CS 141, spring 2004 practice midterm

Name (first last):______________________________

Student ID:______________________________

- This is exam is open book, open notes.
- No electronic equipment allowed (cell phones, PDA's, computers...).
- Write legibly. What can't be read will not be graded.
- 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.
- *You may open the exam and start working when the bell rings at the start of class. You must stop working on your exam and turn it in when the bell rings at the end of class.*
- If you have a question about the meaning of a question, raise your hand.

1.	/30	2.	/10		
3.	/10	4.	/10		
5.	/10	6.	/10		
7.	/10	8.	/10	total:	/100

1. *+3 points for each correct answer, -3 points for each incorrect answer, 0 points for each question not answered.*

$\sum_{i=1}^{n} \log(i)^2 = \Omega(n\log(n))$ ________________ ☐ True ☐ False

$\sum_{i=1}^{n} \log(i)^2 = O(n\log(n)^3)$ ________________ ☐ True ☐ False

$\sum_{i=1}^{n} i^3 = \Theta(n^4)$ ________________ ☐ True ☐ False

If $\alpha = 1$, then $\alpha^0 + \alpha^1 + \alpha^2 + \cdots + \alpha^n = \Omega(1)$ ________ ☐ True ☐ False

Suppose $T(0) = 1$ and $T(n) = n + T(n-1)$ for $n > 0$.
Then $T(n) = \Omega(n^2)$. ________________ ☐ True ☐ False

Suppose $T(0) = 1$ and $T(n) = n^2 + T(n/2)$ for $n > 0$.
Then $T(n) = \Theta(n^2 \log n)$. ________________ ☐ True ☐ False

Suppose $T(0) = 0$ and $T(n) = n + 2T(n/2)$ for $n > 0$.
Then $T(n) = O(n)$. ________________ ☐ True ☐ False

Suppose $T(0) = 1$ and $T(n) = 1 + 3T(n/4)$ for $n > 0$.
The recursion tree for T has depth $\Theta(\log n)$. ________ ☐ True ☐ False

Suppose $T(0) = 1$ and $T(n) = 1 + 2T(n-2)$ for $n > 0$.
The recursion tree for T has $O(2^n)$ nodes. ________ ☐ True ☐ False

Let T be any tree. Let n_1 be the number of degree-1 vertices in T let n_2 be the number of vertices of degree 2 or larger. If every vertex in the tree has degree 4 or less, then $n_1 = O(n_2)$. ________________ ☐ True ☐ False

2. What is the running time of the following subroutine, as a function of n?

```
int mystery4(int n) {
  int x = 0;
  for (int i = 0;  i < n*n*n;  ++i)
      for (int j = i;  j < n;  ++j)
         ++x;
  return x;
}
```

Give pseudo-code for an algorithm that runs in linear time and computes the same value as mystery4(n).

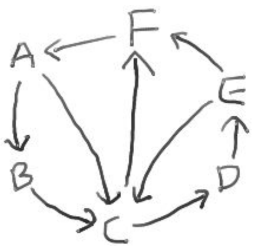
3. Let T(n) be the time taken by the following subroutine:

```
int mystery5(int n) {
  if (n <= 1) return 2;
  else return (2*mystery3(n/2) + 3*mystery3(n/3)) % 10;
}
```

Give a recurrence relation for $T(n)$ (ignoring constant factors).

What is the depth of the deepest node in the recursion tree (in terms of n)?

4.



Run depth-first search on the above directed 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.

Below, draw the DFS tree, using arrows to show the orientation of each edge. Use solid lines for the tree edges, use dashed lines for the non-tree edges.

Label each vertex with its dfs-number (the order in which the dfs encounters the vertex)

Below, under each vertex give its post-order number (as defined in the algorithm for topologically sorting a DAG).

A	B	C	D	E	F

5. Suppose we modified the growable array class so that instead of doubling the table in grow(), we increased the size of the table by a factor of $c = 1.01$.

   Would the modified data structure still take worst-case total time $O(N + M)$ to support a sequence of N accesses (where M is the final size of the table)?

   Why or why not?

   Suppose we modified the growable array class so that instead of doubling the table in grow(), we increased the size of the table by 1000 elements.

   Would the modified data structure still take worst-case total time $O(N + M)$ to support any sequence of N accesses (where M is the final size of the table)?

   Why or why not?

6. Recall the HashTable data structure from programming assignment 1. Explain, in words or pseudocode, what has to be done in the grow() routine. First give a high-level two-sentence description, then describe more specifically the steps and data structures you would use to implement the routine.

What is the worst-case running time of grow(), in terms of the initial table size M and the number N of keys stored in the table, using your implementation?

7. Prove or disprove the following two claims:

   For any digraph G and any edge (u, w) on a cycle in G:

   (a) It is always possible to run DFS on G in such a way that (u, w) is classified as a tree edge.

   (b) It is *never* possible to run DFS on G in such a way that (u, w) is classified as a cross edge.

8. Give a high-level but precise description of an efficient algorithm for the following problem:

   Given a directed acyclic graph G and three vertices S, W, T, compute the number of paths from S to T that go through W.

   A linear-time algorithm is possible. What is the worst-case running time of your algorithm in terms of the number of edges and vertices of G?

   Argue that your algorithm is correct: