In this assignment, you will solve some simple problems that teach you to design and analyze parallel algorithm using the techniques you have learnt in class.

**Problem 1-1. The Stock Market Problem**[40 points]

The problem with the stock market is that, while it is possible to make a great deal of money buying and selling stocks, it’s easy to lose even more. The long-standing—if somewhat unhelpful—maxim to make more money than you lose is “buy low, sell high.”

The stock market problem is finding the best opportunity to follow this advice: for any sequence of integer prices, where the index in the sequence represents time, find maximum jump from an earlier price to a later price. For example, if the sequence of prices was

\[ \langle 40, 20, 0, 0, 0, 1, 3, 3, 0, 0, 9, 21 \rangle \]

then the maximum jump is 21, which happens between the price at time 2 and time 11. More formally, the stock market problem is to compute

\[ \max \{ s_j - s_i \mid 0 \leq i \leq j < |s| \} \]

(a) **Naive Solution** [5 points]

Describe a simple quadratic work solution to the stock market problem and analyze its work and span.

(b) **Divide and Conquer** [20 points]

Solve the stock market problem by divide-and-conquer recursive programming and analyze its work and span.

(c) **Analysis** [15 points]

Use mathematical induction on the length of the input sequence to prove that your divide and conquer implementation is correct. Be sure to carefully state the theorem that you’re proving and to note all the steps in your proof. Remember that a longer proof is not necessarily a more correct proof.

**Problem 1-2. Work/Span Analysis of parallel_for** [20 points]

As an exercise, you were supposed to analyze the work and span the following code that uses a parallel for loop. In class, we saw that we can convert a parallel for loop into code that just uses spawns and syncs using a divide and conquer algorithm. This creates an additive span of \( O(\lg n) \). You will now analyze two other variants for parallelizing this loop.
(a) An Intel engineer comes to you tells you that they have decided to make a parallel
for loop more efficient by doing a divide by 3 divide-and-conquer instead of divide-
by-two as you learned in class. Give the pseudocode for this new version of parallel
for loop and analyze it. Does this modification improve the work and span asymptoti-
cally?

(b) We are now going to coarsen the base case. Analyze the work and span of the follow-
ing computation in terms of \( G \) and \( n \).

\[
\text{COPYARRAY}(A, B, \text{start}, \text{end})
\]

1. \textbf{if} \( \text{end} - \text{start} < G \)
2. \hspace{1em} \textbf{then for} \( j \leftarrow \text{start} \) \textbf{to} \( \text{end} \)
3. \hspace{2em} \textbf{do} \( B[j] \leftarrow A[j] \)
4. \hspace{1em} \textbf{return}

5. \( \text{mid} \leftarrow \lfloor (\text{start} + \text{end})/2 \rfloor \)
6. \textbf{spawn} \text{COPYARRAY}(A, B, \text{start}, \text{mid})
7. \text{COPYARRAY}(A, B, \text{mid} + 1, \text{end})
8. \textbf{return}

**Problem 1-3. Strassen’s Algorithm** [15 points]

We learned strassen’s algorithm for matrix multiplication in class. Now you will write the pseu-
docode and analyze the algorithm’s work and span. You want to compute \( C = A \times B \), where \( A \),
\( B \) and \( C \) are all \( n \times n \) matrices. We first divide them into 4 parts, \( A_{11}, A_{12}, A_{21}, A_{22} \) etc. Then we
can calculate the matrix \( C \) using the following mathematical formulation:

\[
\begin{align*}
M_1 &= (A_{11} + A_{22}) \times (B_{11} + B_{22}) \\
M_2 &= (A_{21} + A_{22}) \times B_{11} \\
M_3 &= A_{11} \times (B_{12} - B_{22}) \\
M_4 &= A_{22} \times (B_{21} - B_{11}) \\
M_5 &= (A_{11} + A_{12}) \times B_{22} \\
M_6 &= (A_{21} - A_{11}) \times (B_{11} + B_{12}) \\
M_7 &= (A_{12} - A_{22}) \times (B_{21} + B_{22})
\end{align*}
\]
Given these matrices, you can now compute the 4 parts of the final matrix.

\[
C_{11} = M_1 + M_4 - M_5 + M_7 \\
C_{12} = M_3 + M_5 \\
C_{21} = M_2 + M_4 \\
C_{22} = M_1 - M_2 + M_3 + M_6
\]

Given this formulation, provide the pseudocode for Strassen's. There is more than 1 correct answer, and all answers with the correct work and reasonable span will get credit.

**Problem 1-4. Sum your ancestors [25 points]**

You are given a complete binary tree of height \( h \) with \( n = 2^h \) leaves, where each node (internal nodes and leaves) of the tree has an associated value \( v \) (which is an arbitrary real number).

If \( x \) is a leaf, we denote by \( A(x) \), the set of ancestors of \( x \) (including \( x \) as one of its own ancestors). That is, \( A(x) \) contains \( x \), \( x \)'s parent, \( x \)'s grandparent, etc. up to the root of the tree. Similarly, if \( x \) and \( y \) are distinct leaves, we denote \( A(x, y) \) as the ancestors of either \( x \) or \( y \). That is \( A(x, y) = A(x) \cup A(y) \). We define a function \( f(x, y) \) as the sum of all values of the nodes in \( A(x, y) \). An example is shown in Figure 1.

Give a parallel algorithm that efficiently finds two leaves \( x_0 \) and \( y_0 \) such that \( f(x_0, y_0) \) is as large as possible (largest among all the pairs of leaves). Analyze the work and span of your algorithm.
Figure 1: $A(x, y)$ is shown in bold. $f(x, y) = 19 + 15 + 36 + 10 + 27 + 30 = 137$. For this tree, the value of $f(x, y)$ is maximized for the leaves $x$ and $y$. 