SOPH 303 The Digital World

Term 3. January 28-May 3. 2013

Midterm Exam

March 6, 2013 2:30-4:30pm Room: Cohort Classroom

Most recent update: March 1, 2013

Note:

1. This exam has three parts. First try to complete Part A, then move to Part B and finally try Part C. You can secure a maximum of 120 points that includes 20 bonus points. If you do not attempt Part C then you can obtain a maximum score of 100 (which will be used as 15 out of 15 in your final course grade.)
2. For bonus questions your program must be **fully** correct; no partial credit will be given. You may be able to earn partial credit for questions in Parts A and B as the graders will look at your program in case it is not fully correct.
3. You may use the Internet to look at APIs, consult your notes and books for help with this exam. Please do not use the Internet to "hunt" for solutions to the problems; by doing so you will be doing a disservice to yourself. *You are not allowed to consult anyone inside or outside of the classroom other than the course instructors for SOPH 303.*
4. Use Idle to test your programs. Once you are satisfied, enter your program into Tutor and either save it or submit. In case you decide to save, then you MUST submit it before the end of the exam.

Summary:

Category	**Description**	**Number of Problems**	**Total points**
Part A	Very short programs	3	30
Part B	Short programs	2	70
Part C	Bonus questions	2	20

Category A: Very short Python programs. Total problems: 3. 10 points each.

1. Write a program that (a) reads two integers $n1$ and $n2$ from the keyboard and (b) creates and prints a list of all odd integers between $n1 > 0$ and $n2 \geq n1$ (inclusive of $n1$ and $n2$).

   Sample tests:

   Test 1

Input:	n1=2, n2=7
Output:	[3, 5, 7]

   Test 2

Input:	n1=13, n2=28
Output:	[13, 15, 17, 19, 21, 23, 25, 27]

2. Write a function named `findKey()` that takes a dictionary `d` and a string `s` as inputs. It returns `True` if s is a key in `d`, otherwise it returns `False`.

   Sample tests:

   Test 1

Input:	`d`=[1: 'Tokyo', 2: 'Qatar', 3: 'Singapore'], `s`='Qatar'
Output:	False

   Test 2

Input:	`d`=[1: 'Tokyo', 2: 'Qatar', 3: 'Singapore'], `s`='2'
Output:	True

   Test 3

Input:	`d`=[1: 'Tokyo', 2: 'Qatar', 3: 'Singapore'], `s`='Kuala Lumpur'
Output:	False

3. Write a function named `check()` that takes a string `s` and an integer `n` as inputs. It returns the sum of `n` and the number represented by `s`. If `s` does not represent any number then it returns `n`. [Hint: Think about using `try-except` to solve this problem.]

   Sample tests:

   Test 1

Input:	s="12", n=9
Output:	21

   Test 2

Input:	s="Hello", n=10
Output:	10

Category B: Short programs. Total problems: 2

1. In this problem you are required to write three functions named `getDict()`, `search-Dict()`, and `test()`.

   (a) `getDict(f)`:

   This function takes a file object `f` as input (NOT the name of a file, but **a file object**) . The file pointed to by `f` contains several lines of data as shown in the sample test cases below. Each line contains four items: a string `r`, string `s1`, another string `s2`, and a number `n`, where `r` denotes an athletic event, `s1` the name of the person who holds a record for that event, `s2` the name of the country to which the person belongs, and `n` the time in seconds. Assume that the name of the record holder and that of the country do not have any space.

   The `getDict()` function reads data from the file and returns a dictionary in which each entry has the value of `r` as the key and a list containing values of `s1`, `s2`, and `n` as its value.

   15 Points

   (b) `searchDict(d)`:

   This function takes a dictionary `d` as input. This dictionary has exactly the same structure as the one created by `getDict()`. It prompts the user to enter an integer `e` that denotes an athletic event. The prompt message is`‘‘Enter an event:"`. If `e` exists in `d` then the function prints the name of the record holder, the corresponding country, and the time–each separated by one space. If the event does not exist then the function prints “No such event.” and continues to prompt the user. The function terminates without returning anything, when the user types an asterisk (*) in response to the prompt. It prints the message “Bye!” before returning.

   15 Points

   (c) `test()`:

   This function opens a file named `data.dat` using `f` as the handle (or pointer) to the file. It then calls `getDict()` with `f` as input. Let `d` denote the dictionary returned by `getDict()`. `test()` then calls `searchDict()` with `d` as input. `test()` terminates when `searchDict()` returns.

   10 Points

   Sample tests:

   Contents of file `data.dat`:

   100m Bolt Jamaica 9.58

   200m Bolt Jamaica 19.19

   400m Johnson USA 43.18

800m Rudisha Kenya 100.91

1000m Ngeny Kenya 131.96

Output from `getDict()`:

{100m: [“Bolt”, “Jamaica”, 9.58], “200m”: [“Bolt, “Jamaica”, 19.19], “400m”: [“Johnson”, ”USA”, 43.18], “800m”: [“Rudisha”, “ Kenya”, 100.91], “1000m”: [“Ngeny”, “Kenya”, 131.96]}

Input to `searchDict()`: *This is the same as the dictionary above.*

User interaction with `searchDict()` (Prompts and computer output are in bold.):

Enter an event: 100m

Bolt Jamaica 9.58

Enter an event: 1000m

Ngeny Kenya 131.96

Enter an event: Wrestling

No such event.

Enter an event: 800m

Rudisha Kenya 100.91

Enter an event: *

Bye!

2. Create a class named `Animal`. The constructor (i.e., the `__init__` function) takes a list of three items as inputs. The first item on the list is the name of the animal (e.g., “Snoopy”, the second element is the weight of the animal represented as a number (e.g., 12.7), and the third element is the type of the animal (e.g., “Dog”).

10 Points

The class should have three other methods named `setAttrib()`, `__call__()` and `__str__()`. The `setAttrib()` method takes an index `k` and an attribute value `v`. It sets the attribute that corresponds to `k` to value `v`. The `__call__` method returns a list of all the attributes of that animal (e.g., its name, weight, and type). The `__str__` method returns a string consisting of the name, weight, and type, separated by one space each, formatted as shown below.

20 Points

A sample interaction with the `Animal` class follows; you can use the commands below to test your code.

```
>>> a=Animal(["Snoopy", 10, "Dog"])
>>> print a
Name: Snoopy Weight: 10Kg  Type: Dog
```

```
>>> print a()
['Snoopy', 10, 'Dog']
>>> b=Animal(["Viva", 7, "Dog"])
>>> b()
['Viva', 7, 'Dog']
>>> print b
Name: Viva Weight: 7Kg  Type: Dog
>>> a.setAttrib(1,12)
>>> print a
Name: Snoopy Weight: 12Kg  Type: Dog
```

Category C: Bonus questions. Total problems: 2

1. *This is not a programming problem.* An astronaut walks into a classroom on a strange planet and sees the following equation.

   $5x^2 + 50x + 125 = 0$

   The astronaut is informed that the roots of the equation are -8 and -5. What is the base of the number system being used in the equation? Note: All information given to the astronaut is in the unknown number system. You must clearly explain how your answer is derived.

   10 Points

2. You are required to write a function named `seriesSummer()`. This function takes four inputs: the first term (`firstTerm`) in an infinite series, the ratio `r` of two successive terms in this series, a small number named `epsilon`, and `termMax`–the maximum number of terms to be added. Note that if s_k is the k^{th} term and s_{k+1} the $(k+1)^{th}$ term then $r = \frac{s_{k+1}}{s_k}$.

   The function adds the successive terms in the series, starting from `firstTerm`. The summation process terminates when either of the following two conditions is true: (a) the absolute difference between the values of the first k and the $(k+1)^{th}$ terms ($k > 0$) is less than `epsilon`, and (b) the number of terms added exceeds `termMax`.

   Your function must return the sum obtained, the number of terms added, and a list of the last three terms added. If the number of terms added is less than three, then the list will contain only the terms that have been added.

   10 Points

   Sample tests:

   Test 1 Function call: seriesSummer(1, 2/3.0, 0.01, 30)
   Output: (2.984585306741481, 13, [0.017341529915832606, 0.011561019943888404, 0.007707346629258935])

   Test 2 Function call: seriesSummer(1, 2/3.0,0.1,2)
   Output: : (1.6666666666666665, 2, [1, 0.6666666666666666])

   Test 3 seriesSummer(7/10.0, 1/10.0, 0.001, 30)
   Output:
   Input: (0.7777, 4, [0.06999999999999999, 0.006999999999999999, 0.0007])

End of midterm exam problem set.