United International University

Department of Computer Science and Engineering

Course: CSI 227 Algorithms　　Trimester: Spring 2019

Midterm Exam　　Marks: 90　　Time: 1 hour 45 minutes

There are FOUR questions. Answer ALL questions.

1. a) Analyze both *best* and *worst* case time-complexities of the algorithm of Figure 1. **7 + 7**

```
n = length[A];   m = length[B];   i = 1;   sum = 0;
while (i <= n) do {
      print A[ i ];
      for ( j ← n; j >= 1; j = j / 3) do {
            if (A[ i ] + A[ j ] < 100) then
                  break;
            for ( k ← 1; k <= n; k = k + 2) do
                  print A[ k ];
      }
      i = i + 1;
}
for ( i ← 1; i < m; i ++) do {
      if (B[ i ] > 50) then
            return –1;
      else
            sum += B[ i ] * B[ i ];
}
```

Figure 1: Algorithm for Q. 1(a) and Q. 1(b)

b) Show both *best* and *worst* case examples of the arrays A and B with $n = 6$ and $m = 7$ for the algorithm of Figure 1. **3 + 3**

c) Prove that $8n^3 + 7n^2 + 5 = O(n^3)$, but $8n^3 + 7n^2 + 5 \neq O(n)$. **5 + 5**

2. a) Propose a divide-and-conquer algorithm to find the *number of even numbers* in an array of n integers. **6 + 4**

If the recurrence equation for the time-complexity of the algorithm is

$T(n) = 2T(n/2) + O(1)$　　with $T(1) = O(1)$

then using the recursion tree method, determine a good asymptotic upper bound of the algorithm.

b) Consider a *modified version* of the Merge sort algorithm as follows: if the array size is less than or equal to 2, then it sorts the array at constant time. Otherwise, it divides the array of size n into 3 subarrays, each with a size of $n/4$. This division takes $O(\log n)$ time. Then the algorithm sorts the subarrays recursively, and then merges their solutions at time $O(n)$. Write a recurrence relation for the running-time $T(n)$ of this algorithm. **3**

c) Using the substitution method, find upper bound on the following recurrence. **7**

$T(n) = 2T(n/4) + O(n)$　　with $T(1) = O(1)$

3. a) Write the principles of *Greedy* method and *Divide-and-Conquer* method for solving a problem. **2 + 3**

b) Provide an example where the greedy strategy fails for the 0/1 knapsack problem. **4**
Explain why the greedy strategy fails for the 0/1 knapsack problem, but works for the fractional knapsack problem.

c) Provide separate examples with exactly 4 activities where the greedy algorithm outputs sub-optimal solution if a greedy choice is made by: **3 + 3**

i) earliest start time, and

ii) shortest interval.

d) Suppose that you have got a set of the following activities, with the given start and end times for the Activity Selection problem: **5**

{ [1, 6), [2, 5), [9, 15), [6, 9), [11, 15), [3, 6) }

Find *two optimal solutions* of the problem.

4. a) i) Write a Dynamic Programming algorithm for the classical *Coin Change problem.* **5 + 5**

ii) *Happy coins* are used by the people of the *Happyland.* Assume that only the following coins are available at this land: \$1, \$7, \$11, and \$15.

By using the Dynamic Programming method, find the minimum number of coins that add up to a given amount of money of \$20.

b) Solve the following instance of the knapsack problem with knapsack capacity 7 for **7 + 3**

i) 0/1 knapsack problem

ii) fractional knapsack problem

Item	**Weight**	**Value**
1	3	150
2	3	180
3	2	170
4	3	120
5	3	210