FIT1045 Introduction to Algorithms and Programming – Workshop 9

Objectives

The **objectives of this workshop** are:

- To implement recursive algorithms.
- To explore recursive sorting algorithms.

Useful Links:

For this workshop, you may find it useful to review some of the following concepts:

- Function decomposition (Lecture 11)
- Divide and conquer (Lecture 13)
- Recursion (Lecture 14)

Task 0 (To be completed before class):

Write a recursive function `printToN` that takes as input a positive integer *n* and prints the numbers 1 to n.
Once you have done this, create a similar function `printSquares` so that it now prints the first n square numbers.
Finally, create a recursive function `sumOfSquares` that returns (not prints) the *sum* of the first n square numbers.

Note: You should not be using any loops for this task. This task should be solved using recursion only.

Output of printToN (to the console) on input n=4:

```
1
2
3
4
```

Output of printSquares (to the console) on input n=4:

```
1
4
9
16
```

Output of sumOfSquares (as a return value) on input n=4:

```
30
```

Note: A Python tutorial on recursion is available at `http://www.python-course.eu/recursive_functions.php`.

Task 1:

Write a recursive function that takes as input a positive integer *n* and returns a list *L* that contains the factorial values, 1!, 2!, . . . , n!.

For example: If the input is n=5, then the output should be:

```
[1, 2, 6, 24, 120]
```

Task 2:

Modify your recursive function in Task 1 so that given a positive integer *n* as input, it prints (via recursive calls) a line of *k* factorial stars for each integer k in the range 1 to n, and also (as before) returns a list *L* that contains the factorial values, 1!, 2!, . . . , n!. Define a function `printStars` to use within your main function. It should accept an integer k as input and print k stars on a single line.

For example: If the input is n=4, then the following should be printed to the console:

```
*
**
******
************************
```

And the following should be output from the function:

```
[1, 2, 6, 24]
```

Task 3:

Task A:

The following task requires you to do (or re-use) Tasks 1–2 from Workshop 8.

Write a Python program that takes as input from the user the name of a file containing postcode/location information. Each line in the file consists of the postcode followed by a tab followed by a comma-separated list of locations that have that postcode. For example, the file Small.txt contains:

```
3015    Newport,South Kingsville,Spotswood
3016    Williamstown
3018    Altona,Seaholme
3019    Braybrook,Robinson
3021    Albanvale,Kealba,Kings Park,St Albans
```

Your program should create a list of postcode/location pairs with each pair stored in the list. It should print the list in a user-friendly, readable format including the list index of each item.

For example: your program may do the following:

```
Enter name of postcode file:  Small.txt
Index Code Town
------------
0 3015 Newport
1 3015 South Kingsville
2 3015 Spotswood
3 3016 Williamstown
4 3018 Altona
5 3018 Seaholme
6 3019 Braybrook
7 3019 Robinson
8 3021 Albanvale
9 3021 Kealba
10 3021 Kings Park
11 3021 St Albans
```

Task B:

In this task you will write a program to partition the list of postcodes/locations into two groups: those that occur before a given suburb in dictionary order, and those that occur after.

Useful Links: For this task, you may find it useful to review some of the following concepts:

- Recursive Sorts (Lecture 15)
  (Not sure why? Ask your workshop leader.)
- Some string methods - for example, split and strip. (`https://docs.python.org/3/library/stdtypes.html#string-methods` or Perkovic book, page 97–98.)

Write a function `partitionTowns` that takes as input the list of postcode/locations and a position p in the list. Call the suburb at this position in the list suburb s. The function should partition the list so that all entries before the one containing suburb s contain suburbs that are before s (with respect to dictionary ordering), and all entries after the one containing suburb s contain suburbs that occur after s. Your function should return the index that suburb s ends up in.

Once you have written `partitionTowns`, incorporate it into a program which asks the user to select an index from the postcode/location list to be the pivot, then calls the partitioning function with this pivot.

For example: your program may do the following:

```
Enter name of postcode file:  Small.txt
Index Code Town
------------
0 3015 Newport
1 3015 South Kingsville
2 3015 Spotswood
3 3016 Williamstown
4 3018 Altona
5 3018 Seaholme
6 3019 Braybrook
7 3019 Robinson
8 3021 Albanvale
9 3021 Kealba
10 3021 Kings Park
11 3021 St Albans

Select an index of the list: 6

Updated List:
Index Code Town
------------
0 3021 Albanvale
1 3018 Altona
2 3019 Braybrook
3 3016 Williamstown
4 3015 South Kingsville
5 3018 Seaholme
6 3015 Newport
7 3019 Robinson
8 3015 Spotswood
9 3021 Kealba
10 3021 Kings Park
11 3021 St Albans
```

In this case, the `partitionTowns` function is called with index 6, corresponding to Braybrook, and returns index 2, the new index of Braybrook. Before index 2 in the new list are suburbs that come before Braybrook in dictionary order, and after index 2 are suburbs that come after Braybrook in dictionary order.

Task 4:

Modify your program in Task 3 so that it prints two lists: the list from position 0 to position p-1, and the list from position p to the end of the list. Design an appropriate function to do the printing. Make sure it is thoughtfully named.

NOTE: in some cases your program will only print one list. When does this occur?

For example: your program may do the following:

```
Enter name of postcode file:  Small.txt
```

```
Index Code Town
------------
0 3015 Newport
1 3015 South Kingsville
2 3015 Spotswood
3 3016 Williamstown
4 3018 Altona
5 3018 Seaholme
6 3019 Braybrook
7 3019 Robinson
8 3021 Albanvale
9 3021 Kealba
10 3021 Kings Park
11 3021 St Albans

Select an index of the list: 9

Updated Lists:

Index Code Town
----------
0 3021 Albanvale
1 3018 Altona
2 3019 Braybrook

Index Code Town
----------
0 3021 Kealba
1 3015 South Kingsville
2 3018 Seaholme
3 3015 Spotswood
4 3019 Robinson
5 3016 Williamstown
6 3015 Newport
7 3021 Kings Park
8 3021 St Albans
```

Task 5:

You have already written code for most of the `Quicksort Algorithm`. Write a function `Quicksort` that takes the list of postcode/locations and sorts it by town.

For example: your program may do the following:

```
Enter name of postcode file:  Small.txt
Index Code Town
------------
0 3015 Newport
1 3015 South Kingsville
2 3015 Spotswood
3 3016 Williamstown
4 3018 Altona
5 3018 Seaholme
6 3019 Braybrook
7 3019 Robinson
8 3021 Albanvale
9 3021 Kealba
10 3021 Kings Park
11 3021 St Albans

Updated List:
```

```
Index Code Place
----------
0 3021 Albanvale
1 3018 Altona
2 3019 Braybrook
3 3021 Kealba
4 3021 Kings Park
5 3015 Newport
6 3019 Robinson
7 3018 Seaholme
8 3015 South Kingsville
9 3015 Spotswood
10 3021 St Albans
11 3016 Williamstown
```