Notes: (i) List the other students with whom you discussed the problems on this assignment: such collaboration is allowed, but you need to write up your solutions by yourself. If you did not discuss any problems with your classmates, please indicate this. Consulting other sources (including the Web) is not allowed. (ii) Write your solutions neatly; if you are able to make partial progress by making some additional assumptions, then state these assumptions clearly and submit your partial solution.

1. Given a stock price over the last $t$ days, we would like to write a parallel algorithm to compute the maximum profit that can be made by buying a stock on one day, and selling on a later day. For example:

   \[ \text{BestTrade([600, 540, 30, 2, 1, 1, 2, 1, 1, 3, 2, 6]) = 5} \]

   since we can buy on day 5 and sell on day 12. The program should have work $O(t)$ and depth $O(\log t)$ on the BF model. (10 points)

2. Fibonacci Numbers. Design a parallel algorithm for computing the first $n$ Fibonacci numbers $F_1, F_2, \ldots, F_n$ in $O(n)$ work and $O(\log n)$ depth. Hint: consider the array $A$ where $A_i$ stores the vector \[ \begin{bmatrix} F_{i-1} \\ F_i \end{bmatrix} \], and the transition matrix that computes $A_{i+1}$ from $A_i$. (10 points)

3. Finding the maximum. Given an array $A$ of length $n$ elements, we would like to find the maximum element by performing comparisons between pairs of elements. Assume that we have a comparison function $\max(x, y)$ which returns the maximum of elements $\{x, y\}$. Clearly this problem can be solved on the BF model in $O(n)$ work and $O(\log n)$ depth, e.g., by calling reduce, or performing a scan operation with the max operator. In the following, we will see how to solve the problem in lower depth on the MP-RAM model.

   1. Briefly describe an algorithm $\text{FindMaxInefficient}$ which solves the problem on the MP-RAM using $O(n^2)$ work and $O(1)$ depth. (5 points)

      You may need to use the test-and-set (TS) instruction, which is defined as follows: TS is an atomic instruction that reads a memory location and if the memory location is zero, sets it to one, returning zero. Otherwise it leaves the value unchanged, returning one. Note that most processors today support the TS instruction in hardware. Please assume that any number of concurrent TS operations to a given memory location can be performed in unit depth.

   2. Although the previous algorithm is highly parallel, it is not work-efficient. Yet, it can still be potentially useful. For example, if we only had $O(\sqrt{n})$ candidates for the maximum, we can solve this subproblem in $O(n)$ work and $O(1)$ depth. This motivates the following algorithm:

      \[
      \text{FindMax}(A, \max) \\
      \text{if} \ (A = []) \ \text{return} \ \bot; \\
      \text{if} \ (|A| = 1) \ \text{return} \ A[0] \\
      \text{else:} \\
      \quad n := |A| \\
      \quad m := \sqrt{n} \\
      \quad R := \text{alloc}(m) \\
      \quad \text{parfor i in [0 : m]:} \\
      \quad \quad s := \min(i * m, n) \\
      \quad \quad e := \min((i+1) * m, n) \\
      \quad \quad R[i] := \text{FindMax}(A[s : e], \max) \\
      \quad \text{return} \ \text{FindMaxInefficient}(R, \max)
      \]

      where $\bot = \bot$ is a special element s.t. $\max(\bot, x) = \max(x, \bot) = x$. Show that this algorithm runs in $O(n \log \log n)$ work and $O(\log \log n)$ depth on the MP-RAM. Hint: when analyzing the work recurrence, you can try rewriting the recurrence in terms of $k = \log n$. (10 points)
3. Design a max-finding algorithm which runs in $O(n)$ work and $O(\log \log n)$ depth on the MP-RAM. 

*Hint:* try reducing the size of the input before calling the algorithm `FindMax` above. (5 points)