Name-Surname: ………………………………………………………………………………………………… ID: ……………………………………………

13016239 Algorithm Design and Analysis
Midterm Examination 2018/2

International College
King Mongkut's Institute of Technology Ladkrabang

Directions:

- No books, documents, or electronic devices are allowed.
- 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 <u>not</u> 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 (16 pts)

Prove or disprove each of the following statements.

1.1 $3n + 10 \in O(n)$ $\rightarrow$ $3n+10 \le 3n+10n$; $3n+10 \le 13n$; $C=13$ $k=1$ $\rightarrow$ $3n+10 \in O(n)$ when $c=13$, for $n \ge 1$

1.2 $3n + 10 \in O(n^2)$ $\rightarrow$ $3n+10 \le 3n^2+10n^2$; $|3n+10| \le |13n^2|$; $C=13$ $k=1$ $\rightarrow$ $3n+10 \in O(n^2)$ when $C=13$, for $n \geq 1$

1.3 $\log_3(n^2) \in \Theta(\log_2(n^3))$ $\longrightarrow$

1.3 $\log_3(n^2) \in \theta(\log_2(n^3))$
$C=1$ $d=2$
$C\log_2(n^3) \le \log_3(n^2) \le d\log_2(n^3)$
$3\log_2(n) \le 2\log_3(n) \le 6\log_2(n)$ $\rightarrow$

$\frac{3\log(n)}{\log(2)} \le \frac{2\log(n)}{\log(3)} \le \frac{6\log(n)}{\log(2)}$
$\frac{3}{\log(2)} \le \frac{2}{\log(3)} \le \frac{6}{\log(2)}$
$3\log(3) \le 2\log(2) \le 6\log(3)$ } False

1.4 $n\log^2 n \in O(n^2)$

$n\log^2 n \le c\, n^2$

$n\log^2 n \le n^2$

Problem 2 (8 pts)

Find closed-form solutions to the following recurrence equations. You may express your solutions using Big-Θ notation. You may apply the Master Theorem.

2.1 Let *g* be the function defined for all non-negative integers as follows:

$$g(0) = 0, g(1) = 1$$
$$g(n) = 3g\left(\left\lfloor\frac{n}{3}\right\rfloor\right) + n, \quad n > 1$$

$a=3$
$b=3$
$c=1$ $\quad 1 = \log_3{}^3 = 1$
$1 == 1$ $\rightarrow$ $\theta(n\log n)$

2.2 Let *h* be the function defined for all non-negative integers as follows:

$$h(0) = 0, h(1) = 1$$
$$h(n) = 3h\left(\left\lfloor\frac{n}{2}\right\rfloor\right) + n^2 + 2n, \quad n > 1$$

$a=3$
$b=2$
$c=2$ $\quad 2 > 1.58$
$\rightarrow$ $\theta(n^2)$

Problem 3 (4 pts)

Analyze the time complexity of the following pseudocode fragment, where m ≥ 0 is the size of the input. Describe its running time using Big-Θ notation.

```
s = 0
for(i = 0 to m)
    for(j = 0 to i)
        s = s + i*j
```

$\} \Theta(i)$

$\} \Theta(1) \} \sum_{0}^{i} \Theta(1) \} \Theta(i) \} \sum_{i=0}^{m} \Theta(i) = \frac{\Theta(m)(m+1)}{2}$

$= O(m)^2$

Problem 4 (4 pts)

Analyze the time complexity of the following pseudocode fragment, where m ≥ 1 is the size of the input, assuming that m is a power of 2. Describe its running time using Big-Θ notations.

```
j = m; s = 0;
while(j > 0) {
    for(i = 0 to j)
        s = s + 1
    j = ⌊j/2⌋
}
```

$\} \Theta = 1 \} \sum_{j=0}^{5} \Theta(1) = \Theta(5) \} \sum_{5=0}^{\log_2 m} \Theta(5)$

$= \Theta(5 \log_2 m)$

Problem 5 (8 pts)

Consider the pseudocode description of the function FunA(A) given below.

5.1 Analyze the best-case time complexity of FunA(A) in terms of the size of the given array A. What are the input arrays A in the best-case scenario? O(1) return false or True when i>n or 0 is in the first one.

5.2 Analyze the worst-case time complexity of FunA(A) in terms of the size of the given array A. What are the input arrays A in the worst-case scenario? O(n), 0 is at the end of array

```
FunA(A[1…n]) {
    return FunA(A, 1)
}

FunA(A[1…n], i) {
    if (i > n) return False
    if (A[i] == 0)
        return True
    else
        return FunA(A[1…n], i+1)
}
```

$\} i > n : O(1)$

$\} A[0] == 0$

$= O(1)$

best

$\} \sum_{i=0}^{n} O(1)$

$= O(n)$

worst

PROBLEM 6 (8 PTS)

Consider the pseudocode description of the function FunB(A) given below.

6.1 Analyze the worst-case time complexity of FunB(A) in terms of the size of the given array A. Describe its running time using Big-Θ notation. To simplify the analysis, assume that the array size, n, is a power of 2 ($n = 2^m$, for some non-negative integer m).

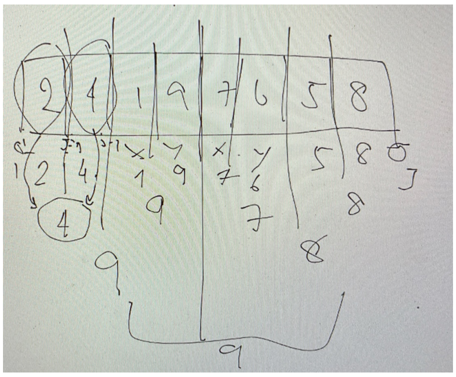
6.2 Give an alternative implementation of FunB(A) that does <u>not</u> use recursion.

```
FunB(A[1…n]) {
    return FunB(A, 1, n)
}

FunB(A[1…n], i, j) {
    if (i == j) return A[i]            Θ(1)
    d = ⌊(j-i+1)/2⌋                    Θ(1)
    x = FunB(A, i, i+d-1)  →  T(n/2)
    y = FunB(A, i+d, j)    →  T(n/2)
    if(x > y)
        return x                       Θ(1)
    else
        return y                       Θ(1)
}
```

$a = 2 \quad b = 2 \quad c = 0$

$$T(n) = T(\tfrac{n}{2}) + T(\tfrac{n}{2}) + \Theta(1)$$
$$= 2T(\tfrac{n}{2}) + \Theta(1)$$
$$= c < \log_b a$$
$$= \Theta(n^1)$$
$$= \Theta(n)$$



```
max = 0
for i in A:
    if i > max
        max = i
return max
```

PROBLEM 7 (8 PTS)

Suppose you are given an array A[1…n] (where n ≥ 3) containing n integers. For simplicity, assume all the integers in A are distinct, i.e. no number occurs twice in the array. You are told that the sequence of values A[1], A[2],...,A[n] is *unimodal*: There exists an index p, called the "*peak index*", where 1 < p < n, such that the values in the array entries increase up to position p in A and then decrease the remainder of the way until position n.

For example, the array A[1…7] = [3,15,23,24,48,35,29] is unimodal with the peak index p = 5.

7.1 Provide a pseudo-code description of an algorithm for the function `findPeak(A)` which, given a non-empty unimodal array A containing 3 or more distinct integers, returns the peak index of A. The function should have the worst-case running time of O(log n).

```
       1  Left = 0
       1  right = length(list)-1
       1  mid = left + right //2
          if List[mid-1] < mid > List[mid+1]
              return mid
       1  if length(list) == 1 return
       1  if list[mid -1] > list[mid]+1
  T(n/2)  {   return findpeak(A[0...half])
       1  if list[mid -1] < list[mid]+1
          {   return findpeak(A[half...n]
```

7.2 Give a worst-case time complexity analysis of your algorithm.

$$\left.\begin{array}{l} a = 1 \\ b = 2 \\ c = 0 \\ 0 = 0 \\ c = \log_b a \end{array}\right\} O(\log n)$$

PROBLEM 8 (8 PTS)

You are asked to design an algorithm for a function

`Boolean findSum(float[] A[1…n], float s)`

which, given an array A of distinct numbers (possibly non-integers) and a number s, determines whether there are two distinct numbers in the array whose sum equals to s. If so, the function returns True; otherwise, it returns False.

For example, if A = [17,24,6,18,25,3] and s = 27, then `findSum` should return True because there are two distinct numbers in A, namely, 24 and 3, whose sum is 27. On the other hand, if A = [17,24,6,18,25,3] and s = 12, then `findSum` should return False.

8.1 Provide a pseudo-code description of your algorithm for `findSum` which has the worst-case running time of O(n log n), where n is the size of the given array.

8.2 Give a worst-case time complexity analysis of your algorithm.

```
            Boolean find Sum (A, S)
  n logn    {  A = mergesort(A)
       1    {  L = A[0]
       1    {  R = A[length(A)-1]
       n    {  while (L != R)
                   1   if L + R = S
                   1       return true
                   1   if L + R > S
                   1       R = index.R - 1
                   1   if L + R < S
                   1       L = index.L+1
                   1   return false
```

This is the end of the exam paper.