

BOOTH'S ALGORITHM:

```
#include <iostream>
```

```
#include<vector>
```

```
#include<algorithm>
```

```
using namespace std;
```

```
vector<int> binary(int x){
```

```
    vector<int> ans;
```

```
    while(x){
```

```
        ans.push_back(x%2);
```

```
        x/=2;
```

```
    }
```

```
    reverse(ans.begin(), ans.end());
```

```
    return ans;
```

```
}
```

```
int to_decimal(vector<int> x){
```

```
    int result = 0;
```

```
    for (auto d : x){
```

```
        result = result * 2 + d;
```

```
    }
```

```
    return result;
```

```
}
```

```
void balance( vector <int> &a, vector <int> &b){
```

```
    int max = a.size() > b.size() ? a.size() : b.size();
```

```
    while(a.size()<max){
```

```
        a.insert(a.begin(), 0);
```

```
    }
```

```

while(b.size()<max){
    b.insert(b.begin(), 0);
}
}

```

```

void display(vector <int> x){
    for(auto i:x){
        cout<<i;
    }

}

```

```

void right_shift(vector <int> & Q, vector <int> &A, int &Qq){
    int n = Q.size();
    Qq = Q[n-1];
    for(int i=n-1; i>0; i--){
        Q[i] = Q[i-1];
    }
    Q[0] = A[n-1];
    for(int i=n-1; i>0; i--){
        A[i] = A[i-1];
    }
}

```

```

vector<int> bool_add(vector <int> A, vector <int> M){
    int n = M.size(); vector <int> res;
    for(int i=0; i<n; i++){
        res.push_back(A[i]+M[i]);
    }
    for(int i=n-1; i>0; i--){
        if(res[i]==2){

```

```

        res[i] =0; res[i-1]++;
    }
    if(res[i]==3){
        res[i] =1; res[i-1]++;
    }
}
if(res[0]==2) res[0] = 0;
if(res[0]==3) res[0] = 1;
return res;
}

```

```

vector<int> init_acc(int n){
    vector <int> ans;
    ans.push_back(0);
    while(n){
        ans.push_back(0);
        n--;
    }
    return ans;
}

```

```

vector <int> twos_complement(vector<int> q){
    vector <int> ans; int n= q.size();
    for(int i=0; i<n; i++){
        if(q[i]==0) ans.push_back(1);
        else ans.push_back(0);
    }
    int i= n-1;
    while(ans[i]==1){
        ans[i]=0; i--;
    }
}

```

```
    ans[i]=1;
    return ans;
}
```

```
vector<int> Booths_algo(vector<int> add_M, vector<int> sub_M,vector<int> Q,vector<int> A )
```

```
{
    int Q1=0;
    int count = add_M.size();

    while(count!=0)
    {
        if(Q.back()>Q1)
        {

            A = bool_add(A,sub_M);

        }
        else if(Q.back()<Q1)
        {
            A= bool_add(A,add_M);

        }
        right_shift(Q,A,Q1);
        count--;
    }
}
```

```

    cout<<endl<<endl<<"The result is: ";
    display(A);display(Q);
    A.insert(A.end(),Q.begin(),Q.end());
    return A;
}

```

```

int main()
{
    int multiplicand;
    int multiplier;
    int a = 0;
    cout<<"Enter the Multiplicand: ";
    cin>>multiplicand;
    cout<<endl;
    cout<<"Enter the Multiplier: ";
    cin>>multiplier;

    vector<int> add_M = binary(multiplicand);

    vector<int> Q = binary(multiplier);

    int j = 0;
    vector <int> A = init_acc(add_M.size());
    balance(A,Q);
    balance(add_M,Q);
    vector<int> sub_M = twos_complement(add_M);

```

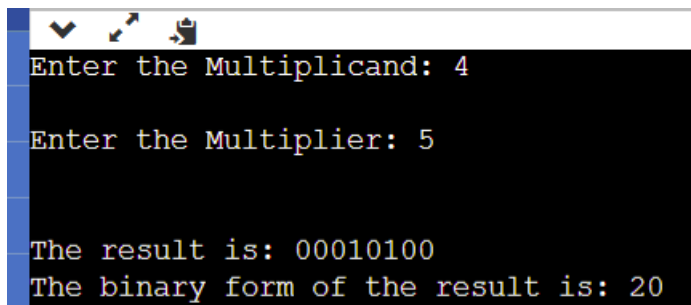
```

vector<int> result = Booths_algo(add_M,sub_M,Q,A);

if(multiplicand<0 || multiplier<0)
{
    result = twos_complement(result);
    int r = to_decimal(result);
    cout<<"The binary form of the result is: -"<<endl<<r;
    cout<<endl;
}
else
{
    int r= to_decimal(result);
    cout<<endl<<"The binary form of the result is: "<<r;
}

return 0;
}

```



```

Enter the Multiplicand: 4
Enter the Multiplier: 5

The result is: 00010100
The binary form of the result is: 20

```