CS 220 MidTerm Exam Practice Questions

1. True/False:

   (a) A function that detects a violation of its precondition should print out an error message.

   (b) Ensuring that the precondition is met is the responsibility of the implementation code, and ensuring the postcondition is met is the responsibility of the client code.

   (c) A well-designed function avoids undocumented side-effects.

   (d) Using the same computer, programming language, and input data, executing a $\Theta(n)$ algorithm must be faster than executing an algorithm that is $\Theta(n^2)$.

   (e) Binary search is a $\Theta(n \log n)$ algorithm.

   (f) ADTs are easily implemented as objects in an object oriented language such as Python.

   (g) When designing a program, a good way of locating potential objects is to look at the verbs in the program description.

   (h) Class variables are shared by all instances of a class.

   (i) In Python, instance variables are accessed through the `self` parameter.

   (j) Every $\Theta(n)$ algorithm is O(n^2).

   (k) Unit tests allow a component of a program to be tested in isolation.

   (l) The indexing operation on a Python list returns a sublist of the original.

   (m) The constructor for the Deck class of Chapter 3 produces a deck of cards in a random order.

   (n) Python lists are implemented using dynamically resized arrays.

   (o) Arrays allow for efficient random access (indexing).

   (p) Inserting into the middle of a Python list is a $\Theta(n)$ operation.

   (q) In Python, built-in operators like + and * cannot be used with objects from non-builtin (programmer defined) classes.

   (r) Selection sort is a $\Theta(n^2)$ algorithm.

2. Write a fragment of Python code for each of the following tasks. Assume `mylist` is a list of numbers.

   (a) print out all the odd integers between 1 and 2000, one per line.

   (b) write an accumulator loop that adds up all the items in `mylist` by looping over the VALUES in the list.

   (c) write an accumulator loop that adds up all the items in `mylist` by looping over the indexes of the list.

   (d) write an accumulator loop that adds up the items at the ODD positions (indexes) of `mylist.`

   (e) write a Boolean expression using the variable year that is True when year is a leap year and False when it is not.

3. Suppose you have a list containing 256 items in sorted order.

   (a) What is the maximum number of “steps” required to determine if a particular item is in the list using linear search?

   (b) How many “steps” would be needed using a binary search?

   (c) In general, how many steps are needed for linear search and binary search for a list of size n?

4. Define/explain the following concepts as they relate to program specification and design:

   (a) design by contract

   (b) assertion

   (c) `self` parameter

   (d) abstract data type

   (e) implementation independence

5. Algorithm analysis.

   (a) Place these algorithm classes in order from best to worst: n, n^2, $\sqrt{n}$, 2^n,$\log n$, n^3, $n \log n$.

   (b) If a $\Theta(n^2)$ algorithm requires 6 seconds to execute on an input of size 10,000. Approximately how long should it take on an input of size 20,000?

   (c) If a $\Theta(2^n)$ requires 5 seconds for an input of size 5, approximately how long should it take for an input of size 10?

6. In class, we examined two different implementations of the `Card` class.

   (a) Show a line of code that would create an queen of diamonds.

   (b) Show a fragment of code that would create a list containing all 52 possible cards.

   (c) Assuming the concrete representation of a card is a single integer from 0 to 51 stored in an instance variable called `cardnum`, give an implementation of the `rank()` method of the `Card` class.

7. Consider the following function specification:

```
def average(nums):
    """pre: nums is a list of numbers
       post: returns the average (mean) of the numbers in nums
    """
```

   (a) What is missing from the precondition of this function?

   (b) Write an implementation of this function.

8. Give a theta analysis for each of the following code fragments:

```
# part a)
for i in range(n):
    for j in range(n):
        print(i,j)

# part b)
for i in range(n):
    for j in range(50):
        print(i,j)

# part c)
for i in range(n):
    for j in range(i,n):
        print(i,j)

# part d)
for i in range(n):
    print(i)
for j in range(n):
    print(i,j)

# part e)
for i in range(n):
    while n > 1:
        n = n // 2
```

9. Write a Python class definition for a `Multiplier` object that can be used to multiply a sequence of numbers. Your methods should include appropriate pre- and postconditions. Your class should include the methods required by the following program that calculates the product of 10 numbers entered by the user:

```
def main()
   mult = Multiplier();
   for i in range(10):
      x = float(input("Enter a number: "))
      mult.times(x)
   print("The product of your numbers is:", mult.product())
main()
```

10. Explain the following elements of the Python unittest framework:

    (a) `TestCase`

    (b) `unittest.main()`

    (c) `assertEqual()`

11. Give a theta analysis of the efficiency of each of the following operations using Python lists.

    (a) Appending an item to the end of a list.

    (b) Inserting an item at the front of a list.

    (c) Building up a list of n items by appending them one-by-one to the back of an initially empty list.

    (d) Building up a list of n items by inserting them one-by-one at the front of an initially empty list.

    (e) Sorting a list using the built-in sorting algorithm.

12. Suppose that you have the `Deck` and `Hand` classes from Chapter 3 available. Write a fragment of code to do each of the following tasks.

    (a) Print out the names of all 52 cards of a deck in standard order.

    (b) Print out the names of 13 random cards from a deck.

    (c) Choose 13 cards at random, and print them out in sorted order.

    (d) Display four 13-card hands dealt from a shuffled deck.

13. Give a theta analysis of the efficiency of each of the following operations using Python lists.

    (a) Appending an item to the end of a list.

    (b) Inserting an item at the front of a list.

    (c) Building up a list of n items by appending them one-by-one to the back of an initially empty list.

    (d) Building up a list of n items by inserting them one-by-one at the front of an initially empty list.

    (e) Sorting a list using the built-in sorting algorithm.

14. Suppose that you have the `Deck` and `Hand` classes from Chapter 3 available. Write a fragment of code to do each of the following tasks.

    (a) Print out the names of all 52 cards of a deck in standard order.

    (b) Print out the names of 13 random cards from a deck.

    (c) Choose 13 cards at random, and print them out in sorted order.

    (d) Display four 13-card hands dealt from a shuffled deck.

15. Consider the problem of keeping track of a large collection of club members. Your system must implement the following operations:

```
add(person) - Adds person object to the collection
remove(name) - Removes person with the given name from the collection
lookup(name) - Returns the object for the person with the given name
listAll() - Returns a list of all the objects in alphabetical order
```

(a) Choose a concrete representation for implementing this collection as a container class and write appropriate `__init__` and `add` methods for your representation.

(b) Give a theta analysis of all four operations for your class. *Note*: *you do not have to write the other operations, just evaluate their efficiency*.