

BOOTH'S ALGORITHM

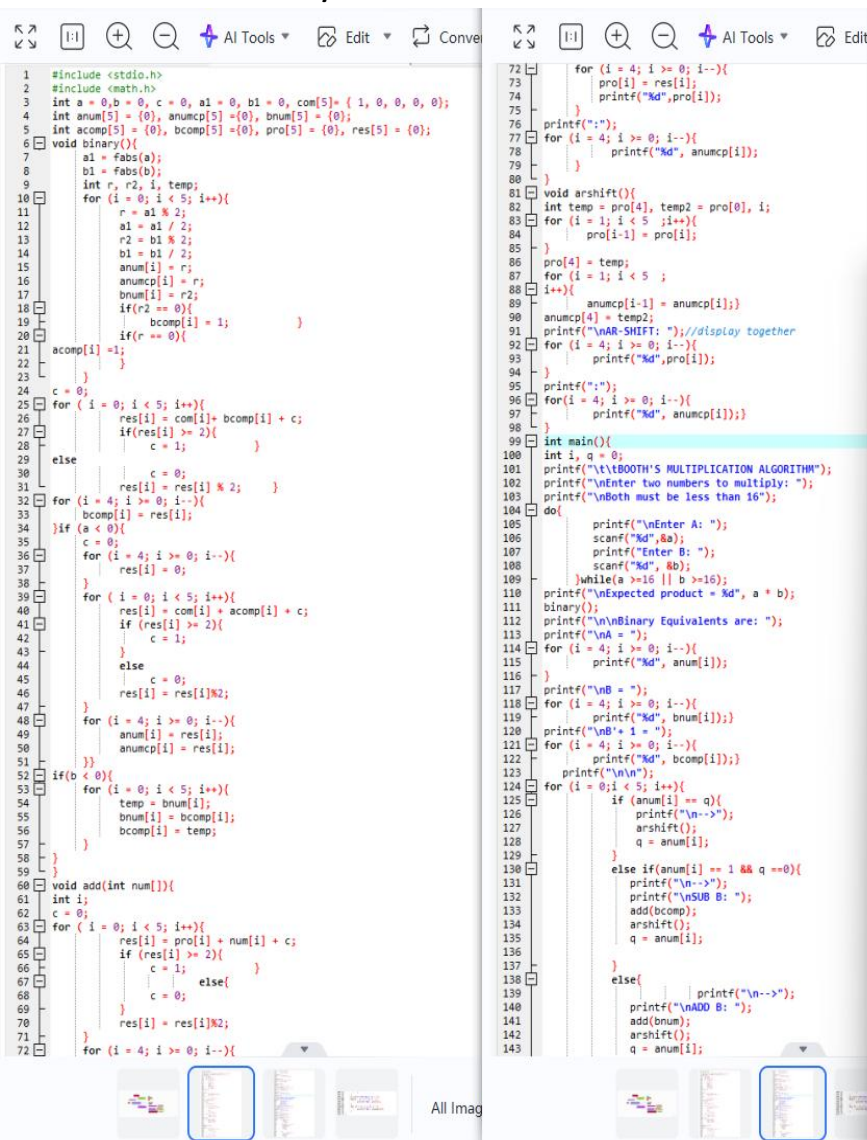
EXP NO: 34

AIM: To write a C program to Booth's Algorithm.

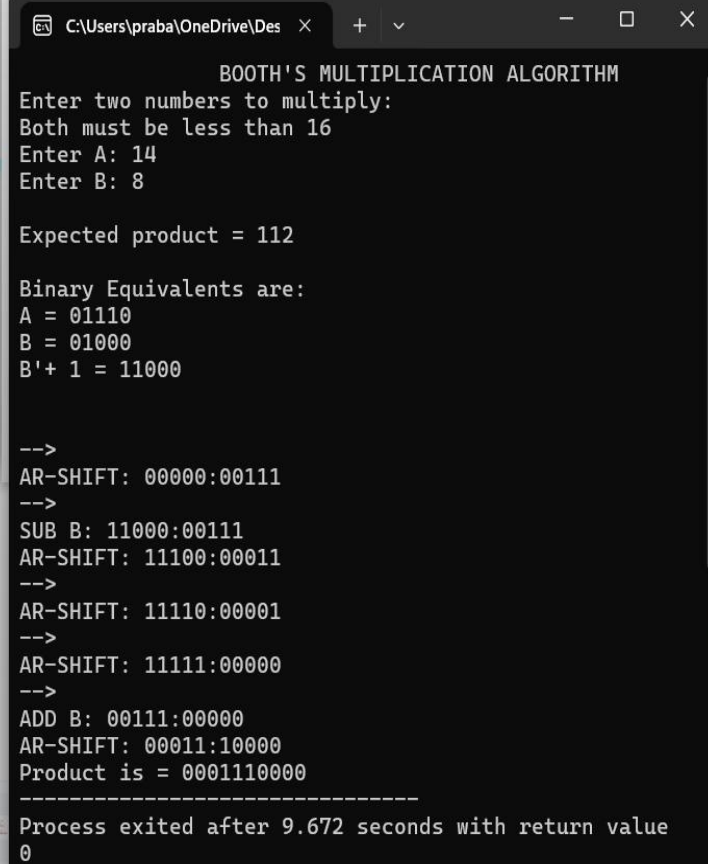
ALGORITHM:

1. Initialize the multiplicand and multiplier, and set up the product register (P), accumulator, and a flag (Q-1).
2. Check the last two bits (Q and Q-1):
 - If Q and Q-1 are 01, add the multiplicand to the product.
 - If Q and Q-1 are 10, subtract the multiplicand from the product.
3. Perform arithmetic right shift on the product and Q, including the Q-1 bit.
4. Repeat steps 2 and 3 for the number of bits in the multiplier.
5. Output the final value in the product register as the multiplication result.

PROGRAM/OUTPUT SS:



```
1 #include <stdio.h>
2 #include <math.h>
3 int a = 0, b = 0, c = 0, a1 = 0, b1 = 0, com[5] = {1, 0, 0, 0, 0};
4 int anum[5] = {0}, anumcp[5] = {0}, bnum[5] = {0};
5 int acomp[5] = {0}, bcomp[5] = {0}, pro[5] = {0}, res[5] = {0};
6 void binary()
7 {
8     a1 = fabs(a);
9     b1 = fabs(b);
10    int r, r2, i, temp;
11    for (i = 0; i < 5; i++)
12    {
13        r = a1 % 2;
14        a1 = a1 / 2;
15        r2 = b1 % 2;
16        b1 = b1 / 2;
17        anum[i] = r;
18        anumcp[i] = r;
19        bnum[i] = r2;
20        bcomp[i] = r2;
21        if (r2 == 0)
22            bcomp[i] = 1;
23    }
24    acomp[i] = 1;
25    c = 0;
26    for (i = 0; i < 5; i++)
27    {
28        res[i] = com[i] + bcomp[i] + c;
29        if (res[i] >= 2)
30        {
31            c = 1;
32            res[i] = res[i] % 2;
33        }
34        else
35            c = 0;
36    }
37    for (i = 4; i >= 0; i--)
38    {
39        bcomp[i] = res[i];
40        if (a < 0)
41        {
42            c = 0;
43            for (i = 4; i >= 0; i--)
44            {
45                res[i] = 0;
46            }
47            for (i = 0; i < 5; i++)
48            {
49                res[i] = com[i] + acomp[i] + c;
50                if (res[i] >= 2)
51                {
52                    c = 1;
53                    res[i] = res[i] % 2;
54                }
55                else
56                    c = 0;
57            }
58            res[i] = res[i] % 2;
59        }
60        else
61            c = 0;
62    }
63    if (b < 0)
64    {
65        for (i = 0; i < 5; i++)
66        {
67            temp = bnum[i];
68            bnum[i] = bcomp[i];
69            bcomp[i] = temp;
70        }
71    }
72    void add(int num[])
73    {
74        int i;
75        c = 0;
76        for (i = 0; i < 5; i++)
77        {
78            res[i] = pro[i] + num[i] + c;
79            if (res[i] >= 2)
80            {
81                c = 1;
82                res[i] = res[i] % 2;
83            }
84            else
85                c = 0;
86        }
87        res[i] = res[i] % 2;
88    }
89    for (i = 4; i >= 0; i--)
90    {
91        for (i = 4; i >= 0; i--)
92        {
93            pro[i] = res[i];
94            printf("%d", pro[i]);
95        }
96        printf("\n");
97        for (i = 4; i >= 0; i--)
98        {
99            printf("%d", anumcp[i]);
100           printf("\n");
101       }
102       void arshift()
103       {
104           int temp = pro[4], temp2 = pro[0], i;
105           for (i = 1; i < 5; i++)
106               pro[i-1] = pro[i];
107           pro[4] = temp;
108           for (i = 1; i < 5; i++)
109               anumcp[i-1] = anumcp[i];
110           anumcp[4] = temp2;
111           printf("\nAR-SHIFT: ");
112           for (i = 4; i >= 0; i--)
113               printf("%d", anumcp[i]);
114           printf("\n");
115           printf("\n");
116           for (i = 4; i >= 0; i--)
117               printf("%d", bnum[i]);
118           printf("\n");
119           for (i = 4; i >= 0; i--)
120               printf("%d", bnum[i] + 1);
121           for (i = 4; i >= 0; i--)
122               printf("%d", bcomp[i]);
123           printf("\n");
124           for (i = 0; i < 5; i++)
125           {
126               if (anum[i] == q)
127               {
128                   printf("\n-->");
129                   arshift();
130                   q = anum[i];
131               }
132               else if (anum[i] == 1 && q == 0)
133               {
134                   printf("\n-->");
135                   printf("\nSUB B: ");
136                   add(bcomp);
137                   arshift();
138                   q = anum[i];
139               }
140               else
141               {
142                   printf("\n-->");
143                   add(bnum);
144                   arshift();
145                   q = anum[i];
146               }
147           }
148       }
149       printf("\nProduct is = ");
150       for (i = 4; i >= 0; i--)
151           printf("%d", anumcp[i]);
152   }
153 }
```



```
C:\Users\praba\OneDrive\Des x + -
BOOTH'S MULTIPLICATION ALGORITHM
Enter two numbers to multiply:
Both must be less than 16
Enter A: 14
Enter B: 8

Expected product = 112

Binary Equivalents are:
A = 01110
B = 01000
B'+ 1 = 11000

-->
AR-SHIFT: 00000:00111
-->
SUB B: 11000:00111
AR-SHIFT: 11100:00011
-->
AR-SHIFT: 11110:00001
-->
AR-SHIFT: 11111:00000
-->
ADD B: 00111:00000
AR-SHIFT: 00011:10000
Product is = 0001110000

-----
Process exited after 9.672 seconds with return value
0
```

RESULT: Thus the code has been executed successfully using DevC++.