BRONX COMMUNITY COLLEGE

of The City University of New York

DEPARTMENT OF MATHEMATICS and COMPUTER SCIENCE

CSI 33

Midterm Exam Sample

Programs must be sent to my e-mail : **natna20@gmail.com**. All other things are at your discretion: you can also send them via e-mail, or submit them in class. Note that everything must be submitted before you leave the class or the time is up. The exam is open book, open notes, but computer use is limited to only Python Idle and electronic version of our book.

1. Answer True/False and Multiple Choice questions (5 questions)

   (a) Class variables can be shared by all instances of the class (state True or False).

   (b) Which of the following is a $\Theta(n)$ operation?
      i. Sorting a list with Selection sort
      ii. Finding the ith item in a Python list.
      iii. Re-assigning the element at the end of a Python list.
      iv. Deleting an item from the middle of a Python list.

   (c) Which of the following is **not true** of Python dictionaries?
      i. They are implemented as hash tables.
      ii. Lookup is very efficient.
      iii. Values must be immutable.
      iv. All of the above are true.

   (d) By definition, a queue must be a(n):
      i. array-based structure
      ii. linked structure
      iii. FIFO structure
      iv. LIFO structure

   (e) A queue allows for the inspection of items at either end (state True or False).

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

```
n = int(input("Enter a positive integer:"))
l = []
while n > 1:
   l.append(n)
   n = n/3
```

3. The integers $2, 17, 2, 34, 12$ and 9 are inserted into the **Queue** in the given order. Then three elements are `dequeued`, and the following elements are `enqueued`: $8, 7, 5$, and 1 in the given order. Give the order in which all the values will be retrieved from the **queue**.

4. Show pictorially how the following postfix expression can be evaluated using **Stack** container: $1 \;\; 2 \;\; 3 \;\; * \;\; + \;\; 6 \;\; 2 \;\; / \;\; -$.

   Don't forget to show the result of evaluation.

5. Give pictorial representation of the Python's memory during execution of the following code. 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','c']
    d = "Hello"
    k = 25
    func(l,d,k)
    print(l,d,k)
```

6. For the given function definition:

```
def function(items):
    """ pre: items is a list of integers """

    if items == []:
        return 0
    else:
        return items[0] + function(items[1:])
```

   (a) Trace the call of `function([5,8,9])`
   (showing the recursive calls and return values).

   (b) Figure exactly how many additions does it do.

7. Write a program to evaluate a valid prefix expression (you can use your homework, where you wrote the code for postfix expression evaluation). For testing

   use the following expressions:

   - $+ * 123 = 1 * 2 + 3 = 5$
   - $-5 + *345 = 5 - (3 * 4 + 5) = 5 - 17 = -12$
   - $* + 234 = (2 + 3) * 4$

8. Write definitions of two new abstract data types (ADTs): `Patron` and `Library`.

   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 a title of the book), a **list of patrons**, **library name** and **library id**. Each *patron* has a **name**, **id**, and the **list of books on loan** (these will also be book titles). Feel free to add more attributes as you see fit.

   Each `Patron` instance should be able to:

   (a) return the patron's name and id,

   (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.

Each `Library` instance should be able to:

(a) return the total number of book titles the library has,

(b) return the list of patron's names who have books on loan,

(c) 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),

(d) loan a book from the library by the title of the book,

(e) return the book back to the library,

(f) return the number of books on loan,

(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.

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 or 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 $\Theta(n \log n)$.

Is there a way to improve this time?

It might be the case that the way you wrote the code, this operation already performs better than $\Theta(n \log n)$. 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?