READ THIS NOW!

- Print your name in the space provided below.
- Unless a question involves determining whether given Java code is syntactically correct, assume that it is, and that any necessary `import`s have been made.
- There are 7 short-answer questions, priced as marked. The maximum score is 100.
- When you have finished, sign the pledge at the bottom of this page and turn in the test and your fact sheet.
- Aside from the allowed one-page fact sheet, this is a closed-book, closed-notes examination.
- No laptops, calculators, cell phones or other electronic devices may be used during this examination.
- You may not discuss this examination with any student who has not taken it.
- Failure to adhere to any of these restrictions is an Honor Code violation.

Name (Last, First) _____

`printed`

Pledge: On my honor, I have neither given nor received unauthorized aid on this examination.

_____

signed

1. Assume that $f(n)$, $g(n)$ and $h(n)$ are positive-valued functions defined on the set of non-negative integers.

a) [6 points] What does the fact given below imply regarding any big-O, big-Ω and/or big-Θ relationships between the functions? Be complete. No justifications are needed.

$\forall n \le 1000,\ g(n) \le 7h(n)$

NO conclusions can be reached. The bound on *n* is pointing in the wrong direction.

b) [6 points] What does the fact given below imply regarding any big-O, big-Ω and/or big-Θ relationships between the functions? Be complete. No justifications are needed.

$\forall n \ge 42,\ 3g(n) \le f(n) \le 5h(n)$

By definition, we can conclude that f is $\Omega(g)$ **and that** f is $O(h)$.

But, of course that also implies that g is $O(f)$ **and that** h is $\Omega(f)$.

And, transitivity yields that h is $\Omega(g)$ **and** g is $O(h)$.

---

2. [15 points] For each part, state the simplest function $g(n)$ such that the given function is $\Theta(g)$; you should use theorems from the course notes; no justification is required

a) $a(n) = 14n^2 + 3n\log n$ n^2

b) $b(n) = 3n^2 \log n + 5000n$ $n^2 \log n$

c) $c(n) = 10n^{17} + 2^n$ 2^n

d) $d(n) = 4n(\log n + 1)$ $n \log n$

e) $e(n) = 2n^7 + 5n^3 + 7n^2 + 2n + 42$ n^7

3. [15 points] The following algorithm is the most obvious, but inefficient, way to evaluate a polynomial $f(x) = \sum_{i=o}^{N} a_i x^i$.

Assume that the coefficients of the polynomial are stored in an array `a[]` and the value for x is stored in a variable `x`.

```
Poly = 0;                          // 1

for (i = 0; i <= N; i++) {         // 1 before, 2 per pass, 1 to exit

   xtoN = 1;                       // 1

   for (j = 1; j <= i; j++)        // 1 before, 2 per pass, 1 to exit
      xtoN = x * xtoN;             // 2

   Poly = Poly + xtoN * a[i];      // 4
}
```

Derive a simplified complexity function T(N) for the given algorithm. Show all supporting work.

$$T(N) = 1+1+\sum_{i=0}^{N}\left(2+1+1+\sum_{j=1}^{i}(2+2)+1+4\right)+1$$

$$= \sum_{i=0}^{N}\left(\sum_{j=1}^{i}4+9\right)+3$$

$$= \sum_{i=0}^{N}(4i+9)+3$$

$$= 4\frac{N(N+1)}{2}+9(N+1)+3$$

$$= 2N^2+11N+12$$

5. [14 points] Suppose that an application performs a large number of searches for data records that are stored in a file on disk. The application uses a buffer pool to cache records that are retrieved from disk as searches are carried out.

Suppose that the buffer pool has 25 slots, and that the most recent search caused a record **R** to be fetched from disk and stored in the buffer pool.

a) [3 points] If the buffer pool uses the FIFO replacement policy, what is the maximum number of records that can be fetched from disk before **R** will be replaced?

When R is fetched, it is placed at the end of the FIFO list; each new record that is fetched pushes R one spot closer to the other end of the list, so it takes 24 fetches to push R to the very end and one more fetch to push R out of the list.

So, 24 new records can be fetched before R will be replaced.

b) [3 points] If the buffer pool uses the FIFO replacement policy, what is the maximum number searches that can be performed before **R** will be replaced?

R is only pushed towards the other end of the list when a search requires loading a new record from disk. So, in principle, there could be an infinite number of searches without requiring that R even move toward the other end, much less be replaced.

c) [8 points] Suppose that **R** is a popular record, and is the target of roughly 5% of the searches that are performed. Discuss precisely how **R** would be treated if LRU replacement were used instead of FIFO replacement.

If R is hit roughly 5% of the time, then R would probably never be replaced once it has entered the list.

Each time R takes a hit, including the search that initially loads R into the LRU list, R will be placed at the top of the list.

Each time some other record takes a hit, whether that record is already in the list or not, R will move one spot toward the bottom of the LRU list.

Now, if R is hit roughly 5% of the time, R will be hit about 1 in every 20 searches, so we would not expect that R will ever reach the bottom of the LRU list and be pushed out before R takes another hit and moves back to the top of the list.

6. Suppose that an application performs a large number of searches for data records that are all stored in some structure in memory. There are **N** different data records.

a) [7 points] Describe a way to organize the records in memory so that the worst-case search cost is guaranteed to be $\Theta(\log N)$.

Placing them in sorted order in an array will permit us to use binary search and that guarantees that the worst-case search cost is $\Theta(\log N)$.

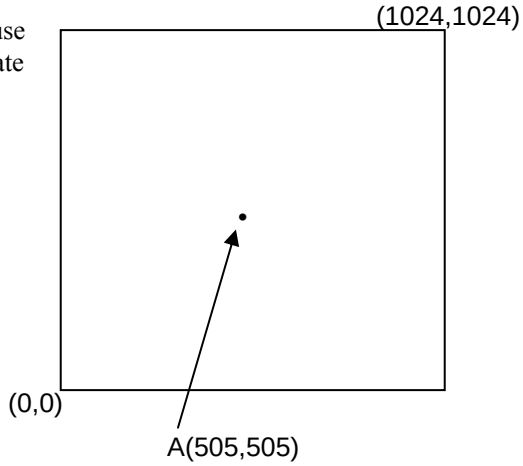
A BST will NOT automatically guarantee that since the tree's structure depends on the order in which the elements are inserted. However, if we sorted the elements and then shuffled them into the tree in the appropriate order, we could build a minimum-height BST and that would also guarantee $\Theta(\log N)$ worst-case search cost.

b) [7 points] If the data records are stored in a binary search tree, why would the worst-case search cost not simply be a function of **N**?

The structure of the tree depends on the order in which the data values are inserted. With an optimal insertion order, the worst-case search cost would be $\Theta(\log N)$ but with a bad order the worst-case search cost would be $\Theta(N)$.

7. Consider using a PR quadtree, with bucket size 1, to organize data points that lie in the region shown below.

a) [8 points] Is it possible that inserting a second data point will cause more than one splitting to occur? If yes, explain how and illustrate with a specific example. If no, explain why not.



Suppose we inserted B(504, 504).

That is close to, and SW of A.

Simply dividing the world square once will not achieve a separation of A and B.

b) [8 points] Is it necessarily the case that inserting a second data point that is a distance of 10 from A will cause more than one splitting to occur? If no, explain why not and illustrate with a specific example. If yes, explain why.

No. Consider inserting B(515, 505). That's exactly 10 units east of A, and if we simply divided the given world square once we'd have A in the SW quadrant and B in the SE quadrant.