Open Source Software — CSCI-4961-01 — Summer 2018
Quiz 2
August 16, 2018

SOLUTIONS

1. Short answers (38 pts)

   (a) There are numerous fields in which Scientific Computing is essetntial. In class, we defined 16. Give 4 of the fields we called out. (8 pts)

   - Finite Element Modeling - Structural Safety
   - Fluid Mechanics - Aerodynamics - Aircraft Design
   - Circuit Simulation
   - Genomics and Drug Simulation
   - Image Processing Applications
   - Airline Scheduling, Linear and Nonlinear Programming Application
   - Astronomy and Space Computations
   - Simulation of Materials
   - Manufacturing Process
   - Network Simulation
   - Weapon Simulation/Modeling
   - Solving NP-hard/complete Problems
   - Protein Folding
   - SETI, Milkyway@Home
   - Physics Engines for Game Playing
   - Chemistry

   (b) Looking at the Angry Birds Game, answer the following two questions. (4 pts)

   i. Which module encapsulates the physics of the simulation?:
      - pymunk

   ii. The entirety of the physics simulation for a given time step is contained in a single call. What is the call to advance the simulation one step?:
      - space.step(dt)

   (c) From our notes, name 3 statistical languages with a BSD based license (6 points):

   - SciPy
   - Panda
   - Shogun

   (d) From our notes, name four benefits of incremental testing (8 points):

- Errors easier to isolate, find, fix reduces developer bug-fixing load
- System is always in a relatively working state - good for developers and customers.
- System complexity can be understood incrementally.
- Users can develop a better understanding of their requirements.

(e) From our notes, name what percent of open source projects have a single developer (as of 2013). Any answer within 5 points of the correct solution will be accepted. (2 points)]?

- 51

(f) From our notes, name three governance models (6 points):

- Benevolent dictator
- Meritocracy
- Dr. Who Rengeneration Model

(g) Who `"owns"` an open source project? (4 points)

- Who: No one
- Why: Forking

2. *Scientific Computation* Feel free to review, edit or run code from the Scientific Computation lecture or lab to answer the following questions. (16 pts):

   Note: If you import `networkx` into python and issue the command `help(networkx.shortest_path)` it will provide you with additional information on the shortest path algorithm. In particular, notice that if you do not provide a target, then the algorithm returns a dictionary where the keys are words that can be reached from the source and the values are the list of nodes. This information will help you with this question.

   (a) Consider your word ladder code to find the shortest path from one five letter word to another.

      i. Assuming that you are not allowed to change the order of the letters, what is the length of the longest, shortest path from the word *party*? (4 points)

         - 18

      ii. What is the final word in the path? (4 points)

         - amigo

   (b) Now consider the degree of a word as the number of other words that can be made from it by changing a single letter and keeping the word order the same.

      i. What words have the maximum degree of 25? (4 points)

         - cores, bares

      ii. There are far more words with degree 0. How many are there? (4 points)

         - 671

3. *Statistical Computation* Feel free to review, edit or run code from the Statistical Computation lecture or lab to answer the following questions. (10 pts):

   (a) Consider the topmovie data we used in our R examples.

      i. What is the Minimum, 1st Quartile, Median, Mean, 3rd Quartile, and Maximum values for the box office? (4 points)

         - 52.58, 70.28, 93.60, 117.47, 134.62, 759.56

      ii. If we want to look at the relationship between the box office and the year, how would we generate a scatterplot with year on the *X-axis* and box office on the *Y-axis*? Type the command below: (6 points)

         - plot(topmovies$year, topmovies$box)

4. `Testing and Continuous Integration` Feel free to review, edit or run code from the Testing and Continuous Integration lecture or lab to answer the following questions. (15 pts):

   Consider the following Python module implementing the merge_sort algorithm:

```
import random

def merge(L1, L2):
  """
  Assume L1 and L2 are sorted.
  Create a new list L that is the merged
  version of L1&L2.
  """
  L = []
  i = 0
  j = 0
  while i < len(L1) and j < len(L2):
    if L1[i] < L2[j]:
      val = L1[i]
      L.append( val )
      i += 1
    else:
      val = L2[j]
      L.append( val )
      j += 1
  ## at this point, either L1 or L2 has run out of values
  ## add all the remaining values to the end of L.
  L.extend(L1[i:])
  L.extend(L2[j:])
  return L

def merge_sort_recursive(L):
  """
  Complexity: O(n logn)
  The function calls itself recursively logn times,
  and each time about n elements are merged.
  """
  if len(L) <= 1:
    return L

  length = len(L)
  mid = length // 2
  left = merge_sort_recursive(L[:mid])
  right = merge_sort_recursive(L[mid:])
  return merge(left,  right)

if __name__ == "__main__":
  ##Testing code
  k = 10
  L = list(range(k))
  random.shuffle(L)
  print("Before:", L)
  L = merge_sort_recursive(L)
  print("After:", L)
```

Generate a python file, *test_merge.py* that uses the unittest framework to thoroughly test the *merge* and *merge_sort_recursive* functions. Assume *white box* testing. You should be able to come up with at least 3 test cases for *merge* and at least 2 test cases for *merge_sort_recursive*

```
import unittest
import merge_sort

class TestMS(unittest.TestCase):

  def setUp(self):
    pass

  def test_merge_1(self):
    L1 = list(range(0, 10, 2))
    L2 = list(range(1, 10, 2))
    ans = list(range(10))

  def test_merge_2(self):
    L1 = list(range(0, 10))
    L2 = []
    ans = list(range(10))
    self.assertEqual( merge_sort.merge(L1, L2), ans)

  def test_merge_3(self):
    L1 = list(range(0, 10))
    L2 = []
    ans = list(range(10))
    self.assertEqual( merge_sort.merge(L2, L1), ans)

  def test_merge_4(self):
    self.assertEqual( merge_sort.merge_sort_recursive([]), [])

  def test_merge_5(self):
    self.assertEqual( merge_sort.merge_sort_recursive(list(range(9, -1, -1))), list(range(10)))

  def test_merge_6(self):
    self.assertEqual( merge_sort.merge_sort_recursive([1, 4, 2, 0]), [0, 1, 2, 4])

if __name__ == '__main__':
  unittest.main()
```

5. `MongDB` Feel free to review, edit or run code from the MongoDB lecture or lab to answer the following questions. (15 pts):

   Consider the definitions file we used for our MongoDB lab, particularly for *checkpoint4.py* and *checkpoint5.py*.

   (a) Write the sequence of commands to reset the database to the contents of the file `definitions.json`. Use `$` to indicate commands typed on the command line and `>` to indicate commands typed into the mongo prompt. You should assume that the database is already in mongodb with the name `mongo_db_lab`, i.e. (3 points)

```
$ mongo
MongoDB shell version v4.0.0
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 4.0.0
> show dbs
admin         0.000GB
config        0.000GB
local         0.000GB
mongo_db_lab  0.000GB
> quit()
$
```

```
$ mongo
MongoDB shell version v4.0.0
connecting to: mongodb://127.0.0.1:27017
MongoDB server version: 4.0.0
> use mongo_db_lab
switched to db mongo_db_lab
> db.definitions.drop()
true
> quit()
ubuntu@eec278129a30:~/mongodb_lab$ mongoimport --host=127.0.0.1 --db mongo_db_lab --collection defini
2018-08-15T16:21:34.169+0000 connected to: 127.0.0.1
2018-08-15T16:21:34.205+0000 imported 147 documents
ubuntu@eec278129a30:~/mongodb_lab$
```

(b) How many entries in the database have the string `" RPI "` in them? (2 points) 13

(c) Now write Python code to search through the database, replace every occurrence of `" RPI "` in a definition with `" Rensselaer Polytechnic Institute "`, and write the modified records back out to the database. Be careful with the search and replacement strings. You do not want to change the string `"(RPI)"`. (Also, if you are testing your code make sure that you reset your database before running.) (10 points)

```
from pymongo import MongoClient

client = MongoClient()

if __name__ == '__main__':
  db = client.mongo_db_lab
  collection = db.definitions

  # All records
  all_documents = list(collection.find())

  print(len(all_documents))
  found = 0
  ids = []
  for document in all_documents:
    if ' RPI ' in document['definition']:
      document['definition'] = document['definition'].replace(' RPI ', ' Rensselaer Polytechnic Insti
      collection.update({'_id':document['_id']}, document)
      ids.append(document['_id'])
      found += 1

  print(found)
  for id in ids:
    print(collection.find({'_id': id})[0])
```

6. `Virtualization and Containers` Feel free to review, edit or run code from the Virtualization and Containers lecture or lab to answer the following questions. (10 pts):

   (a) Look at the following command to run a docker instance.

   ```
   $ docker run -i -t -p 8888:8888 ubuntu:latest
   ```

   Briefly describe what the following parts of the line do to the execution: (5 points)

   i. -i: Run in interactive mode

   ii. -t: Allocate a pseudo-TTY

   iii. -p 8888:7777 publish container port 7777 to host port 8888

   iv. ubuntu Image to be run

   v. :latest Version (tag) to run on the selected image

   (b) Assume you have the Dockerfile below:

   ```
   # Comments in Dockerfiles
   FROM ubuntu:latest

   RUN apt-get update
   RUN apt-get install sudo
   RUN apt-get --yes install apt-transport-https
   RUN apt-get --yes install dbus
   RUN apt-get --yes install python3
   RUN apt-get install python3-pip

   RUN mkdir -p /data/db

   RUN useradd -d /home/ubuntu -ms /bin/bash -g root -G sudo ubuntu
   RUN echo "root:Docker!" | chpasswd
   RUN echo "ubuntu:ubuntu" | chpasswd

   USER ubuntu
   WORKDIR /home/ubuntu
   ```

   i. Write the command to compile this docker file into an image with the name profsfavorites (Assume the Dockerfile is in the current directory) (2 points):

   ```
   > docker build -t profsfavorites .
   ```

   ii. What commands would be added to the Dockerfile to install `git` and `vim` (both are available through apt-get) (2 points)?

   ```
   RUN apt-get --yes install git
   RUN apt-get --yes install vim
   ```

   iii. How many user accounts are there on the machine (do not count the root account) (1 point)?

   ```
   1 - ubuntu
   ```