CODE: .................................................................

Grade: 1: ... 2: ... 3: ... 4: ... 5: ... Total: .......

Solve all 5 (FIVE) problems in the exam books/sheets provided

Note: Be concise in describing your algorithms. If you use a known algorithm, then you can use it as a black-box (subroutine) without going into details of how that known algorithm works. Avoid useless ramblings about known algorithms, as that will be viewed negatively!

Definition (Little oh): For functions $f(n), g(n)$, we say $f(n)$ is $o(g(n))$, if and only if:

$$\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0.$$ 

For example, $\log n$ is $o(n)$, but $n/2$ is not $o(n)$. Sometimes we say $f(n)$ is asymptotically smaller than $g(n)$.

Definition (log base two): We define $\lg n$ to be the logarithm base two of $n$. For example, $\lg 2$ is one, and $\lg 8$ is 3.

Definition (Stirling’s approximation formula): For $n$ factorial denoted by $n!$, the following approximation can be used in this exam without proof: $n! = (n/e)^n$, where $e \approx 2.7182\ldots$, and thus the following can then be derived: $\lg (n!) = n \lg n - 1.44n$. 
Problem 1. (20 points)
You may assume that \( n \log n \) and \( n^2 \) are both integer in this problem. You are given two sorted sequences stored in arrays \( A \) and \( B \) of \( n \log n \) and \( n^2 \) keys respectively. We are interested in forming the union \( C = A \cup B \), i.e. an array \( C \) containing one copy of all the keys of \( A \) and \( B \). Give a space and time efficient algorithm for solving this problem. (Time efficient means its running time cannot be asymptotically improved upon; space efficient means likewise for space requirements.) Justify your arguments.

Problem 2. (20 points)
We have two types of scales to weigh coins.

1. A balance scale. One coin can be put on one side and another on the other side and the scale will tell us whether the two sides weigh the same or which side weighs more (or less) by pointing out the lightest coin. Thus if we have two coins \( a, b \) on either side, a function \( \text{LIGHTEST}(a, b) \) returns the coin that is lighter (i.e. \( a \) or \( b \)) or \(-1\) if both weigh the same.

2. A triple side scale. Such a scale also accepts only one coin on each side and points to the side of \( \text{MEDIAN}(a, b, c) \). That is if three coins \( a, b, c \) are placed one per side, and they are \( a < b < c \), \( \text{MEDIAN}(a, b, c) \) will return \( b \), i.e. the scale will point towards \( b \).

From now on we call a scale by its function: \( \text{LIGHTEST} \) and \( \text{MEDIAN} \). We have \( n \) coins \( c_1, \ldots, c_n \) all with different weights.

(a) Give a short argument that one cannot use just the \( \text{MEDIAN} \) scale to distinguish between the lightest and heaviest of \( n > 1 \) coins; however the \( \text{LIGHTEST} \) scale can do so.

(b) If you know the lightest coin \( L \) (i.e. the index \( l \) such that \( c_l = L \)), show how you can determine for two arbitrary coins \( a \) and \( b \) (other than \( L \)) which one is lighter by using only \( \text{MEDIAN} \) and not \( \text{LIGHTEST} \).

(c) For the \( n \) coins \( c_1, \ldots, c_n \) use \( o(n \log n) \) calls to \( \text{MEDIAN} \) to determine the lightest and heaviest coins \( L \) and \( H \). Your algorithm need only return the \( L \) and \( H \); it does not need to clearly identify which of the two is the lightest and which one is the heaviest.

Problem 3. (20 points)
You are given two Binary Search Trees \( T_1, T_2 \) with \( n_1 \) and \( n_2 \) keys respectively, where \( n = n_1 + n_2 \). Build a balanced binary search \( T \) of the \( n \) keys of \( T_1, T_2 \) in worst-case linear time. Prove your claims. (You may create an AVL or red-black tree, or just a regular binary search tree that is balanced.)

Problem 4. (20 points)
Given two sequences of numbers \( X = \langle x_1, \ldots, x_n \rangle \) and \( Y = \langle y_1, \ldots, y_n \rangle \) he must find the \( m \leq \sqrt{n} \) largest values of the sums \( x_i + y_j, 1 \leq i, j \leq n \) in SORTED ORDER (smallest to largest). One may assume that \( m \leq \sqrt{n} \) but one must not assume that \( m \) is a constant. The proposed algorithm should have worst-case running time that is \( o(n \log n) \).

Example. If \( X = \langle -2, -3, -1, 0, 100, 290, 300, 4 \rangle \) and \( Y = \langle -200, -189, -18, -100, 11, 1, 10, -7 \rangle \) and \( m = 3 \) then the answer is \( \langle 301, 310, 311 \rangle \). Note that 301 can result either as the sum of 290 + 11 or 300 + 1.

The fifth problem is on the following page.
Problem 5. (20 points)
We have a directed graph $G = (V, E)$ enhanced with weight values on its edges i.e. $G = (V, E, w)$. For an edge $(i, j) \in E$, the weight $w(i, j)$ is a real number greater than 0 and less than 1.
For a given source vertex $s$ we want to determine $B(s, u)$ where $B(s, u)$ is the maximum product-weight of any path from $s$ to $u$. Thus we want to maximize $\prod_{j=1}^{t} w(v_{j-1}, v_j)$ for every path $v_0 = s, v_1, \ldots, v_{t-1}, v_t = u$ from $s$ to every $u$ other than $s$. Give an algorithm that solves this problem and analyze its worst-case running time.