CS10 Fall 2017 Midterm 1 Answers

Question 1: Which of the following is true based on the *Programming Paradigms* lecture? The *Imperative* programming paradigm allows code written in the *Functional* paradigm within it. (so "`set X to foo(bar(baz(3)))`" would be fine)

Question 2: What was one of the lessons from the *HCI lecture*? There may be many nay-sayers as you are working to "invent the future"; history may prove them wrong. (This was evidenced by Prof. Paulos' story of the Apple Watch feature Real Touch, and the early "Google Streetview-like" Virtual Aspen which won the golden fleece award).

Question 3: Which of the following is a *true statement* based on the *Privacy* lecture? Once something is shared (via social media), it has the potential for near-instant worldwide distribution. (Thanks to digital bits and the reach of the Internet)

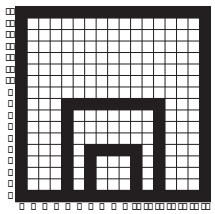
Question 4: Which of the following is *true* based on the *International Politics of Computing* lecture? *StuxNet* was an attack championed by the US and Israeli intelligence agencies on Iran's centrifuges.

Question 5: Which is a *true statement* based on the *Computing in Education* lecture? None of these

Question 6: What number do I need to add to 9_{16} to get 10_{16}? $9_{16} = 9_{10}$, and $10_{16} = 16_{10}$, so that's $16\text{-}9=7_{10} = 111_2$

Question 7a: Shade in (completely!) *all* the pixels that are filled in after 

.



Question 7b: If the test in the **repeat until** were changed to 

, what would happen after the call in (a) above? Length would start at 16, then go to 8, then 4, then 2, 1, ½ and ¼ etc and the code would run forever.

Question 8a: If they didn't store the SID, what could happen? All of the above (two "Jane Doe"s exist, both go in, one goes out, it doesn't know which is which and can't know, thus whichever guess it took, we could set it up so it would be wrong)

Question 8b: If they didn't boot the system when the lab was empty, what could happen? Even a perfect algorithm could mark a student NOT in the lab when they were [if it booted and people were in the lab, the system wouldn't know it, so given the example we did it would believe only Alan was in the lab when in fact Dan was in the lab all along having stayed up all night making this midterm.

Question 8c: What could you do if we only want to know *if the lab is empty*? We wouldn't store the students' name or the timestamp, since in the traditional technique of using DB we could just go through each SID in the DB and make sure they had eventually exited. Or we could store a Boolean value for each student (like a light switch), all reset to False or True initially, and "flipped" every time that student entered or exited (effectively ignoring which way they were going). If they were all False, we would do an OR of them all to see if anyone were in there (that OR would be True if anyone is in there, and False if the not and the lab is empty), then do a NOT at the end which would be True if the lab were empty. If they were set to True initially then they are really storing "is the person outside the lab", we would do an AND of them all to make sure they ALL were outside the lab, and just return that value which would be True when the lab is empty.

Question 8d: To find out *who* is in the lab, assume we have no control structures other than higher-order functions **map** and **keep**. I.e., no **repeat**, no **repeat until**, no **for**, no **for each**, no **combine**, and no recursion. Also assume there are no global variables, and we only have access to **DB** which can't be edited, but would simply be fed into the **map**(s) and/or **keep**(s). What "machinery" would we need to be able to report all the students who are in the lab, reported as a list of SIDs? None of the above. No amount of single **map**s or **keep**s fed into each other can create the list we need, if we can't use a global variable; we'd need a **combine** or **for each** or for to do it to remember whether a particular student was in or out. The functions passed into **map**s and **keep**s only look at events and only transform these events or remove them. We intended to disallow the functions passed into **map** and **keep** not to have any control structures involved in the functions passed into them, but we didn't say that explicitly. There was a particularly clever solution that involved **map**ping first to just pull out the SIDs. Given the list of SIDs, you could feed that into a **keep** that looked in the DB and returned whether that SID was in the DB an odd number of times, meaning they were still in the lab (that would have involved another **keep**). If so, it kept the SID, otherwise it wasn't kept. What would come out of that would be a list of the SIDs who were in the lab, BUT there might be multiple entries for each SID, one for each time they went through the door. If we just ran that through a "remove duplicates" block, it would work. Quite clever, so we also awarded full marks to "**map**'s output fed into **keep**".