ECM1421 Systems Development 1
Individual Programming Exercise

Hand-out date: Tuesday 22nd January 2019
Hand-in date: Monday 11h February 2019
Feedback: Monday 4th March 2019

This assignment is worth 30% of the marks for ECM1421. This is an individual assessment. Your attention is drawn to the university's regulations on plagiarism.

Aim: To test your knowledge of programming in Python, as covered in the first part of the ECM1421 module.

Submission method: Anonymous submission via TurnItIn.com (see ELE for link).

Submission format: A PDF document with all program listings and sample output. Please ensure your Candidate Number is shown clearly at the top of your submission. You should also provide a ZIP file containing the Python programs and a copy of your PDF report with Candidate Number.

1. Reverse string (20 marks)

Write a function `print_reverse(text, print_spaces=False)` to print a string in reverse order. The argument called `print_spaces` should control whether the program should add spaces before each character. The number of spaces before each character should be equal to the position of the character in the original string. Test your function with the string "reversestring", printed without and with spaces. You MUST use a `for` loop to implement your function. Save your work in a file named `reverse.py`.

HINTS:
Strings in Python are lists of characters. That is, if `s = "hello"`, then `print(s[0])` will output `"h"`, while `print(s[-1])` will output `"o"`.
In Python strings can be multiplied. That is, `pad = "x" * 3`, will result in the variable pad being set to `"xxx"`.

Here is the expected output from the function.

```
print_reverse("reversestring")
g
n
I
r
t
s
e
s
r
e
v
e
```

```
r
print_reverse("reversestring", print spaces=True)
            g
           n
          i
         r
        t
       s
      e
     s
    r
   e
  v
 e
r
```

2. Rotate string (20 marks)

Write a function `print_rotate(text, print_direction=n)` to print a string rotated in one of eight compass directions, N-S, NE-SW, etc. Save your work in a file named `rotate.py`.

The argument called `print_direction` should control the direction of printing as in these examples.

```
print_rotate("rotate",0)
    r
    o
    t
    a
    t
    e
print_rotate("rotate",1)
     r
    o
   t
  a
 t
e
print_rotate("rotate",2)



etator

print_rotate("rotate",3)
e
 t
  a
   t
    o
     r
print_rotate("rotate",4)
```

```
    e
    t
    a
    t
    o
    r
print_rotate("rotate",5)
     e
    t
   a
  t
 o
r
print_rotate("rotate",6)




rotate




print_rotate("rotate",7)
r
 o
  t
   a
    t
     e
```

HINTS:
Printing is always from top left to bottom right. Establish which character in the string needs to be printed first and use your `print_reverse` algorithm to reverse the string when needed.

3. Anagrams (40 marks)

Two words are anagrams of each other if they contain precisely the same letters, for example orchestra and carthorse are anagrams of each other, as are damned, demand and madden. The aim of this exercise is to write a program that will find the anagrams in a list of words (`words.txt` from ELE).
Write a function `make_anagram_dict(words)` that returns a dictionary of anagrams given a list of words. Use your function to write a program `anagram.py` that uses `words.txt` to find and print the anagrams. Save your work in a file named `anagrams.py`.

HINTS:
The main idea is to construct a dictionary whose keys are the letters of a word sorted into alphabetical order and whose values are lists of all the words with that key. Thus a key-value pair might be:

```
 "acehorrst" : [ "orchestra" , "carthorse"]
```

Notice that all anagrams of a word have the same key.

You can sort a sequence with the built-in Python function `sorted()` and a quick way to join a list of characters together is the string `join()` method:

```
>>> "".join([ "H" , "e" , "l" , "l" , "o" ])
 "Hello"
```

4. Term dates (20 marks)

Write a function `term_dates` that outputs a list of dates for a Degree Apprenticeship term of 12 weeks given the year, month and day of the first week. Save your work in a file named `term_dates.py`.

You can assume a term never crosses a year boundary, but should allow for leap years every 4 years (2020, 2024, 2028, etc). You MUST NOT use any Python date or calendar library functions.

HINT:
Use a list of days in each month.

```
days = [31,28,31,30,31,30,31,31,30,31,30,31]
```

Here is the expected output from the function.

```
>>> term_dates(2019,1,7)
[[2019, 1, 7], [2019, 1, 14], [2019, 1, 21], [2019, 1, 28], [2019, 2
, 4], [2019, 2, 11], [2019, 2, 18], [2019, 2, 25], [2019, 3, 4], [20
19, 3, 11], [2019, 3, 18], [2019, 3, 25]]
```

Marking criteria

Please attempt all questions, as partial solutions will be marked. The aim of the coursework is to test your knowledge of the material in the module, not your ability to find, or create, complicated or clever programs.

Work will be marked against the following criteria. The criteria have approximately equal weight.

- Does your algorithm correctly solve the problem?
  In most of these exercises the algorithm has been described in the question, but not always in complete detail and some decisions are left to you.
- Does your program correctly implement the algorithm?
  Check with suitable inputs and ensure the results are correct.
- Is the code syntactically correct?
  Ensure there are no Python language errors, regardless of whether the solution is correct.
- Is the program well laid out and commented?
  Functions and variables should be sensibly named. Functions should each have a docstring and comments used to describe the purpose of loops and other significant sections. Python insists that you use indentation, you should also use spaces and blank lines to improve readability.