READ THIS NOW!

- Print your name in the space provided below.
- There are 8 short-answer questions, priced as marked. The maximum score is 100.
- This examination is closed book and closed notes, aside from the permitted one-page formula sheet. No calculators or other computing devices may be used. The use of any such device will be interpreted as an indication that you are finished with the test and your test form will be collected immediately.
- Until solutions are posted, 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.
- When you have finished, sign the pledge at the bottom of this page and turn in the test and your fact sheet.

Name (Last, First) Solution
printed

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

signed

1. [12 points] Determine the exact count complexity function T(N) of the following algorithm. Your answer for T(N) should be in simplest form. Show supporting work!

```
for (i = 1; i <= N; i++) {      // Line  1:  1 before, 2 per pass, 1 to exit
   x = 0;                       // Line  2:  1
   y = N;                       // Line  3:  1
   for (j = 1; j <= 2*i; j++) { // Line  4:  1 before, 3 per pass, 2 to exit
      if ( i % j < 2 ) {        // Line  5:  2
         x = y + j;             // Line  6:  2, if done
         y--;                   // Line  7:  1, if done
      }
      else {
         x = 2*y – j;           // Line  8:  3, if done
         y++;                   // Line  9:  1, if done
      }
   }
}
```

$$T(N) = 1 + \sum_{i=1}^{N}\left(1+1+1+1+1+\sum_{j=1}^{2i}\left(2+1+2+\max(3,4)\right)+2\right)+1$$

$$= \sum_{i=1}^{N}\left(\sum_{j=1}^{2i}(9)+7\right)+2$$

$$= \sum_{i=1}^{N}\left(9\cdot 2i+7\right)+2$$

$$= 18\frac{N(N+1)}{2}+7N+2$$

$$= 9N^2+16N+2$$

2. [12 points] Assume that $f(n)$ and $g(n)$ are positive-valued functions defined on the set of non-negative integers.

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

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

Because the statement only holds for values of *n* that are less than 1000, this implies NO conclusions about O or Ω or Θ relationships.

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

$$\forall n \ge 1,\ 7g(n) \le f(n)$$

Straight from the definitions, this implies that f is Ω(g), and if we divide through by 7 we see that it also implies that g is O(f).

3. [12 points] Suppose that you need to implement a resizing method for a hash table, which will create a larger array and install all of the records that are in the current array into the new array. It was stated in class that you would have to rehash the keys of all the existing records in order to place them in the new array.

It is suggested that since all the records that fall into the same "chain" will have keys that hash to the same integer, it is only necessary to rehash one key per "chain" in order to put the records into the new array.

Is this argument correct, or not? Explain why or why not.

No.

The fact that two records wind up in the same home slot (remember, no probing when chaining is used), simply means that when their keys are hashed and modded by the table size, the result is the same.

It does not mean that the keys are mapped to the same integer by the hash function.

And, since modding by a different table size can yield a different remainder, we must know exactly what integer each key hashes to.

4. [12 points] Suppose that four keys, Key1, Key2, Key3 and Key4, all collide in the same home slot when inserted into a hash table. The hash table uses a hash function, H(), to determine the home slot, and uses probing to resolve the collisions.

The designer of the hash table is considering two possible probing schemes:

i) quadratic probing, using H(key) + k^2 to pick the slot for the k-th probe step

ii) double hashing, using H(key) + k*H2(key) to pick the slot for the k-th probe step, where H2() is a second hash function that is different from H()

Is there any reason to expect that one of these schemes might produce better results than the other? Explain why or why not.

If quadratic probing is used, then exactly the same sequence of probe slots will be used for all four records, and therefore a search for the last one inserted must require a minimum of 3 probe steps (not counting the home slot).

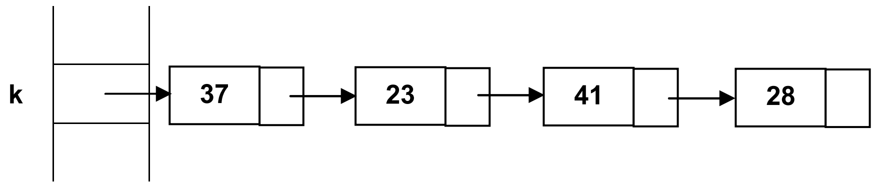
If double hashing is used, then there is at least a chance that the four keys will not all be hashed to the same integer value (using H2), and therefore that they will not all generate the same sequence of probe slots.

In fact, it is possible that each key will hash to a different integer value and generate an entirely different set of probe slots. So, using double hashing could result in searches for each of the four records (except the first), requiring only one probe step.

Of course, this isn't guaranteed, but it is not even possible if quadratic hashing is used.

5. [12 points] Two developers are told to implement a hashing scheme for the same set of data records, and to use the same hash function and table size. One developer decides to use linear probing to resolve collisions. The other developer decides to use chaining with singly-linked lists in each table slot, appending new records to the lists.

When testing her implementation, the second developer discovers that one of the table slots ends up in the following state (displaying key values for the records):



Given that information, should the first developer be confident that if the record with key 28 is searched for in his implementation, exactly 3 probe steps will be required (accessing the home slot doesn't count as a probe step)? Explain clearly why or why not. If not, could the number of required probe steps be less than x? more than x?

No. The problem with linear probing is that records that have different home slots can share probe slots, and hence collisions can result in clusters of filled slots that lead to increased probing costs.

For example, the probe sequence from the home slot to the slot holding 23 could be separated from the slot holding 37 by another record whose home slot was immediately after the slot holding 37.

6. [12 points] Recall Pugh's probabilistic skip list. Assume that an implementation does use a random number generator that has the necessary property of producing a uniform distribution of values, and in fact when 1023 data values are inserted to the skip list, we get 512 nodes in level 0, 256 in level 1, 128 in level 2, 64 and level 3, 32 in level 4, 16 nodes in level 5, 8 nodes in level 6, 4 nodes in level 7, 2 nodes in level 8, and 1 node in level 9.

In other words, we get a "perfect" distribution of nodes to levels.

Explain how it is still possible that many, if not most of the searches in the skip list will require $\Theta(N)$ comparisons. Be specific – describe the circumstances precisely – a diagram would probably be useful.

The key to the probabilistic skiplist's performance is that although the node heights are selected randomly, it is unlikely that they will randomly fall into a pattern that defeats the advantage of having all of those forward pointers.

Suppose that the level 9 node is first, followed by both the level 8 nodes, then all the level 7 nodes, and so forth, with the 512 level 0 nodes coming last.

Then the forward pointers do no good. Each node only has pointers to the node that immediately follows it, and every traversal of the list will be node-by-node just as if we were using a simple singly-linked list.

That's the worst case. It's still possible to get poor performance if we just create enough runs of consecutive nodes all in the same level; some searches might be efficient, but any search that falls into one of those runs will degrade to a linear search until it reaches the end of the run or finds its target within the run.

7. [12 points] Recall the AVL tree. Determine whether each of the following statements is true or false, and give a very brief explanation.

a) If an AVL tree and a BST contain the same values, then searching for the same value may require fewer steps in the AVL than in the BST.

True.

Search in the AVL is expected to be require fewer steps than the BST, on average, since the AVL will frequently have many fewer levels than the BST.

b) If an AVL tree and a BST contain the same values, then there may be more nodes in the AVL than in the BST.

False.

There is a one-to-one match between data values and nodes in every BST and AVL. They will have the same number of nodes if they contain the same number of values.

c) If an AVL tree and a BST <u>have the same structure</u> and contain the same values, then insertion of the same new value may require fewer steps in the AVL tree than in the BST.

False.

The insertion process is identical in the AVL and BST up to the point that the new leaf node (holding the new data value) is added to the tree.

At that point, the BST insertion is complete, but the AVL may perform restructuring operations after the insertion has been completed.

8. [16 points] Consider using a PR quadtree (with bucket size 1) to organize data objects that have integer coordinates in the square region of the xy-plane that is bounded between the points (0, 0) and (1024, 1024).

a) If there are 20 data points, what is the smallest number of levels the quadtree could have? Justify your conclusion.

Since data values are stored only in the leaves, we must have enough levels to support at least 20 leaves.

The possible number of leaves increases by a factor of 4 as we go from level to level:

1 4 16 64 ...

So we need at least 4 levels.

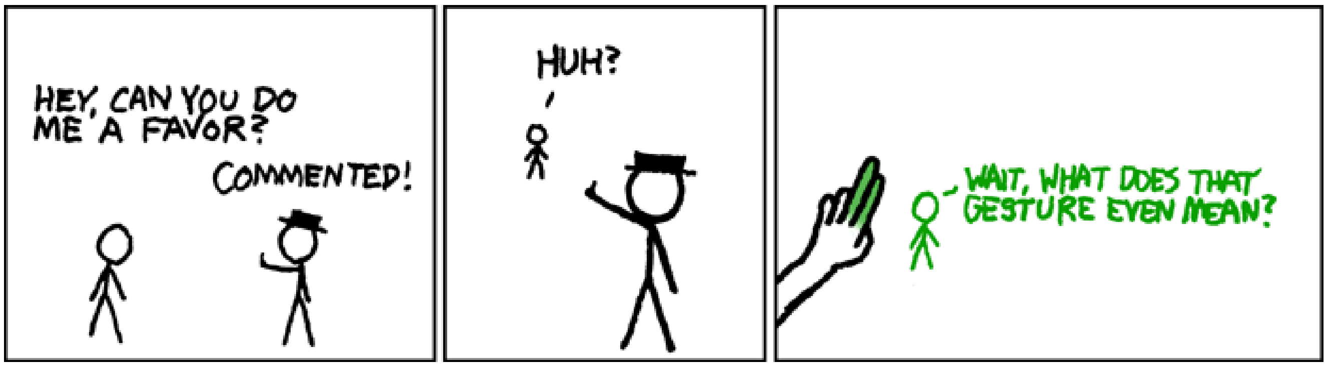
b) Suppose the quadtree contains 50 data points, and that every leaf node is the child of an internal node represents a region that is 64 x 64.

Suppose that another data object is inserted to the quadtree, and that the insertion of the new data object causes the splitting of a former leaf region. What is the maximum distance the new data object can be from the data object that was already in that leaf region (before the insertion)? Explain your reasoning clearly.

Each leaf node represents a 32 x 32 region; in order to force a split, the new data object must fall into the same leaf region as an existing data object.

The furthest apart two data objects can be within a 32 x 32 square (if they are at opposite corners) would be

$$\sqrt{32^2 + 32^2} = \sqrt{2 \cdot 32^2} = 32\sqrt{2}$$


HEY, CAN YOU DO ME A FAVOR?
COMMENTED!
HUH?
WAIT, WHAT DOES THAT GESTURE EVEN MEAN?