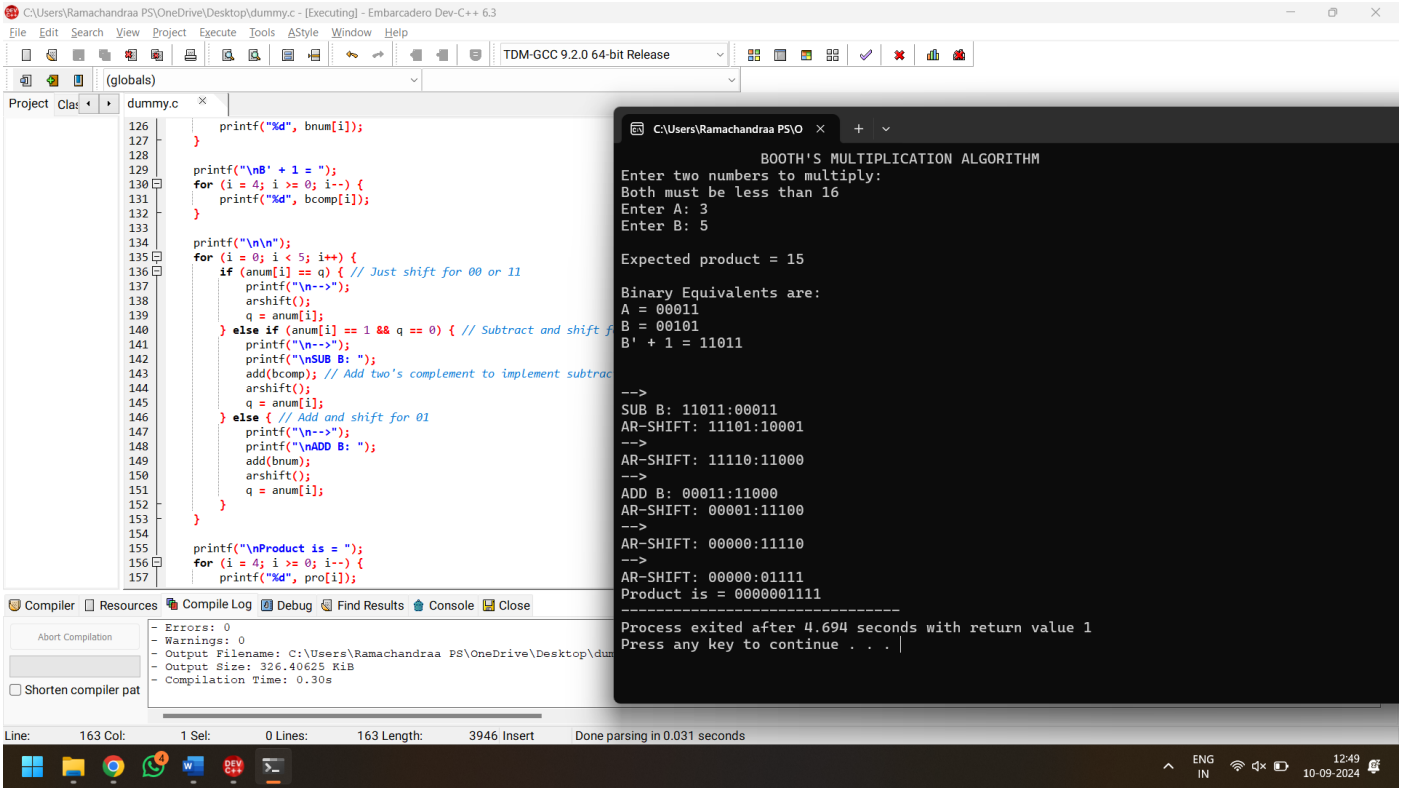
```
    }

    if (b < 0) {
        for (i = 0; i < 5; i++) {
            temp = bnum[i];
            bnum[i] = bcomp[i];
            bcomp[i] = temp;
        }
    }
}

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

void arshift() { // Arithmetic shift right
    int temp = pro[4], temp2 = pro[0], i;
    for (i = 1; i < 5; i++) {
        pro[i - 1] = pro[i];
    }
```

```
    pro[4] = temp;
    for (i = 1; i < 5; i++) {
        anumcp[i - 1] = anumcp[i];
    }
    anumcp[4] = temp2;
    printf("\nAR-SHIFT: ");
    for (i = 4; i >= 0; i--) {
        printf("%d", pro[i]);
    }
    printf(":");
    for (i = 4; i >= 0; i--) {
        printf("%d", anumcp[i]);
    }
}

void main() {
    int i, q = 0;
    printf("\t\tBOOTH'S MULTIPLICATION ALGORITHM");
    printf("\nEnter two numbers to multiply: ");
    printf("\nBoth must be less than 16");

    // Simulating for two numbers each below 16
    do {
        printf("\nEnter A: ");
        scanf("%d", &a);
        printf("Enter B: ");
        scanf("%d", &b);
    } while (a >= 16 || b >= 16);

    printf("\nExpected product = %d", a * b);
    binary();

    printf("\n\nBinary Equivalents are: ");
    printf("\nA = ");
```

```
for (i = 4; i >= 0; i--) {
   printf("%d", anum[i]);
}

printf("\nB = ");
for (i = 4; i >= 0; i--) {
   printf("%d", bnum[i]);
}

printf("\nB' + 1 = ");
for (i = 4; i >= 0; i--) {
   printf("%d", bcomp[i]);
}

printf("\n\n");
for (i = 0; i < 5; i++) {
   if (anum[i] == q) { // Just shift for 00 or 11
      printf("\n-->");
      arshift();
      q = anum[i];
   } else if (anum[i] == 1 && q == 0) { // Subtract and shift for 10
      printf("\n-->");
      printf("\nSUB B: ");
      add(bcomp); // Add two's complement to implement subtraction
      arshift();
      q = anum[i];
   } else { // Add and shift for 01
      printf("\n-->");
      printf("\nADD B: ");
      add(bnum);
      arshift();
      q = anum[i];
   }
}
```

```
    printf("\nProduct is = ");
    for (i = 4; i >= 0; i--) {
        printf("%d", pro[i]);
    }
    for (i = 4; i >= 0; i--) {
        printf("%d", anumcp[i]);
    }
}
```

INPUT & OUTPUT:



RESULT: Thus, the program was executed successfully using DevC++.