10.009 The Digital World

Term 3. January 26-May 2. 2014

Midterm Exam

March 5, 2014 2:30-4:30pm Room: Cohort Classroom

Most recent update: February 28, 2014

Note:

1. This exam has two parts. First try to complete Part A and then move to Part B. You can secure a maximum of 100 points.
2. You may consult all material on your laptop and in your notes. You may also use any book(s) as a reference. **You are not allowed to (a) access any material on the Internet and (b) to consult anyone inside or outside of the classroom other than the proctors in the examination room.**
3. 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.**
4. All answers will be graded manually.

Summary:

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

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

1. Write a function `stc(s1, s2)` that takes two strings `s1` and `s2` as input arguments. The function returns the difference in the lengths of the two strings.

   Sample tests:

   Test 1

Input:	s1="SUTD", s2="NUS"
Output:	1

   Test 2

Input:	s1="SMU", s2="NTU"
Output:	0

   Test 3

Input:	s1="Japan", s2="Singapore"
Output:	4

→ *Submit:* `stc(s1, s2)` *in Tutor.*

2. Write a function `sumVal(d)` that takes a dictionary `d` as an input argument. Each key in `d` is an integer, the corresponding value is also an integer. The function returns the sum of values for keys that are less than 3. If the dictionary is empty, the function returns `None`.

   Sample tests:

   Test 1

Input:	`d`={4: -7, 8: 4, 2: 10, 1: -2 }
Output:	8

   Test 2

Input:	`d`={}
Output:	`None`

→ *Submit:* `sumVal(d)` *in Tutor.*

3. Write a function `count(a,b,c)` that takes three integers `a`, `b` and `c` as input arguments. It returns `True` if the list generated by `range(a,b)` has more elements than the list generated by `range(b,c)`.

   Sample tests:

   Test 1

Input:	a=2, b=4, c=7
Output:	False

   Test 2

Input:	a=3, b=7, c=-1
Output:	True

→ *Submit:* `count(a,b,c)` *in Tutor.*

Category B: Short programs. Total problems: 2

1. Write and submit two functions named `getMRT(f)` and `distance(d,s)`. A `test()` function is given to you at the end of this question. You may use `test()` to test `getMRT(f)` and `distance(d,s)` before submitting them.

   (a) `getMRT(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 case below. Each line in the file contains a string `r` and a list `st`, where `r` denotes the name of an MRT line in Singapore (e.g., EW Line (Green)) and `st` is a partial list of names of stations on that line. Names of the MRT line and each station are separated by a comma (,).

   The `getMRT(f)` 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 the station names as the value corresponding to this key. (*Note*: Use the Python `strip()` function to remove the $\backslash n$ from the last element of each list in the dictionary.)

   15 Points

   Test :

   Contents of file `mrt.txt`:

   EW Line (Green),Pasir Ris,Tampines,Simei,Tanah Merah,Bedok

   Circle Line (Yellow),Dhoby Ghaut,Museum,Convention Center,Millennia

   NS Line (Red),Jurong East,Bukit Batok,Bukit Gombak,Choa Chu Kang

   NE Line (Purple),Harbour Front,Outram Park,China Town,Clark Quay,Dhoby Ghaut

   Downtown Line (Blue),Chinatown,Telok Ayer,Downtown,Bayfront,Promenade,Bugis

   ---

   Dictionary returned by `getMRT(f)`:

   {‘EW Line (Green)’: [‘Pasir Ris’, ‘Tampines’, ‘Simei’, ‘Tanah Merah’, ‘Bedok’],
   ‘Circle Line (Yellow)’: [‘Dhoby Ghaut’, ‘Museum’, ‘Convention Center’, ‘Millennia’],
   ‘NE Line (Purple)’: [‘Harbour Front’, ‘Outram Park’, ‘China Town’, ‘Clark Quay’, ‘Dhoby Ghaut’],
   ‘NS Line (Red)’: [‘Jurong East’, ‘Bukit Batok’, ‘Bukit Gombak’, ‘Choa Chu Kang’],
   ‘Downtown Line (Blue)’: [‘Chinatown’, ‘Telok Ayer’, ‘Downtown’, ‘Bayfront’, ‘Promenade’, ‘Bugis’]}

Note: The dictionary will be unordered in Idle. While grading, your dictionary will be automatically ordered before comparing it with Tutor output.

→ *Submit:* `getMRT(f)` *in Tutor.*

(b) `distance(d, s)`:

This function takes as input a dictionary `d` created by `getMRT(f)` and the names of two MRT stations in string `s`. The station names in `s` are separated by a comma.

If the two stations are on the *same* MRT line then the function returns the distance between these two stations in terms of the number of stops between them that includes the last stop at the destination station. If the stations are not on the same MRT line then the function returns -1. If the user provides only one station, then a -1 is returned. The function returns -2 if `s` is empty.

15 Points

Note: If you wish, you may use the following Python function to test `getMRT(f)` and `distance(d,s)` prior to submitting these two functions. DO NOT SUBMIT `test()`. The `test()` function is available at eDimension. File `mrt.txt` is also available on eDimension; copy it and save it on your computer to test your programs. Both files are under 3 March - 9 March.

```python
def test():
    f=open("mrt.txt", "r")
    d=getMRT(f)
    done=False
    while not done:
        s=raw_input("Two stations please:")
        dist= distance(d,s)
        if dist!=-2:
            print dist
        else:
            done=True

    print "Bye!"
    f.close()

test()  # Test getMRT(f) and distance(d).
```

Sample user interaction using `test()`:

```
>>> test()
Two stations please:Convention Center,Millennia
1
Two stations please:Pasir Ris,Bedok
4
Two stations please:Dowtown,Pasir Ris
-1
```

```
Two stations please:Bedok
-1
Two stations please:
Bye!
```

→ *Submit:* `distance(d, s)` *in Tutor.*

2. Create a class named `Matrix`. This class contains several methods as described below that you are required to write and submit.

   (a) `__init__(self, m, s, f):` Takes the following three input arguments: `m`: a list of lists that denotes an $n \times n, n \geq 1$ square matrix, `s`: a string that denotes the name of the matrix, and `f`: a format string to format each element of the matrix for printing. Each sublist in `m` denotes a row of the input matrix. The name and format strings are default parameters. The default value of matrix name is `''Matrix A''` and the default value of the format string is "%6.2f".

   5 Points

   (b) __str__(self): This method returns a string consisting of the name, dimension and the elements of the matrix in the format described in examples below. It uses the default or the user provided format string to format each element of the matrix.

   5 Points

   (c) `diag(self):` Returns `True` if all elements on the diagonal of the matrix object are non-zero, otherwise it returns `False`.

   5 Points

   (d) `upperDiag(self):` Returns `True` if all elements in the upper diagonal of the matrix object are non-zero otherwise it returns `False`.

   5 Points

   (e) `lowerDiag(self) :` Returns `True` if all elements in the lower diagonal of the matrix object are non-zero otherwise it returns `False`.

   5 Points

   (f) `triDiag(self):` Returns `True` if the matrix object is tridiagonal else it returns `False`.

   15 Points

Note: For an $n \times n$ matrix A, elements a_{ii} are on the diagonal, $a_{i,i+1}$ are on the upper diagonal, and $a_{i,i-1}$ are on the lower diagonal. A tridiagonal matrix has non-zero elements on its diagonal, upper diagonal and lower diagonal; all other elements of the matrix are 0. For example, consider the following matrices.

$$A = \begin{bmatrix} 2 & 0 & 4 \\ 1 & 5 & 6 \\ 0 & -1 & 3 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 4 & 0 & 0 \\ 3 & 8 & 2 & 0 \\ 0 & 2 & 8 & 5 \\ 0 & 0 & 4 & -3 \end{bmatrix}$$

In A the diagonal elements are 2, 5, 3; the upper diagonal elements are 0, 6; and the lower diagonal elements are 1, -1. *This matrix is not tridiagonal.* The diagonal elements of B are 1, 8, 8, -3; the upper diagonal elements are 4, 2, 5; and the lower diagonal elements are 3, 2, 4; all other elements of the matrix are 0 and hence B is a tridiagonal matrix.

You may test your code using the following tests.

Test 1:

```
>>> m1=[[1,4,0,0], [3, 4, 1, 0], [0, 2, 3, 4], [0,0,1,3]]
>>> a=Matrix(m1)
>>> a.triDiag()
True
>>> print a
Matrix A: Rows: 4 Columns: 4
  1.00  4.00  0.00  0.00
  3.00  4.00  1.00  0.00
  0.00  2.00  3.00  4.00
  0.00  0.00  1.00  3.00
```

Test 2:

```
>>> m2=[[1,0,0,0], [3, 4, 1, 0], [0, 2, 3, 4], [0,0,1,3]]
>>> a=Matrix(m2)
>>> a.lowerDiag()
True
>>> a.upperDiag()
False
```

Test 3:

```
>>>m3=[[1,4,0,0], [3, 0, 1, 0], [0, 2, 3, 4], [0,0,1,3]]
>>>a=Matrix(m3, "DW Matrix", "%6.1f")
>>> print a
DW Matrix: Rows: 4 Columns: 4
   1.0   4.0   0.0   0.0
   3.0   0.0   1.0   0.0
   0.0   2.0   3.0   4.0
   0.0   0.0   1.0   3.0
```

→ *Submit:* `Matrix` and the associated functions *in Tutor.*

End of problem set.