Exercise 8. Answer Sheet

Student's Name: Yuta Nemoto Student's ID: s1240234

Problem 1. Write pseudo-code for the Strassen's algorithm.

```
STRASSEN(A, B)
// Input: A, B - n x n matrix
// Output: C - n x n matrix
n = |A.rows|
C = new (n×n) matrix
if n == 1
        C11 = A11 * B11
else
        /* Calculate the sum matrices */
        S1 = B12 - B22
        S2 = A11 + A12
        S3 = A21 + A22
        S4 = B21 - B11
        S5 = A11 + A22
        S6 = B11 + B22
        S7 = A12 - A22
        S8 = B21 + B22
        S9 = A11 - A21
        S10 = B11 + B12
        /* Calculate the product matrices */
        P1 = STRASSEN(A11, S1)
        P2 = STRASSEN(S2, B22)
        P3 = STRASSEN(S3, B11)
        P4 = STRASSEN(A22, S4)
        P5 = STRASSEN(S5, S6)
        P6 = STRASSEN(S7, S8)
        P7 = STRASSEN(S9, S10)
        /* Calculate the final product sub matrices */
        C11 = P5 + P4 - P2 + P6
        C12 = P1 + P2
        C21 = P3 + P4
        C22 = P1 + P5 - P3 - P7
return C
```

Problem 2. Use Strassen's algorithm to compute the matrix product:

$$\begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix} \begin{pmatrix} 6 & 8 \\ 4 & 2 \end{pmatrix}$$

Show your work below:

Let A = $\begin{pmatrix} 1 & 3 \\ 7 & 5 \end{pmatrix}$, B = $\begin{pmatrix} 6 & 8 \\ 4 & 2 \end{pmatrix}$.

To calculate the sum matrices,
$S_1 = B_{12} - B_{22} = 8 - 2 = \mathbf{6}$
$S_2 = A_{11} + A_{12} = 1 + 3 = \mathbf{4}$
$S_3 = A_{21} + A_{22} = 7 + 5 = \mathbf{12}$
$S_4 = B_{21} - B_{11} = 4 - 6 = \mathbf{-2}$
$S_5 = A_{11} + A_{22} = 1 + 5 = \mathbf{6}$
$S_6 = B_{11} + B_{22} = 6 + 2 = \mathbf{8}$
$S_7 = A_{12} - A_{22} = 3 - 5 = \mathbf{-2}$
$S_8 = B_{21} + B_{22} = 4 + 2 = \mathbf{6}$
$S_9 = A_{11} - A_{21} = 1 - 7 = \mathbf{-6}$
$S_{10} = B_{11} + B_{12} = 6 + 8 = \mathbf{14}$

To calculate product matrices,
$P_1 = A_{11}S_1 = 1 * 6 = \mathbf{6}$
$P_2 = S_2B_{22} = 4 * 2 = \mathbf{8}$
$P_3 = S_3B_{11} = 12 * 6 = \mathbf{72}$
$P_4 = A_{22}S_4 = 5 * (-2) = \mathbf{-10}$
$P_5 = S_5S_6 = 6 * 8 = \mathbf{48}$
$P_6 = S_7S_8 = (-2) * 6 = \mathbf{-12}$
$P_7 = S_9S_{10} = (-6) * 14 = \mathbf{-84}$

To calculate the final product sub matrices,
$C_{11} = P_5 + P_4 - P_2 + P_6 = 48 - 10 - 8 - 12 = \mathbf{18}$
$C_{12} = P_1 + P_2 = 6 + 8 = \mathbf{14}$
$C_{21} = P_3 + P_4 = 72 - 10 = \mathbf{62}$
$C_{22} = P_1 + P_5 - P_3 - P_7 = 6 + 48 - 72 + 84 = \mathbf{66}$

Then, the result of C = A * B is

C = $\begin{pmatrix} \mathbf{18} & \mathbf{14} \\ \mathbf{62} & \mathbf{66} \end{pmatrix}$

Problem 3. Make two programs implementing the Recursive matrix multiplication and the Strassen's algorithm. Upload your code. Generate two random matrices A and B of size n×n, multiply them using your programs and measure the time needed to get the result. Fill the following table:

Time needed to multiply two n×n matrices. (May depend on the programming language, computer, etc.)

Algorithm	n					
	32	64	128	256	512	1024
Recursive (sec)	0.021	0.130	0.820	5.678	46.595	407.868
Strassen (sec)	0.033	0.182	1.086	7.022	50.042	340.350

How to compile/run:

1. For the Recursive matrix multiplication code, execute the following:
   javac RecursiveMatrixMultiplication.java
   java RecursiveMatrixMultiplication [n]

Actual interface is like the screenshot below.

```
std6dc33{s1240234}108: javac RecursiveMatrixMultiplication.java
std6dc33{s1240234}109: java RecursiveMatrixMultiplication 32
Ellapsed time: 21ms
std6dc33{s1240234}110: java RecursiveMatrixMultiplication 64
Ellapsed time: 130ms
std6dc33{s1240234}111: java RecursiveMatrixMultiplication 128
Ellapsed time: 820ms
std6dc33{s1240234}112: java RecursiveMatrixMultiplication 256
Ellapsed time: 5678ms
std6dc33{s1240234}113: java RecursiveMatrixMultiplication 512
Ellapsed time: 46595ms
std6dc33{s1240234}114: java RecursiveMatrixMultiplication 1024
Ellapsed time: 407868ms
```

2. For the Strassen's algorithm code, execute the following:

   javac StrassenAlgorithm.java
   java StrassenAlgorithm [n]

   Actual interface is like the screenshot below.

```
std6dc33{s1240234}120: javac StrassenAlgorithm.java
std6dc33{s1240234}121: java StrassenAlgorithm 32
Ellapsed time: 33ms
std6dc33{s1240234}122: java StrassenAlgorithm 64
Ellapsed time: 182ms
std6dc33{s1240234}123: java StrassenAlgorithm 128
Ellapsed time: 1086ms
std6dc33{s1240234}124: java StrassenAlgorithm 256
Ellapsed time: 7022ms
std6dc33{s1240234}125: java StrassenAlgorithm 512
Ellapsed time: 50042ms
std6dc33{s1240234}126: java StrassenAlgorithm 1024
Ellapsed time: 340350ms
```

3. If you want to check the actual result of the matrix calculation, you can check it by adding "-CHECK" to the second argument like below.

```
std6dc33{s1240234}128: java RecursiveMatrixMultiplication 4 -CHECK
Initial Matrixes:
Matrix A:
1 3 5 9
2 7 7 0
5 3 3 3
9 1 8 9
Matrix B:
2 9 8 1
7 2 1 2
0 9 1 8
6 4 8 3
Result of multiplied matrix C:
77 96 88 74
53 95 30 72
49 90 70 44
79 191 153 102
Ellapsed time: 0ms
std6dc33{s1240234}129: java StrassenAlgorithm 4 -CHECK
Initial Matrixes:
Matrix A:
4 6 1 4
3 1 3 5
6 5 3 7
4 3 3 9
Matrix B:
7 5 8 0
8 7 8 6
7 4 3 7
5 2 2 0
Result of multiplied matrix C:
103 74 91 43
75 44 51 27
138 91 111 51
118 71 83 39
Ellapsed time: 0ms
std6dc33{s1240234}130:
```