This part for lecturer's grading use only	CS3230

NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

SEMESTER 2 (2012/2013)

EXAMINATION FOR

CS3230: Design and Analysis of Algorithms

May 2013 **Time Allowed: 2 Hours**

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **NINE (9)** problems and comprises **NINE (9)** printed pages, including this page.
2. Answer **ALL** questions within the space provided in this booklet.
3. This is an **Open Book** examination.
4. All your asymptotic bounds must be in its simplest form. Namely, if the correct answer is $O(n^2)$ and you answered $O(n^2+n)$, you will be penalized.

MATRICULATION NO:

This portion is for examiner's use only

Problem 1 and 2	
Problem 3	
Problem 4	
Problem 5	
Problem 6	
Problem 7	
Problem 8	
Problem 9	
Total	

Problem 2 (6 points)
Consider the recurrence relation $T(n) = 2T(\text{floor}(n/4)) + n^2$, for $n > 3$. Here the function floor(x) returns the floor of x. We also know that T(1)=T(2)=T(3)=1. Use mathematical induction to prove that $T(n) = O(n^2)$. (You must use mathematical induction, otherwise you will not get any points. In particular, you cannot invoke the master recurrence theorem.)

Problem 3. (6 marks)
We would like to apply prefix coding on a certain document to compress its size. The frequency counts of the characters in the document are:

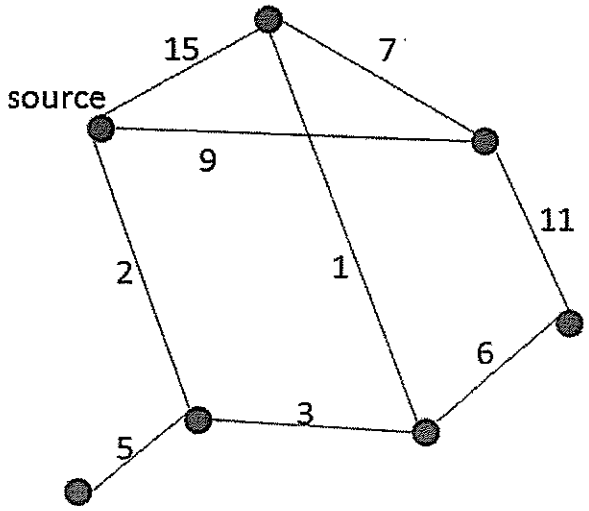
A: 56 B: 12 C: 3 D: 4 E: 6 F: 20 G: 1

Draw the optimal Huffman tree for compressing this document:

What is the minimum number of bits needed to represent this document using prefix codes? Write your answer below, no need to provide steps of derivation.

Your answer is _____.

Problem 4. (6 marks)
Imagine that we run Dijkstra's single-source shortest path algorithm on the following undirected graph where the weight of each edge is indicated next to that edge.



We call an edge as *useful* if it is part of the shortest path (as computed by the algorithm) from the source to some node. Otherwise the edge is called *useless*. Indicate which 3 edges in the graph are useless. You can indicate the 3 edges by indicating their weights, since the edges in the graph all have distinct weights. Write your answer below:

The weights of the 3 useless edges are _____, _____, and _____, respectively.

Problem 5. (4 marks)
Recall that Prim's algorithm finds a spanning tree with the minimum cost in a graph G. Explain clearly how to modify Prim's algorithm so that the algorithm can now find a spanning tree with the **maximum** cost in G. Your modification should not increase the worst-case **asymptotic** time complexity of Prim's algorithm.

Write your suggested modification to Prim's algorithm below:

Explain clearly why using your modification will find the spanning tree with the maximum cost **and** why the complexity satisfies our requirement:

Problem 6. (6 marks)
Consider the following variant of the knapsack problem. There are n different volunteer jobs, where the ith volunteer job takes exactly $v[i]$ hours for i from 1 to n. A volunteer job may be done more than once (i.e., repetition is allowed). But it is not allowed to do only part of a volunteer job – in other words, a job that has started must be completed. Alice has total S hours to do the volunteer jobs, and she would like to **fully spend** the S hours so that she does not waste her life. All the $v[i]$'s and S are integers. Design a deterministic algorithm using dynamic programming to determine whether it is possible for Alice to achieve her goal. You algorithm should have $O(nS)$ worst-case time complexity.

First, write out the recurrence relation for your dynamic programming, with $F[x]$ denoting whether Alice can achieve her goal if Alice has exactly total x hours:

Next, write the pseudo-code for your algorithm:

Finally, briefly explain why your algorithm is correct **and** why its time complexity is $O(nS)$:

Problem 7. (6 marks)
Consider the following game between you and Alice. Alice holds a positive integer n, where n is no larger than some other integer M. You do not know n. You do not know M either. You are allowed to choose an arbitrary value x and invoke query(x), and Alice will tell you whether n is larger than x, smaller than x, or equal to x. Alice does not lie. Each invocation of query() takes $\Theta(1)$ time. Design a deterministic algorithm to find out the value of n, such that the worst case time complexity is a function of n. In other words, **the time complexity should not depend on M.** Also, you are required to incur as small asymptotic worst-case time complexity as you can in your solution. **Algorithm efficiency will be a factor considered during grading.** (Hint: Doing a binary search on over $[0, M]$ will not work, for two reasons. First, you do not know M. Second, the time complexity will depend on M and does not satisfy the requirement.) Your answer must have 3 parts: a concise explanation of your high-level idea, your pseudo-code, and a concise explanation on correctness and time complexity.
Write your high-level idea below:

Write your pseudo-code below (you are allowed to invoke algorithms that we have learned in the module without writing out the pseudo-code of those algorithms):

Write your explanation on the correctness **and** the time complexity below:

Problem 8. (4 marks)
There are two decision problems A and B. We already know that problem A has a lower bound of $2n^2$ on its worst-case time complexity. We also have worked out a reduction algorithm that reduces A to B. For any given input X (of size n) to problem A, the reduction algorithm will change it to another input Y (of the same size n) to problem B. The answer to problem B is then directly output as the answer to problem A. The worst-case time complexity of the reduction algorithm itself (not including the complexity incurred for invoking the algorithm for B) is exactly n^2. Based on these conditions, is there any conclusion we can make regarding the lower bound on the worst-case time complexity for solving B? If yes, what is the strongest conclusion that we can make? If not, why? Rigorously justify and prove your answer.

Problem 9. (6 marks)
Prove rigorously that the following decision problem is NP-hard: Given an undirected graph G and an integer k, determine whether G simultaneously contains a clique of size at least k **and** an independent set of size at least k.

END-OF-PAPER