CSC1015F Assignment 8

Arrays

Assignment Instructions

This assignment involves constructing Python programs that manipulate lists, dictionaries and strings. You can use the following built-in functions in Python: `map( )`, `math.sqrt( )`, `split( )`.

NOTE Your solutions to this assignment will be evaluated for correctness and for the following qualities:

- Documentation
  - Use of comments at the top of your code to identify program purpose, author and date.
  - Use of comments within your code to explain each non-obvious functional unit of code.
- General style/readability
  - The use of meaningful names for variables and functions.
- Algorithmic qualities
  - Efficiency, simplicity

These criteria will be manually assessed by a tutor and commented upon. In this assignments, up to 10 marks will be deducted for deficiencies.

Question 1 [20 marks]

Write a Python program called '`right-align.py`' where the user can repeatedly enter a list of strings. Use the sentinel "DONE" to end the input list. The list of strings is then printed out right-aligned with the longest string.

(***NOTE***: *The input from the user is shown in* ***bold font***.)

Sample I/O:

```
Enter strings (end with DONE):
```

Tumelo

Marcus

Rufaro

Dominique

Amanda

Elizabeth

Panayioti

Amiera

Absolom

```
Oge
Christopher
DONE


Right-aligned list:
     Tumelo
     Marcus
     Rufaro
  Dominique
     Amanda
  Elizabeth
  Panayioti
     Amiera
    Absolom
        Oge
Christopher
```

Question 2 [30 marks]

Write a module of utility functions called `util.py` for manipulating 2-dimensional arrays of size 4x4. (These functions will be used in Question 3.)

The functions you need to write are as follows:

```
def create_grid(grid):
    """create a 4x4 array of zeroes within grid"""

def print_grid (grid):
"""print out a 4x4 grid in 5-width columns within a box"""

def check_lost (grid):
    """return True if there are no 0 values and there are no

adjacent values that are equal; otherwise False"""

def check_won (grid):
"""return True if a value>=32 is found in the grid; otherwise

False"""

def copy_grid (grid):
    """return a copy of the given grid"""
```

```
def grid_equal (grid1, grid2):
"""check if 2 grids are equal - return boolean value"""
```

Note: these functions are described using docstrings.

Use the `testutil.py` test program to test your functions. This program takes a single integer input value and runs the corresponding test on your module. This is a variant of unit testing, where test cases are written in the form of a program that tests your program. You will learn more about unit testing in future CS courses.

(***NOTE***: *The input from the user is shown in* ***bold font***.)

Sample I/O:

2

```
+--------------------+
|2          2        |
|      4          8  |
|      16         128|
|2     2    2     2  |
+--------------------+
```

Sample I/O:

7

```
True
```

Sample I/O:

17

```
4 4
16 16
2 2
64 4
64 16
64 2
```

Sample I/O:

9

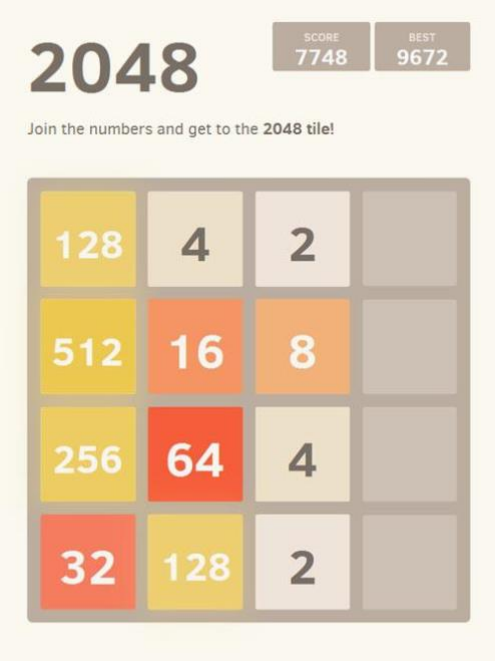
```
True
```

Sample I/O:

18

```
True
```

Question 3 [30 marks]



2048 is a puzzle game where the goal is to repeatedly merge adjacent numbers in a grid until the number 2048 is found. Your task in this question is to complete the code for a 2048 program, using the utility module (`util.py`) from Question 2 and a supplied main program (`2048.py`).

The heart of the game is the set of merging functions that merge adjacent equal values and eliminate gaps - you are required ONLY to write these functions in a module named `push.py`:

```
def push_up (grid):
    """merge grid values upwards"""

def push_down (grid):
    """merge grid values downwards"""

def push_left (grid):
    """merge grid values left"""

def push_right (grid):
    """merge grid values right"""
```

The original game can be played at: http://gabrielecirulli.github.io/2048/

Note:

- • The `check_won`() function from `util.py` assumes you have won when you reach 32 - this is simply to make testing easier.
- • The random number generator has been set to generate the same values each time for testing purposes.

(***NOTE***: *The input from the user is shown in* ***bold font***.)

Sample I/O:

```
+--------------------+
|                    |
|                    |
|          2         |
|2                   |
+--------------------+
Enter a direction:
l
+--------------------+
|                    |
|                    |
|2                   |
|2         2         |
+--------------------+
Enter a direction:
u
+--------------------+
|4         2         |
|                    |
|                    |
|     4              |
+--------------------+
Enter a direction:
d
+--------------------+
|     2              |
|                    |
|                    |
|4    4    2         |
+--------------------+
Enter a direction:
r
+--------------------+
|                2   |
|                    |
|     2              |
|          8    2    |
+--------------------+
Enter a direction:
x
```

Question Four [20 marks]

Write a program called '`histogram.py`' that takes in a list of marks (separated by spaces) and outputs a histogram representation of the marks according to the mark categories at UCT:

- fail < 50%
- 50% ≤ 3rd < 60%
- 60% ≤ lower 2nd < 70%
- 70% ≤ upper 2nd < 75%
- 75% ≤ first

From the given marks, count the number of marks that satisfy the conditions of each category. For example, for the following list of marks: 32 67 54 90 77, there is one fail, one 3rd, one lower 2nd, no upper 2nds and two firsts. Then print out horizontal bars, using "X", that correspond to the counters, along with some hard-coded axes and labels to represent a histogram.

You should use an array/list to store the counters.

NOTE: *The input from the user is shown in* ***bold font***.)

Sample I/O:

```
Enter a space-separated list of marks:
12 23 34 45 56 67 78 89 90
1 |XXX
2+|
2-|X
3 |X
F |XXXX
```

Submission

Create and submit a Zip file called '`ABCXYZ123.zip`' (where ABCXYZ123 is YOUR student number) containing `right-align.py, util.py, push.py` and `histogram.py`.

END