CS10 Spring 2018 Final Exam Answers

Question 1: Which does NOT make the Genome assembly computing problem difficult? The letters in the genome input are all 26 letters (A-Z), a vast number of possible combinations. In fact, the input strings are only GCTA.

Question 2: What was wrong with early versions of the Google self-driving car at 4-way stops? It kept waiting for human drivers who would inch forward (looking for an advantage), so it’d be stuck there!

Question 3: Prof Marti Hearst talked about what device/concept her HCI group helped invent? Faceted Navigation (or similar web technology that lets you “prune” down what you want when searching)

Question 4: What was one memorable moment from the Alumni Panel? None of the above

Question 5 What is the Halting Problem? It was used to prove that not all problems are decidable.

Question 6 What is 46/16 divided by 129 written in Binary? 4*16+6*1=70 1*8+2*1=10 70/10=7 70/10=4+2+1=1112

Question 7a Your job is to shade in (completely!) all the pixels that will be colored in after calls to Fun with length set to 2, 4 and 8. See the diagrams

Question 7b Could Fun be rewritten iteratively? Yes, and in fact any recursive procedure can be written iteratively.

Question 8a What is equivalent to the expression reported by shuffle? helper? The helper checks whether list-of-booleans doesn’t contain False (means it only contains True). The Boolean “and” returns True if and only if both its inputs are True, so cascading it with combine is equivalent.

Question 8b What does shuffle? report for different inputs (of length at least 2)? The numbers 1 through N, in reverse order: True (and it’s supposed to be True). Is a valid shuffle, but one of the numbers is replaced with -99 False (and it’s supposed to be False) A list of N ones True (but it’s supposed to be False)

Question 9a Which of the following could be answered by analyzing only the data in DB after the exam is finished? (select ALL that apply) How many students saw the “You’re done!” screen after they finished the exam. Nope, since we don’t know how many questions are on the exam. The number of questions on the exam. Nope (witness the sample DB, we don’t know how many questions that had, only how many the students actually saw) The number of different students who took the exam. Yep, we just count the size of the set of all Unique SIDs (we could easily use a python set on the first column) The highest overall score any student achieved. Sure, that’s just the max for each student of their scores.

Question 9b Somehow the first column (Unique student ID) gets corrupted (lost). If we didn’t care about the actual student ID, but only that each student had a different ID that was consistent throughout the DB, when are we guaranteed to be able
to recreate the first column using only the other values in DB? (select ALL that apply) When there is only one student Yes, that would just be one number When no student finishes the first question Yes, because there would be no ambiguity, they’d all just be a different number. When only one student finishes the first question before time is up (i.e., clicked submit themselves) Yes, since there’d be no ambiguity. The first row would be the student who clicked submit themselves. The other rows would be people who, like student 99999999 had the system click submit for them, so they would be next. There would be one student who finished more than one, and they would have the same new Unique Student ID as the student in the first row. When exactly two students finish the first question before time is up (i.e., clicked submit themselves) No, just like the example, this one can’t be disambiguated. When no two students are ever on the same question at the same moment in time (except the first) Yes, we can do it. This one is a little complicated, but essentially it involves never having a situation that can’t be disambiguated. If there’s never two people at the same question, then as the students are moving forward, there’s never a case when (just like the example in the table) we don’t know who completed question 2 in the third row.

Question 9c Assume the first column is fine, but now the third column (Question #) gets corrupted (lost). We want to recreate it using only the values in the "Unique Student IDs" column in the DB. That is, assuming you pulled out the first column in DB as a list called SIDs, the following function should return the third column as a list.

```python
def SIDs_to_questions(SIDs):
    D = {}
    questions = []
    for SID in SIDs:
        if SID in D:
            D[SID] += 1
        else:
            D[SID] = 1
        questions.append(D[SID])
    return questions
```

Question 10a We recreated a script of playing in the interpreter, but we may have gotten some of it wrong. For each response, indicate if it is correct, and if it’s not, what the right answer should be.

```python
>>> S = "1234"
>>> S[1:3]
"23"
>>> S+S
"12341234"
>>> "3" in S
True
>>> S[2]="0"
Error (strings are immutable)
>>> [x+1 for x in range(4) if x != 2]
[1,2,4]
>>> def mystery(word):
...   if word[0] in "aeiou":
...       return word+"ay"
...   else:
...       return mystery(word[len(word)-1] + word[0:len(word)-1])
>>> mystery("school")
"olschoaay"
>>> mystery("sky")
Infinite Loop (Error also accepted, since Python errors on infinite recursion)
```

Question 10b Assume for this problem that mystery was given a word with many characters, and returned successfully. What is its running time (as a function of the length of word)? You may also assume that slicing, concatenation and determining length are constant-time operations Linear (constant work is done for each letter) & Reasonable Time

Question 11a You want to write code that will simulate your golf swing, which means randomly hitting one of these posts exactly (so there’s an equal chance of hitting -80 as there is of hitting 10). You don’t want your simulation ever to miss a post, so it should never return, say, 93. Only the #s: -100, -90, ..., 90, 100. There are two possible answers:

```python
Golf Swing Simulation
report: pick random 1 to 21 - 11 x 10

Golf Swing Simulation
report: pick random 1 to 21 x 10 - 110
```

Question 11b We want to simulate swinging the club 1,000 times, and we’d like to record every unique swing into a list called UNIQUE SWINGS. For example, if the first five swings of our simulation were 0 (great swing!), it should only add 0 once to the UNIQUE SWINGS list. We code it as shown above. What do you think about it? It’s buggy; it could add the same number twice to the list, since it’s calling a random simulation twice, once in the check (let’s say the list didn’t contain the swing, so it went into the body of the if) and once in the add (which could have simulated a swing that WAS on the list)