BRONX COMMUNITY COLLEGE
of the City University of New York
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE

CSI31 **Sample Final Exam**

NAME_____

Directions: Answer all questions in Part 1, and two of the three in Part 2. You may refer to the textbook, handouts, or notes taken in class. Do not use any computer or calculator. If you prefer, each question can be answered in the exam booklet provided. Hand in all papers, with answers clearly marked and with your name on each sheet. The questions are worth the points indicated.
PART 1: Do all four problems 1-4:

1. (10 points) Answer TRUE or FALSE:

   i. A function has access to all variables used in a program.

   ii. An `else` statement must have a matching `if` statement.

   iii. If you need to write a loop that will execute a fixed number of times, you should use a `for` loop.

   iv. A `while` loop written as `while True` will never exit no matter what the loop body contains.

   v. The boolean expression `random() >= 0.25` is likely to be true 25% of the time.

   vi. With top-down design, the function in your design you will usually implement and test first will not be the topmost function.

   vii. Unit testing is testing separate parts/units of the program separately.

   viii. By convention, a variable in a method which is an instance variable for the class is prefixed with `self`.

   ix. The Python dictionary class is not mutable.

   x. One reason to use functions is to avoid duplicating code.

2. (20 points) Write Python code that uses a loop to let the user enter a series of integers, and prints the sum of the squares of all the entered integers. The user exits the loop by pressing `Enter` without any number. Identify any accumulator variables used.

3. (10 points) Write the decision flowchart for this `if-elif-else` statement:

```
if n > 0:
  print(n, "is positive")
elif n < 0:
  print(n, "is negative")
else:
  print(n, "is zero")
```

4. (20 points)
Use a top-down design to code the `main()` function for an ATM which will allow bank customers to (1) log in with ID and password to see the accounts they have, (2) select an account and be asked whether they want to deposit or withdraw an amount, (3) choose which type of transaction and see an entry field for the amount, then (4) perform the transaction of that type and print a receipt for that transaction with the updated account balance and log the customer out. The body of `main()` should be a sequence of function calls, each function performing one step of the task. When appropriate, each function will return a value which may be a parameter in the functions called afterward. Do not write the functions themselves, but write a comment with each function called, to guide the next stage of progamming when the functions will be implemented.

PART 2: Choose any two of the three problems 5-7:

5. (20 points) You are upgrading the ATM system of the previous problem using a new object-oriented design. You must create a Python class, `SavingsAccount`, whose objects have the following instance variables: `accountNumber`, `customerName` and `balance`.

   Write the constructor method, `__init__(self, acctNum, acctName, acctBal)` which creates an account with the given values for the instance variables. Also write methods `getBalance(self)`, which returns the `balance` of an account, `deposit(self, amt)`, which increases the balance by `amt`, and `withdraw(self, amt)`, which decreases the account balance by `amt`. Assume the balance, deposit, and withdrawal amounts are all in cents (integers).

6. (20 points) As a system developer, you are writing a function in Python, `maxValue(l)`, which takes a list of integers `l` and returns the maximum value. So far it crashes when the parameter `l` is an empty list.

   A. (10) Write code to add to the beginning of the function body of `maxValue(l)` to raise the exception `IndexError` if the list `l` is empty.

   B. (10) There is test code for this function, an infinite test loop, but it crashes whenever an empty list is entered:

```
while True:

  l = eval(input("Enter a list of integers:"))
  print(maxValue(l))
```

   Modify this code by using a `try` block and an `except` clause, so that the loop handles the exception raised by `maxValue(l)`, and will continue even after an empty list is entered.

7. (20 points) The following test code calls two functions which are supposed to add the value of the second variable to the first when they are passed as actual parameters. The first, `addItems1(item1, item2)`, takes two integers as parameters, while the second, `addItems2(items)`, takes a list `items` of two integers as a parameter:

```
def addItems1(item0, item1):  # item0, item1 are two integers
  item0 = item0 + item1

def addItems2(items):  # items is a list of two integers
  items[0] = items[0] + items[1]

def test():
  n0, n1 = 6, 7
  addItems1(n0, n1)
  print(n0)
  nlist = [6, 7] # nlist[0] = 6, nlist[1] = 7
  addItems2(nlist)
  print(nlist[0])
```

A. (15) Draw box-and-arrow diagrams showing local variable values, including list values, after each statement as the test is run.

B. (5) In `test()`, which function successfully adds the second value to the first after returning?
(a) Neither (b) `addItems1` (c) `addItems2` (d) both