Name-Surname: …… ID: ……

13016239 ALGORITHM DESIGN AND ANALYSIS
MIDTERM EXAMINATION 2015/2

International College

King Mongkut's Institute of Technology Ladkrabang

DIRECTIONS:

- You are allowed to bring up to ONE A4 sheets of note, handwritten by yourself to the exam. Any other document and electronic peripherals (e.g. calculators, mobile phones, media players, and computers) are not allowed in the exam room.
- Please write your answers in the accompanied answer book. There is no need to write the question statements in the answer book, but please clearly indicate in front of each of your answers which question the answer is for.
- Anything written in this exam paper will not be considered part of your answer.
- Before submission, please check that you have written you name and student ID clearly on all your answer booklets and also on this exam paper.

PROBLEM 1 (10 PTS)

Prove or disprove each of the following statements.

1.1 $5n + 2 \in O(n)$

1.2 $n^2 + 2n - 1 \in \Omega(n)$

1.3 $\sum_{i=1}^{n} i \in \Theta(n^2)$

1.4 $\log_2 n \in O(\log_3 n)$

1.5 $3^n \in O(2^n)$

PROBLEM 2 (5 PTS)

Analyze the time complexity of the following pseudocode fragment, where m is the size of the input. Describe its running time using Big-O or Big-Θ notations.

```
1: s = 0
2: for(i = 1 to m)
3:     for(j = 1 to m)
4:         s = s + i*j
```

PROBLEM 3 (5 PTS)

Analyze the time complexity of the following pseudocode fragment, where m is the size of the input. Describe its running time using Big-O or Big-Θ notations.

```
1: s = 0
2: for(i = 1 to m)
3:     for(j = i to m)
4:         s = s + i*j
```

PROBLEM 4 (5 PTS)

Suppose the running time of the function `fac(`n`)`, where n is a non-negative integer, is $\Theta(n)$. Analyze the time complexity of the following pseudocode fragment, where m is the size of the input. Describe its running time using Big-O or Big-Θ notations.

```
1: i = 1; j = m; s = 0
2: while(j > 0) {
3:     s = s + fac(i)
4:     j = j-i
5:     i = i+1
6: }
```

PROBLEM 5 (10 PTS)

Consider the following pseudocode description of the bubble sort algorithm. The function `swap(A, i, j)`, which swaps the item at index `i` with the item at index `j` in array `A`, uses constant time.

```
 1: BubbleSort(A[1..n]) {
 2:     changed = true
 3:     m=n
 4:     while(changed && m>1) {
 5:         changed = false
 6:         for j = 2 to m {
 7:             if(A[j-1]>A[j]) {
 8:                 swap(A,j-1,j)
 9:                 changed = true
10:             }
11:         }
12:         m = m-1
13:     }
14: }
```

5.1 Analyse the best-case time complexity of this function. Describe its best-case running time using asymptotic notations, where n is the size of the input array. When does the best-case scenario occur?

5.2 Analyse the worst-case time complexity of this function. Describe its worst-case running time using asymptotic notations, where n is the size of the input array. When does the worst-case scenario occur?

PROBLEM 6 (10 PTS)

Consider the following pseudocode description of a strange sorting algorithm. Analyze the time complexity of `StrangeSort(A)` in terms of the size of the given array `A`. Describe its running time using Big-O or Big-Θ notations.

To simplify the analysis, assume that the array size, n, is a power of 3 ($n = 3^m$, for some non-negative integer m). Also assume that the instruction `merge(A, i, j)` on Line 13 uses $\Theta(j-i+1)$ seconds and other primitive instructions use constant time.

```
 1: StrangeSort(A[1…n]) {
 2:     StrangeSort(A, 1, n)
 3: }
 4:
 5: StrangeSort(A[1…n], i, j) {
 6:     if (i >= j) return
 7:     k = (j-i+1)/3
 8:     if (k > 1) {
 9:         StrangeSort(A, i, i+k-1)
10:         StrangeSort(A, i+k, j-k)
11:         StrangeSort(A, j-k+1, j)
12:     }
13:     merge(A, i, j)
14: }
```

PROBLEM 7 (10 PTS)

Solve the following recurrence equations. Describe the solutions as closed-form formulas, possibly using asymptotic notations.

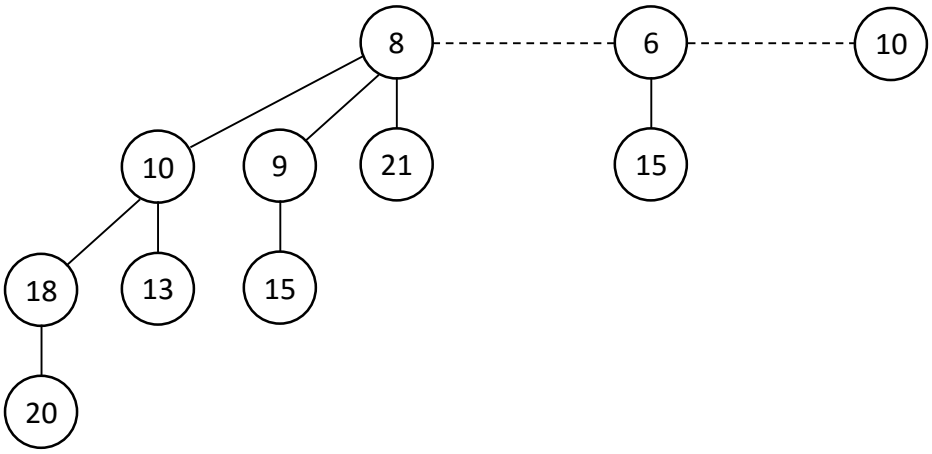
7.1 $f(0) = 1$
$f(n) = f(n-1) + 2^n + n - 1$, for $n \geq 1$.

7.2 $g(1) = 1$
$g(n) = 2g\left(\frac{n}{2}\right) + n^2$, for $n > 1$ and n is a power of 2.

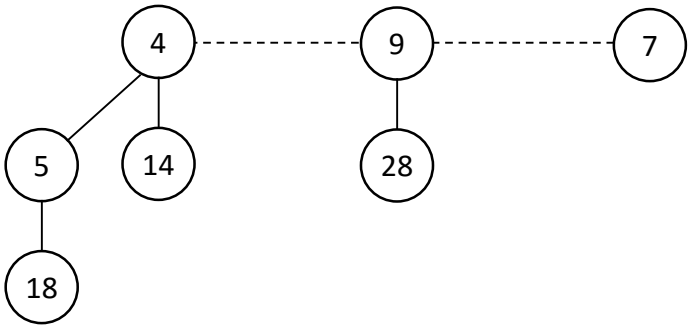
PROBLEM 8 (10 PTS)

Suppose H_1 and H_2 are the binomial heaps given below. Find the binomial heap resulting from melding (i.e. merging) H_1 and H_2.

H_1:



H_2:



PROBLEM 9 (10 PTS)

Describe an algorithm which, given an unsorted integer array of size n ($1{,}000 \le n \le 10{,}000{,}000$) and a positive integer k ($1 \le k \le 100$), computes the average of the k largest integers in the array. You may assume that the integers in the given array are all distinct, i.e. no integers occurring more than once. Your algorithm should have a worst-case time complexity of O(n log k). Describe your algorithm by means of pseudocode and analyze its worst-case time complexity.

PROBLEM 10 (10 PTS)

Recall that a standard (FIFO) queue maintains a sequence of items subject to the following operations.

- ENQUEUE(Q, x): Add item x to the end of queue Q.
- DEQUEUE(Q): Remove and return the item at the beginning of queue Q; return NULL if Q is empty.
- SIZE(Q): Return the number of items currently stored in queue Q.

It is easy to implement a queue using a doubly-linked list, so that it uses O(n) space (where n is the number of items in the queue) and the worst-case time for each of these operations is O(1).

Consider the following new operation, which removes every tenth element from the queue, starting at the beginning, in O(n) worst-case time.

```
1:  DECIMATE(Q) {
2:      n = SIZE(Q)
3:      for i = 0 to n-1
4:          if (i mod 10 == 0)
5:              DEQUEUE(Q)
6:          else
7:              ENQUEUE(Q,DEQUEUE(Q))
8:  }
```

Prove that, for any sequence of m operations (where each operation the sequence is either ENQUEUE, DEQUEUE, or DECIMATE), the total running time for the sequence is O(m).

Note:

1. You may assume that there is unlimited amount of memory available to store a queue.
2. (Cost model) Each enqueue or dequeue operation costs 1 credit.

This is the end of the exam paper.