CM1103 Lab Worksheet Week 7

Recursion *[Think Python! 5.8–5.10]*

1. The following code defines a recursive function that counts from an initial value to 100.

```
def counting(i):
    print(i)
    i = i + 1
    if i == 100: return
    counting(i)
```

   (a) Enter the code into Python, call the function with an initial value of 1 and check the results are what you would expect. Program crashes because it exceeded Python's maximum depth of recursive calls

   (b) Remove the if statement from the code. What happens now?

   (c) Modify the code given in Question 1 so that it counts in reverse from 100.

   (d) Further modify that code so that it outputs the following sequence of numbers

$$2, 4, 8, 16, 32, 64, \ldots, 1024$$

2. In the lectures, we showed that the number of function calls involved in calculating the nth Fibonacci number by using recursion was given by the equations:

$$g(1) = 1$$
$$g(2) = 1$$
$$g(n) = g(n-1) + g(n-2) + 1$$

   Write a **recursive** function to calculate these values.

Calling functions *[Think Python! 10.2, 10.11, 10.12]*

3. Consider the following function to print items of a list:

```
def printList(a):
    while len(a) > 0:
        print(a.pop(0))
```

   What happens if you create a list and try to print it twice using this function? Can you explain this? Read Sections 10.2, 10.11 and 10.12 of Think Python, then following the advice in 10.13 (3), pass the function a copy of the list instead, and see what behaviour results.

CSV files

Comma separated values files are an easy and portable way to work with tabular data stored in plain text, for example, spreadsheets or databases. Each line of the file denotes a single record, with each record consisting

of a number of fields separated by commas. The `csv` module contains classes and functions to read and write csv files in various formats.

The code below shows the basic functionality.

```
import csv
with open('facup.csv') as csvfile:
    rdr = csv.reader(csvfile)
    for row in rdr:
        print(row[0] + "␣last␣won␣in␣" + row[1])
```

4. Download the file `facup.csv`. Verify that the code above works.
   (a) Modify the code so that it also prints out the `type` of the year field. What are the implications when reading in numerical data?
   (b) Modify the code so it prints `True` if the year was an even number, and `False` otherwise.
5. Download the file `MultipleTourWinners.csv` which contains information about all riders that have won the Tour de France more than once. Write code that reads in the file, and prints out the number of wins for each French rider (e.g. `Bernard Hinault,5`).

Iteration *[Think Python! 12.5]*

6. Enter the following code and see what it does:

```
for n in range(1,100):
   print(n)
```

   (a) Modify the code so it counts from 100 to 200.
   (b) The range function has three parameters **range (start, stop, step)** , where `step` is the size of increase in each iteration. Using this information modify the above code so that it lists all multiples of 2 between 0 and 100 (inclusive).

The `enumerate` function allows Python to elegantly loop through a list accessing both items and their indices. Each call to `enumerate` yields a `tuple` of two values – the first is a count of the iterated items (starting from 0), and the second is the item itself. For example, run the following code.

```
allSeasons = ["Spring", "Summer", "Autumn", "Winter"]
for i, season in enumerate(allSeasons):
   print(i, season)
```

7. Write code that reads in the file `MultipleTourWinners.csv`, and using the `enumerate` function, prints out the row numbers corresponding to riders with 3 or more wins.

Debugging *[Think Python! Appendix A]*

8. Try to find the errors in the following piece of code:

```
mark = input("Enter␣mark:␣")
if mark <= 50 and mark > 60:
   print("Result␣is␣2:2")

# Print the first 10 square numbers
for n in range (11,1):
   print(n * n)

def f(n):
   if n = 1 or n == 2:
        return 1
   else
      return f(n-1) + f(n-2)
```

Are the errors syntax errors (meaning the code will not execute), runtime errors (something goes wrong when the program runs) or semantic errors (program runs, but incorrectly due to error in logic)?

Formative assessment

The question in this section will help to prepare you for the coursework that will be set in week 9.

Download the file `Week7Template.py`, and complete the function as indicated below. Try to ensure your code passes all of the doctests.

9. Suppose we have a list of tuples representing student names and their marks, e.g.

```
a = [('Jones', 54), ('Anna Smith', 56), ('Barry Thomas', 88)]
```

Complete the function `sortStudents(a)` so that it returns a copy of the list sorted by their marks, from highest to lowest. Students with equal marks should be ordered alphabetically by their surname, followed by their forename.

You may assume (i.e. without any error checking) that each name is provided as a single forename followed by a space and a single surname (e.g. `Anna Smith`). You do not need to consider upper or lower case.

If you can, you should accomplish this using a lambda expression, so that the function consists of only 2 lines (1 for the definition and 1 for the body).

Most of the doctests provided consider integer marks, however the final test case uses a grading system from A to E, where A is the highest mark. Your function should handle both cases without additional parameters and still within 2 lines. It may be useful to look at:

`https://docs.python.org/3/reference/expressions.html#conditional-expressions`

and

`http://stackoverflow.com/questions/3501382/checking-whether-a-variable-is-an-integer-or-not`

Advanced

10. You should know that the formula to compute a^n is $a^n = a * a * a * a ... * a$. However, an alternative method to compute this is: $a^n = a^{\frac{n}{2}} * a^{\frac{n}{2}}$. I.e. $a^8 = a^4 * a^4$. This example can be further broken down:

$$a^8 = a^4 * a^4$$
$$a^4 = a^2 * a^2$$
$$a^2 = a^1 * a^1$$
$$a^1 = a$$

This can be expressed as a recursive function in Python. Using the code below as a starting point write a Python function to calculate a^n where n is even.

```
def recPower(a,n):
    # if n is 1 then return a
    # recursively call this function for n/2 and call it factor
```

```
    # if n/2 is even return the square of factor
    # if n/2 is odd then return the square of factor multiplied by a
```

Challenge: ***Doughnut party***

I'm planning a party to share my love of doughnuts, and have booked a venue that can hold between 1 and 100 people. Every guest will be given exactly one doughnut when they arrive, whether they want one or not. However, I'm very careful with my money, so I want to make sure I don't have any doughnuts left over. My local bakery sells doughnuts in boxes of 6, 9 or 20. Assuming everyone I invite will definitely turn up, which numbers of guests must I avoid inviting? [Hint: if there are 5 or less people, even if I bought the smallest box, there would be some waste. For another hint, see the section *Doughnut party* of *Lab Worksheet Week 7*.]

Email AllenSM@cardiff.ac.uk with your answer and any code/maths you used to solve the problem, with the subject line "CM1103 week 7 challenge" by midnight on Wednesday in Week 8 – a winner will be picked at random from the correct answers **but** you'll get one extra entry for particularly noteworthy or interesting code or maths in your answer.