The exam will be closed book, closed notes, except for 1 page of notes.

- No electronic equipment allowed (cell phones, PDA’s, computers...).
- Write legibly.
- Use pseudo-code or English to describe your algorithms.
- When designing an algorithm, you are allowed to use any algorithm or data structure we have covered in cs141 or in cs14 without giving its details, unless the question specifically asks for such details.
1. +3 points for each correct answer, -2 points for each incorrect answer, 0 points for each question not answered.

The running time of the following algorithm is $O(2^n)$:  

1. void recurse(unsigned int n) {
2. if (n <= 0) return;
3. return recurse(n-1) + recurse(n-1) + recurse(n-1);
4. }

The running time of the following algorithm is $O(n \log n)$:  

1. void loop(unsigned int n) {
2. for (int i = 0; i < n; ++i)
3. for (int j = 0; j < i*log(i); ++i)
4. print "Hi\n";
4. }

The running time of the following algorithm is $O(n)$:  

1. Array<int> F(-1);
2.
3. int fib(unsigned int n) {
4. if (n <= 1) return n;
5. if (F[n] == -1) F[n] = fib(n-1) + fib(n-2);
6. return F[n];
7. }

The largest number printed by mystery(n, 0) is $\Theta(\log n)$:  

1. void mystery(int n, int d) {
2. std::cout << d << std::endl;
3. if (n <= 1) return;
4. mystery(n/2, d+1);
5. }

Suppose $T(0) = 1$ and $T(n) = T(n-1) + T(n-3)$ for $n > 0$. The recursion tree for $T$ has $O(2^n)$ nodes.
Suppose $T(0) = 1$ and $T(n) = T(n - 1) + T(n - 3)$ for $n > 0$. The recursion tree for $T$ has $Ω(2^{n/3})$ nodes. □ True □ False

Suppose $T(0) = 1$ and $T(n) = 2 * T(n/2) + n$ for $n > 0$. Then $T(n) = Θ(n \log n)$. □ True □ False

There exists a graph with 4 vertices such that every acyclic set of edges in the graph is a spanning tree. □ True □ False

Recall that an $s, t$-cut vertex is a vertex other than $s$ or $t$ whose removal separates $s$ from $t$. Every $s, t$-cut vertex in a graph is also a cut vertex. □ True □ False

Dijkstra’s algorithm runs in linear time. □ True □ False

If the hash function is bad, accessing a key in a hash table can take $Ω(n)$ time, where $n$ is the number of items in the table. □ True □ False
Run depth-first search on the above graph. Start at A, and whenever you have a choice about which edge to explore next, choose the one that goes to the vertex that is earliest in the alphabet. Draw the DFS tree below, drawing the tree edges with solid lines and the non-tree edges using dashed lines. In your drawing, label each vertex with its dfs post-order-number.

Below, give the corresponding topological ordering (the one that would be produced by the topological sorting algorithm based on DFS).
3. Describe the linear-time algorithm for finding shortest paths in an acyclic directed graph with edge weights. Illustrate the algorithm on an example. State the high-level idea of the algorithm, and explain why it takes only linear time. To be sure your answer is precise and detailed enough, you may want to give pseudo-code (if you have time).
4. Describe a linear-time algorithm for finding maximum-bottleneck paths in an acyclic directed graph with edge weights.

Illustrate your algorithm on an example.

State the high-level idea of the algorithm, and explain why it takes only linear time. To be sure your answer is precise and detailed enough, you may want to give pseudo-code (if you have time).
5. Describe a linear-time algorithm for finding a shortest-path tree from a given vertex $s$ in any directed graph where every edge has weight 0 or 1.

Illustrate your algorithm on an example.

State the high-level idea of the algorithm, and explain why it takes only linear time. To be sure your answer is precise and detailed enough, you may want to give pseudo-code (if you have time).
6. Given an array $A[1..n]$ of integers (possibly negative), define partial sum

$$S(i, j) = \sum_{k=i}^{j} A[k] = A[i] + A[i + 1] + \cdots + A[j].$$

We want a fast algorithm to compute the largest partial sum:

$$\max_{i, j} \{S(i, j) : 1 \leq i \leq j \leq n\}.$$ 

Define $M[j] = \max_i \{S(i, j) : 1 \leq i \leq j\}$. (That is, $M[j]$ is the maximum partial sum whose last term is $A[j]$.)

Prove the following recurrence: $M[j] = \max\{M[j - 1] + A[j], A[j]\}$. 

– continued on next page –
Describe a linear-time algorithm, based on the recurrence, to compute $\max_{i,j}\{S(i,j) : 1 \leq i \leq j \leq n\}$. 