CSI 33 Midterm Exam In-class Practice

Part 1 Answer True/False and Multiple Choice questions

1. Which of the following is a Θ(n) operation?
- (a) Sorting a list with Selection sort
- (b) Finding the ith item in a Python list.
- (c) Re-assigning the element at the end of a Python list.
- (d) Deleting an item from the middle of a Python list.

2. Which of the following is **not** true of Python dictionaries?
- (a) They are implemented as hash tables.
- (b) Lookup is very efficient.
- (c) Values must be immutable.
- (d) All of the above are true.

3. How many iterations will the while loop of the *Binary Search* do when searching for 21 in the sequence [1, 5, 12, 14, 17, 21, 28]? Use the Binary Search algorithm I presented in class.

- (a) 5
- (b) 4
- (c) 3
- (d) 2

Part 2. Answer short-answer questions

1. Consider the following code fragment:

```
from ListNode import *

z = ListNode(34)
y = ListNode(25,z)
x = ListNode(12,y)
t = ListNode(20,y)
```

What will be produced by this code fragment (draw a pictorial representation)?

For your reference, the definition of the `ListNode` class:

```
class ListNode:

    def __init__(self, item = None, link = None):

        '''creates a ListNode with the specified data value and link
        post: creates a ListNode with the specified data value and
link'''

        self.item = item
        self.link = link
```

2. Give a theta analysis of the time efficiency of the following code fragment. Provide explanations.

```
n = int(input("Enter a positive integer:"))
myList = []
while n > 1:
        myList.insert(0, n)
        n -= 3
```

T(n) = Θ (   )

3. Give pictorial representation of the Python's memory during execution of the code given below. Show the result of print statements.

```
def func(a,b,c):
     a.append(c)
     b = b + ", world!"
     c = c/5
     a = [1,2,3]
     print(a,b,c)

def main():
     l = ['a','b']
     d = "Hello"
     k = 25
     func(l,d,k)
     print(l,d,k)
```

Part 3. Coding and related to coding questions

Write definitions of two new abstract data types (ADTs): **Library** and **PatronsRecord**, in Python or C++.
Assume that we want to create a simple **library**: a library has a *list of different book titles* (assume that we ignore the information about the author, different publishers, publish year, and other information; we are interested only in the title of the book; also assume that there infinitely many books of each title), a *number of books on loan* at this moment, a *list of patrons' records*, *library name* and *library id*.
Each **PatronsRecord** has a *library card id*, *name*, and the *list of books on loan* (these will also be book titles).
Feel free to add more attributes to any of the classes as you see fit.

Each **Library** instance should be able to:
(a) return the total number of book titles the library has,
(b) return the number of books on loan,
(c) return the list of patrons' names and their card ids who have books on loan,
(d) loan a book with title from the library by a patron (note that when a book is loaned, its title is not removed from the library's list of book titles),
(e) return the book back to the library by proving the title of the book and patron's information
(f) return the alphabetized list of book titles the library has, (make sure not to return the reference to the book titles list attribute of the class otherwise it can be modified outside the class),
(g) add a new book title to the list of books the library has,
(h) remove a book title from the list of books the library has

Each **PatronsRecord** instance should be able to:
(a) return the patron's card id and name,
(b) return the alphabetized list of book titles the patron borrowed and did not return yet,
(c) return the number of books on loan,
(d) add a book title to the list of books on loan,
(e) remove a book title from the list of books on loan.

Comment: if you decide to use C++ for this problem, announce that **Library** class is a *friend* of **PatronsRecord** class, so that **Library** instances have access to attributes of **PatronsRecord** instances.

After you finished with the two new ADTs, answer the following questions, but do not change your code:
1) Assume book titles will be often added and removed, and we will be calling the operation of "return the alphabetized list of book titles the library has" very often. If we will be calling the method `sort()` for the list of book titles every time, each call is Θ(n log n).
Is there a way to improve this time?

It might be the case that your definition of this operation already performs better than Θ(n log n).
In this case just describe your method's time efficiency and how you reached it.

2) Libraries sometimes have several copies of the same book. How can we store this information?

Here you can find class diagrams that will help you with this project.
The top field is occupied with the name of the class, the middle field is for data attributes and the bottom field is for the methods of the class.

Library	PatronsRecord

name //library name
id // library ID
patrons // container for patrons
bookTitles // container for the list
//of book titles
loaned = 0 // number of books on loan
	cardID
	
	name
	
	booksOnLoan

getNumBooks() returns the total number of book titles the library has as integer BooksOnLoan() returns the number of books on loan PatronsStillHaveBooks() returns the list of patron's names with their card ids who have books on loan loan(title, patron) patron loans a book with title from the library if the title is available, returns True, otherwise returns False returnBook(title, patron): return the book back to the library by patron's BooksList() returns a copy of the alphabetized list of book titles the library has add(title) add a new book title to the library remove(title) remove a book from the library by the title, if the book title is not found returns False, otherwise removes the book title from the list of book titles	info() returns the patron's name and card id booksLoaned() returns the alphabetized list of book titles the patron borrowed and did not return yet getNumBooksLoaned() returns the number of books on loan by this patron add(title) adds a book title to the list of books on loan remove(title) removes a book title from the list of books on loan