CS 5010 – Exam – Study Guide

1. Data conceptualization
    a. What is data?
    b. Different types of data
        i. Qualitative
        ii. Quantitative data (discrete and continuous)
        iii. Categorical
    c. Properties of data
        i. Source
        ii. Quality – data maintained and data is clean
        iii. Scale – data is representative?
        iv. Variety – different types of data
    d. Value of data is extracted through context and presentation
        i. Case studies
    e. Data □ Information □ Knowledge
        i. Difference between each of them
    f. Data products
2. Python
    a. Be able to read and understand code
    b. Be able to show the output of a piece of code
    c. Be able to compare two pieces of code
    d. Describe benefits of applying object-oriented principles
    e. Describe what the major data structures are and their properties (and scenarios in which they would be best suited)
    f. Describe the major functions used and their properties (and scenarios in which they would be best suited) In particular, be familiar with the *linear (sequential) search* and *binary search* algorithms (you will *not* be asked to code these on the exam, however you will be expected to demonstrate knowledge of the algorithms by answering questions about them similar to the ones we have done in class.) Built-in sorting algorithms will not be covered on this exam.
    g. A high-level question might be asked on NumPy & Pandas (*no coding* will be involved, however)
    h. Explain what makes “good quality” code

3. Testing & Debugging

a. Explain what testing is
b. Why is testing important?
c. Be able to write a unit test given a scenario (code will be provided)
d. Explain the difference between white-box and black-box testing
e. Best practices
f. Test-driven development – including benefits
g. Be able to write a unit test, be familiar with at least the top 3 assert statements (assertEqual, assertTrue, assertFalse)
h. Amount of *time* spend on testing vs. writing main code
i. Amount of *code written* for testing vs. writing main code
j. "*Code a little, Test a little*"

4. Basics of Software Engineering

a. Software is becoming so prevalent in nearly everything we do
b. Failures impact everyone
c. Cost of failure becoming *very high*
    i. Financial
    ii. Loss of life
    iii. Time
    iv. Loss of equipment
    v. Inconvenience
d. Describe what Software Engineering is
    i. "Technological"
    ii. "Managerial"
    iii. "Systematic"
    iv. Development that is "on time" and "within cost estimates"
e. Programming in the large vs Programming in the small
f. Software Development Lifecycle and Phases
    i. Requirements
    ii. Design
    iii. Integration
    iv. Coding/Implementation
    v. Maintenance
    vi. (… don't forget testing *throughout!*)
g. Relative cost or time per phase
    i. Which phase takes up the biggest chunk? Be able to explain why
h. Requirements
    i. Client point-of-view
    ii. Objective

iii. Validation tests

i. Functional requirements
   i. *"What"* the system or application does (describes a function or behavior)

j. Non-functional requirements
   i. *"How"* the system does things (qualifies a given functional requirement)
   ii. A *"quality"* or property the product has (e.g. efficiency)

k. Constraint – a *design requirement* directly from the *client*

l. Cost increases as faults are found later / importance of front-end

5. Software Development Methods/Models

a. Waterfall model
   i. Pros/Cons
   ii. Understand it is "historical" (not in use today as-is)

b. Spiral model
   i. Pros/Cons
   ii. Understand it is "historical" (not in use today as-is)

c. Why learn about these methods?
   i. Inject quality into software
   ii. Avoiding problems early that cause huge problems later
   iii. Software engineering is not just about writing code

d. "Plan-driven" methodologies
   i. "Think it through to the end"
   ii. Document-driven
   iii. Heavy-weight
   iv. How "traditional" engineering has been done

e. "Agile" methodologies
   i. "Focus on mostly just the next thing"
   ii. Less focus on documentation
   iii. Frequent interaction with customer
   iv. Iterative development
   v. Flexible, ability to change
   vi. Light-weight
   vii. Relatively new (since 1990s)

f. Agile approaches
   i. Extreme Programming (XP)
   ii. XP made popular techniques that have been widely adopted:
      1. Pair-programming
      2. Unit-testing

3. Test First (test driven development)

iii. SCRUM (see next section)

6. SCRUM

a. Explain what Scrum is– (Agile process)
b. Focus on delivering the highest business value in the shortest time
c. User Stories (and Acceptance criteria) – *"As a ___, I want ___."*
d. Product Backlog
e. Sprint and Sprint Backlog
f. Tasks and task assignment