Data Structure and Algorithms

Written Assignment 1

Problem 1. (15 marks)

What is the total number of basic operations that the following two code pieces consume? Make your answer precise and list your steps of calculation.

For simplicity, you may assume that *n* is always a power of 2 and is at least 2, e.g., 2, 4, 8, etc.

(a)

```
int f1 (int n) {
        int res = 0;
        for(int i = 0; i < n; i++)
                res ++;
        return res;
}
```

(b)

```
int f2 (int n) {
        int res = 0;
        for(int i = 1; i < n; i*=2)
                for(int j = 0; j < n; j++)
                        res += i * j;
        return res;
}
```

(c)

```
int f3(int n) {
        if(n<=0)
                return 0;
        return 2 * f3(n-2) + 1;
}
```

Problem 2. (20 marks)

For each pair of *f(n)* and *g(n)* below, decide if *f(n)* = *O(g(n))*, *f(n)* = *Ω(g(n))*, or *f(n)* = *Θ(g(n))*. Justify your answer using the definition of these asymptotic notation. Note that more than one of these relations may hold for a given pair; list all correct ones.

(a) $\mathrm{f(n)} = n$ and $\mathrm{g(n)} = 2n + \log_2 n$.

(b) $\mathrm{f(n)} = n^2$ and $\mathrm{g(n)} = n^3$.

Problem 3. (20 marks)

Let *f(n)* be an asymptotically positive function. Prove or disprove each of the following conjectures.
Hint: You can prove a conjecture using its definition or disprove a conjecture by giving negative examples.

(a) f(n) = Θ (f(n) + 1).

(b) f(n) = Θ (f(n+1)).

Problem 4. (15 marks)

Solve the following recurrence relation and represent *T(n)* using a formula of *n*.

$$T(n) = \begin{cases} T\left(\frac{n}{2}\right) + n, & n > 1 \\ 1, & n = 1 \end{cases}$$

Problem 5. (15 marks)

In the merge sort algorithm, we divide an array into two halves, recursively sort the subarrays, and then merge them into a sorted array. Now Ming proposes a “merge sort pro” algorithm. In “merge sort pro”, an array is divided evenly into 3 subarrays instead of two, and the rest of the steps are similar to those of merge sort. What do you think is the time cost of “merge sort pro” if the input size is *n*? Prove your answer.

Problem 6. (15 marks)

Let's call an array *A* of size *n* a ***mountain*** if the following properties hold:

- *n* >= 3
- There exists some *i* such that *0 < i < n-1* and that *A[0] < A[1] < ... A[i-1] < A[i] > A[i+1] > ... > A[n-1]*

Further, *A[i]* is called the ***peak*** of the mountain. For example, if *A=[2, 3, 5, 7, 4]*, then its peak is *7*.

Given an array A of size *n* that is definitely a mountain, design an algorithm that returns the value of its peak. Describe your algorithm using pseudo code.

Note: The cost of your program should be *O(log n)*. Don't use brute force.

Hint: Use divide and conquer. The base case is when the subarray size is 3.