READ THIS NOW!

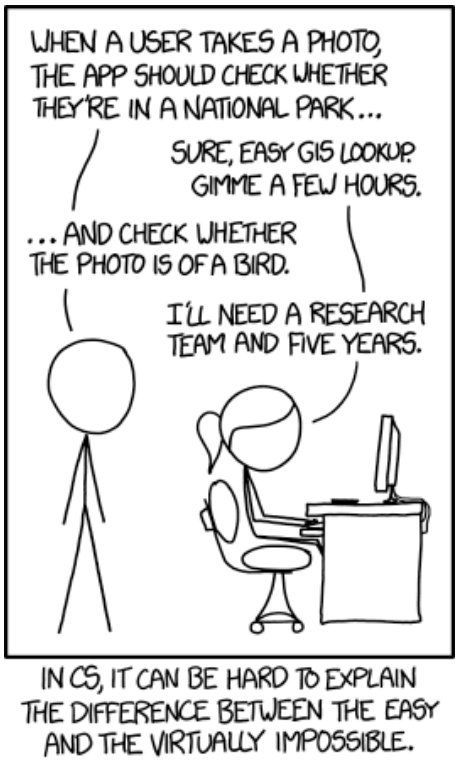
- Print your name in the space provided below.
- There are 4 short-answer questions, priced as marked. The maximum score is 50.
- Most questions require analyzing a scenario, drawing a conclusion, and justifying that conclusion. Credit will be awarded for making statements that are both true and relevant to the question. Do not feel obligated to fill all the available space when answering the questions.
- This examination is closed book and closed notes.
- No calculators, cell phones, 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, or gone over in class, 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.

Name (Last, First) Solution

printed

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

signed



In the 60s, Marvin Minsky assigned a couple of undergrads to spend the summer programming a computer to use a camera to identify objects in a scene.

He figured they'd have the problem solved by the end of the summer.

Half a century later, we're still working on it

xkcd.com

1. [12 points] Using the rules from the notes, derive and simplify an exact-count complexity function for the following algorithm:

```
metric = 0;                          // 1
for (pos = 1; pos < N; pos++) {      // 1 before, 2 per pass, 1 to exit loop
   curr = values[pos];               // 2
   metric = metric * pos + curr;     // 3
   if ( curr % 2 == 1 ) {            // 2
      metric = metric + curr;        // 2
   }
   metric = metric / pos;            // 2
}
```

$$T(N) = 1 + 1 + \sum_{pos=1}^{N-1} \left(2 + 2 + 3 + 2 + 2 + 2\right) + 1$$

$$= \sum_{pos=1}^{N-1} \left(13\right) + 3$$

$$= 13\left(N - 1\right) + 3$$

$$= 13N - 10$$

Common errors:

- **upper bound on summation as N rather than N-1**
- **miscounting cost of a loop pass (13); most errors were off by 1**

2. [12 points] The complexity function for an algorithm is: $T(N) = 5N\log^2 N + 3N^2$

What is the simplest function $f(N)$ such that the algorithm is $\Theta(f(N))$? Prove your answer is correct. You may use any of the theorems covered in the notes.

T(N) is $\Theta(f(N))$. Unfortunately, this doesn't follow from any Theorem except the limit theorem:

$$\lim_{N\to\infty}\frac{5N\log^2 N+3N^2}{N^2}=\lim_{N\to\infty}\left(\frac{5\log^2 N}{N}+3\right)=\lim_{N\to\infty}\left(\frac{10\log N/N\ln 2}{1}\right)+3$$
$$=\lim_{N\to\infty}\left(\frac{10/N\ln^2 2}{\ln 2}\right)+3=3$$

Common issues:

- **working limit out incorrectly**
- **not showing details of l'Hopital's Rule applications in working out limit**
- **setting up wrong limit and then working it out incorrectly**

3. [10 points] The complexity function for an algorithm is $\Theta(N\log N)$. Given an input of size $N = 2^{20}$, the running time on a certain computer is about 30 seconds. Haskell Hoo IV offers the opinion that if we doubled the size of the input, the same algorithm, on the same computer, would take about 120 seconds to execute. Is this reasonable? Explain.

If the complexity is $\Theta(N\log N)$, then doubling the size of the input is expected to somewhat more than double the number of operations. More precisely, the number of operations would be expected to be about $2N\log 2N = 2N(1+\log N) = 2N + 2N\log N$.

Or, using the specific input sizes, the ratio of the number of operations is reasonably estimated by $T(2*2^{20}) / T(2^{20})$. This yields:

$$\frac{T(2*2^{20})}{T(2^{20})}=\frac{T(2^{21})}{T(2^{20})}=\frac{2^{21}\log(2^{21})}{2^{20}\log(2^{20})}=\frac{21\cdot 2^{21}}{20\cdot 2^{20}}=2\cdot 1.05=2.10$$

So, we'd expect the running time to be a bit more than twice the original time, or a bit more than 60 seconds. Haskell is unreasonable.

The most common issue was to not show calculations the precisely supported a conclusion.

4. [16 points] An algorithm is, in the average case, $\Theta(N^2)$. For each part, circle a choice to indicate whether the given property definitely applies to the algorithm, definitely does not apply to the algorithm, or may or may not apply (depending on information not given). No justification is required.

a)	in the best case, $O(N)$	definitely yes	definitely no	(maybe yes, maybe no)
b)	in the average case, $O(N^2)$	(definitely yes)	definitely no	maybe yes, maybe no
c)	in the worst case, $\Omega(N)$	(definitely yes)	definitely no	maybe yes, maybe no
d)	in the worst case, $O(N^2)$	definitely yes	definitely no	(maybe yes, maybe no)
e)	in the best case, $O(\log N)$	definitely yes	definitely no	(maybe yes, maybe no)
f)	in the average case, $\Omega(N \log N)$	(definitely yes)	definitely no	maybe yes, maybe no
g)	in the worst case, $\Omega(1)$	(definitely yes)	definitely no	maybe yes, maybe no
h)	in the best case, $\Omega(\log N)$	definitely yes	definitely no	(maybe yes, maybe no)