FIT1045 Algorithmic Problem Solving – Workshop 8

Objectives

The **objectives of this workshop** are:

- To investigate the "divide and conquer" technique.
- To understand how to implement recursion in Python.
- To understand binary search.
- To implement an algorithm for binary search using iteration.
- To implement an algorithm for binary search using recursion.

Task 0 (Prelab): To be completed before class

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

- Using files (Lecture 4)
- Loops (Lecture 4)
- Lists (Lecture 5)
- Some string methods - for example, split and strip. (`https://docs.python.org/3/library/stdtypes.html#string-methods` or Perkovic book, page 97–98.)

Note: If you need help completing this task, you are welcome to come to a consultation prior to your workshop.

Begin by downloading the data files you will require for this workshop from Moodle. Observe the contents of scores.txt. The file consists of multiple lines, each of which contains the name of a person and their scores on a (hypothetical) test. The format of a line is as follows:

- The name of the person and their scores are separated by a tab character (in a string you write this as the \t symbol).
- The scores are separated by spaces. There are exactly four scores for each person.
- Each score is an integer between 0 and 10, stored as a text string. You will need to convert them from strings to integers once you have split them apart.

Write a Python program that reads in scores.txt and prints out the scores achieved by each person in the format shown below, using the string `format` function to help you. You should utilize the split function to access the name and scores from each line of the file. Think carefully about what to split on. You will need to use the split function exactly twice per line.

Note: The scores in the data file are out of 10, so you will need to multiply each score by 10 to get them as a percentage.

Example: your program output should appear as:

```
Name: Alan Mathison Turing
Scores: 80%, 100%, 100%, 90%
Name: Ada Lovelace
Scores: 100%, 90%, 70%, 90%
Name: Douglas Carl Engelbart
Scores: 60%, 80%, 60%, 70%
Name: John von Neumann
```

```
Scores: 60%, 50%, 100%, 80%
Name: Charles Babbage
Scores: 80%, 80%, 70%, 100%
Name: Gordon Moore
Scores: 10%, 20%, 40%, 80%
```

Task 1:

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, so that you can check that your program has performed correctly.

For example: your program may do the following:

```
Enter name of postcode file:  Small.txt
[[3015, 'Newport'], [3015, 'South Kingsville'], [3015, 'Spotswood'], [3016, 'Williamstown'],
[3018, 'Altona'], [3018, 'Seaholme'], [3019, 'Braybrook'], [3019, 'Robinson'],
[3021, 'Albanvale'], [3021, 'Kealba'], [3021, 'Kings Park'], [3021, 'St Albans']]
```

Note: You will need to use split twice: first to separate the postcode from the rest of the string and secondly to separate the rest of the strings into individual locations.

Task 2

For this task, you may find it useful to look at the following:

- Function decomposition (Lecture 11)
- Python tutorial on functions: `http://www.python-course.eu/python3_functions.php`
- Python tutorial on `print`: `http://www.python-course.eu/python3_print.php`

Modify your program in Task 1 so that it prints the list in a more readable format.

For example: your program may do the following:

```
Enter name of postcode file:  Small.txt
3015 Newport
3015 South Kingsville
3015 Spotswood
3016 Williamstown
3018 Altona
3018 Seaholme
3019 Braybrook
3019 Robinson
3021 Albanvale
3021 Kealba
3021 Kings Park
3021 St Albans
```

Task 3:

Modify your program in Task 2 so that your program sorts the list in alphabetical order of location names. You can modify your sorting algorithm from Workshop 7 to sort the list.

Testing: Print out the list to check if it is sorted correctly.

For Example: Your program may do the following:

```
Enter name of postcode file:    Small.txt
3021  Albanvale
3018  Altona
3019  Braybrook
3021  Kealba
3021  Kings Park
3015  Newport
3019  Robinson
3018  Seaholme
3015  South Kingsville
3015  Spotswood
3021  St Albans
3016  Williamstown
```

Task 4:

Task A

Write a function `binarySearch1` that takes as input a list L and a string and uses an **iterative** binary search to find an entry in the list with a suburb that matches the given string. Your function should return the index in L if the string is found or -1 if it is not found.

You may find it useful to look at the Python code for iterative Binary Search given in:

- Divide and conquer (Lecture 13)

Modify your program in Task 3 so that it asks the user for the name of a suburb and uses `binarySearch1` to find the postcode of the given suburb. If the suburb is not in the database, your program should print "Not Found", otherwise it should print the postcode.

For Example: Your program may output:

```
Enter name of postcode file:  SMALL.txt
Enter Suburb Name:  Newport
Post code is 3015
```

Task B

Write a function `binarySearch2` that takes as input a list L and a string and uses a **recursive** binary search to find an entry in the list with a suburb that matches the given string. Your function should return the index in L if the string is found or -1 if it is not found.

You may find it useful to look at:

- Recursion (Lecture 14)

Task 5

Modify your program in Task 3 so that it asks the user for the name of a suburb and uses `binarySearch2` to find the postcode of the given suburb. If the suburb is not in the database, your program should print "Not Found", otherwise it should print the postcode.

Optional: Task 6

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.
You should be able to reuse your code from tasks 1 and 2

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

- Divide and Conquer (Lecture 13)
  (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                          -\ Things smaller than Braybrooke
1 3018 Altona                             _/
2 3019 Braybrook                          Braybrooke
3 3016 Williamstown                       -\
4 3015 South Kingsville                    |
5 3018 Seaholme                            |
6 3015 Newport                             |
7 3019 Robinson                            | Things larger than Braybrooke
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 Braybrook** (index 2) in the new list are suburbs that would appear **before Braybrook in dictionary order**, and **after Braybrook** (index 2) are suburbs that would appear **after Braybrook in dictionary order**.