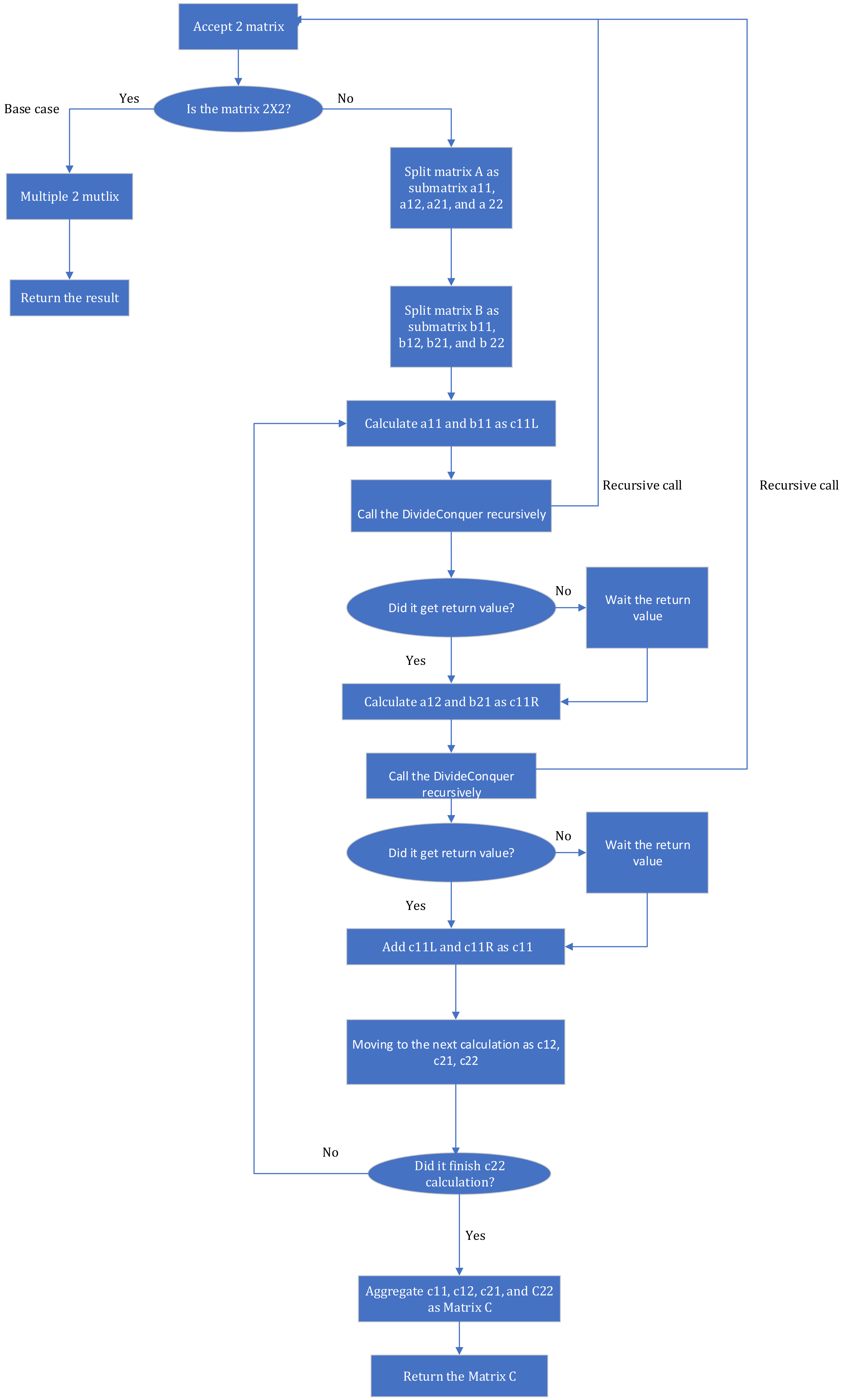
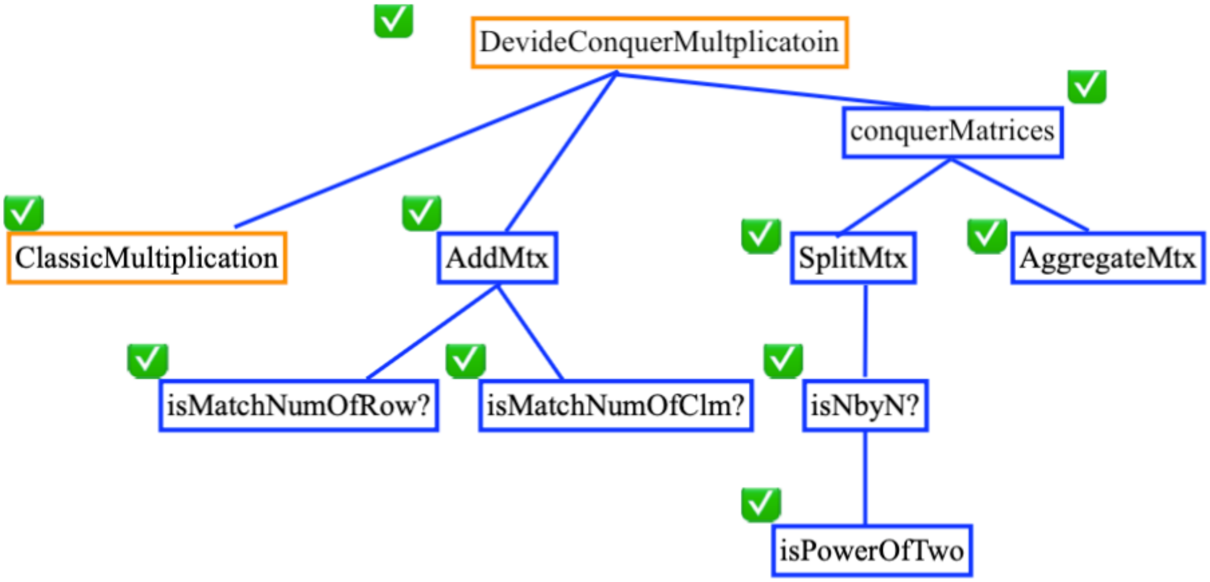

Accept 2 matrix
Is the matrix 2X2?
Yes
No
Base case
Multiple 2 mutlix
Return the result
Split matrix A as submatrix a11, a12, a21, and a 22
Split matrix B as submatrix b11, b12, b21, and b 22
Calculate a11 and b11 as c11L
Call the DivideConquer recursively
Recursive call
Recursive call
Did it get return value?
No
Wait the return value
Yes
Calculate a12 and b21 as c11R
Call the DivideConquer recursively
Did it get return value?
No
Wait the return value
Yes
Add c11L and c11R as c11
Moving to the next calculation as c12, c21, c22
No
Did it finish c22 calculation?
Yes
Aggregate c11, c12, c21, and C22 as Matrix C
Return the Matrix C

Algorighm b, Divide conquer Multiplication pseudocode.



Let ClassicMultiplication // Method a)

//Input: two matrix
//Process: add two matrix
//Output: sum of matrix

AddMtx

```
0. Start
1. Accept two matrix, mtxA and mtxB
2. Check mtxA and mtxB has the same number of row // Call isMatchNumOfRow
        If not, return null
3. Check mtxA and mtxB has the same number of column //Call isMatchNumOfClm
        If not, return null
4. Create a new matrix, mtxC
5. Set loop which iterates form the first row to the last row of mtxA and mtxB
    a. Set loop which iterates from the first column to the last columns to mtxA and mtxB
        i. Sum value in the corresponding row and column of mtxA and mtxC
                Eg. Sum = mtxA[0][0] + mtxB[0][0]
        ii. Store the result to mtxC corresponding row and column
6. Return mtxC
7. Stop
```

```
//Input: matrix
//Process: Extract left top, right top, left bottom, and right bottom partition value and copy
to 4 sub matrices.
//Output: 4 sub matrices container with array list
SplitMtx
0.  Start
1.  Accept an original matrix,
2.  Check the original matrix is N^2 by N^2 // Call NbyN
3.  Create 4 sub matrices size by N/2 by N/2

//Make left top partition a11
4.  Set loop which iterates from the first row to the n/2 row
        a.  Set loop which iterates from the first column to the n/2 column
                i.  Extract value from the original matrix and copy to the empty matrix

//Make right top partition a12
5.  Set loop which iterates from the first row to the n/2 row
        a.  Set loop which iterates from the n/2 column to the last column.
                i.  Extract value from the original matrix and copy to the empty matrix

//Make left bottom partition a21
6.  Set loop which iterates from the n/2 row to the last row
        a.  Set loop which iterates from the first column to the n/2 column
                i.  Extract value from the original matrix and copy to the empty matrix

//Make right bottom partition a22
7.  Set loop which iterates from the n/2 row to the last row
        a.  Set loop which iterates from the n/2 row to the last column.
                i.  Extract value from the original matrix and copy to the empty matrix // a22

8.  Create 4 sub matrices container // Using ArrayList to aggregate 4 sub matrices.
9.  Add a11 to the container.
10. Add a12 to the container.
11. Add a21 to the container.
12. Add a22 to the container.

13. Return sub matrices container.
```

```
//Input: two matrix
//Process: check the matrix column and row is N^2
//Output: bool
isNbyN
0.  Start
1.  Accept two matrix
2.  Check the number of row is power of N // Call isPowerOfTwo
            If not, return false
3.  Check the number of column is power of N // Call is PowerOfTwo
            If not, return false
4.  Check the number of row and column is the same
            If not, return false
5.  Return true
```

```
//Input: integer
//Process: check whether the integer is power of 2 or nor.
//Output: bool
isPowerOfTwo
0.  Start
1.  Accept integer
2.  Check the log base 2 integer becomes integer? Log base 2 to 4 => 2
3.  Return  true or not
4.  Stop
```

```
//Input: 4 submatrices array list
//Process: Iterates 4 submatrices vector and copy the value to the new integrated matrix
//Output: new matrix
AggregateMtx
0.  Start
1.  Accept 4 sub matrices arraylist.
2.  Check the 4 matrix's number of row and column are the power of 2
            //Call isPowerOfTwo
3.  Add the number of two matrix's column for the new matrix column.
```

```
4. Check the 4 matrix’s number of row and column are the power of 2
//Call isPowerOfTwo
5. Create a new matrix by the sum of row and sum of column.

//Transfer value to the left top partition a11
6. Set loop which iterates from the first row to the n/2 row in the new matrices.
    a. Set loop which iterates from the first column to the n/2 column.
        i. Extract value from the sub matrix and copy to the new matrix

//Transfer value to the right top partition a12
7. Set loop which iterates from the n/2 row to the last row in the new matrices.
    a. Set loop which iterates from the n/2 column to the last column.
        i. Extract value from the sub matrix and copy to the new matrix

//Transfer value to the left bottom partition a21

8. Set loop which iterates from the n/2 row to the last row in the new matrices.
    a. Set loop which iterates from the n/2 row to the last row.
        i. Extract value from the sub matrix and copy to the new matrix

//Transfer value to the right bottom partition a22
9. Set loop which iterates from the n/2 row to the last in the new matrices.
    a. Set loop which iterates from the n/2 row to the last column.
        i. Extract value from the sub matrix and copy to the new matrix

10. Return the new matrix.
11. Stop

//Input: 2 matrices which contains 4 submatrices
//Process: Integrate  total 8 sub matrices to 1 matrix with addition and multiplicaiton
//Output: 1 matrix
conquerMatrices
//Make C11
0. Calculate a11 and b11 as c11L // Recursive call DevideConquerMultiplication(a11, b11)
1. Calculate a12 and b21 as c11R// Recursive call DevideConquerMultiplication(a12, b21)
2.  Add c11L and c11R as c11 // Call AddMtx(c11L, c11R)

//Make C12
3. Calcluate a11 and b12 as c12L // Recursive call DevideConquerMultiplication(a11, b12)
```

```
4.  Calculate a12 and b22 as c12R// Recursive call DevideConquerMultiplication(a12, b22)
5.   Add c12L and c12R as c12 // Call AddMtx(c12L, c12R)

//Make C21
6.  Calculate a21 and b11 as c21L // Recursive call DevideConquerMultiplication(a21, b11)
7.  Calculate a22 and b21 as c21R// Recursive call DevideConquerMultiplication(a22, b21)
8.   Add c21L and c21R as c21 // Call AddMtx(c21L, c21R)

//Make C22
9.  Calculate a21 and b12 as c22L // Recursive call DevideConquerMultiplication(a21, b12)
10. Calculate a22 and b22 as c22R// Recursive call DevideConquerMultiplication(a22, b22)
11.  Add c22L and c22R as c22 // Call AddMtx(c22L, c22R)

//Matrix C
12. Aggregate sub matrix c11,  c12, c21 and c22 //Call AggregateMtx(c11, c12, c21,c22)
13. Return mtxC
14. Stop
```

```
//Input: 2 matrix
//Process: multiple two matrix with splitting and combine submatrix recursively
//Output: 1 matrix
```

<u>DevideConquerMultiplicatoin</u>

```
0.  Start
1.  Accept 2 matrix, mtxA and mtxB
2.  If the size of mtxA and size of mtxB are 2 // Base case
            Calculate 2 by 2 matrix // Call classicMultiplicatoin
            Return the new matrix C, mtx_c
    else
            Split mtxA into sub matrix as a11, a12, a21, and a22 // Call SplitMtx
            Split mtxB into submatrix as b11, b12, b21, and b22 // Call SplitMtx
            Conquer mtxA and mtxB calling recursively into mtxC// Call conquerMatrices
            Return mtxC
3.  Stop
```