FIT1045 Algorithmic Problem Solving – Assignment 2 (10%).

Due: Midnight, Friday 13th May, 2016.

Objectives

The **objectives of this assignment** are:

- To demonstrate the ability to implement algorithms for sorting and searching in Python.
- To gain experience in designing an algorithm for a given problem description and implementing that algorithm in Python.
- To demonstrate the ability to decompose code into functions in Python.
- To demonstrate the ability to implement simple recursive algorithms in Python.

Submission Procedure

1. Put your name and student ID in a comment at the start of each file in your solution.
2. Save your files into a zip file called yourFirstName_yourLastName.zip
3. Submit your zip file containing your solution to Moodle.
4. Your assignment will not be accepted unless it is a readable zip file.

Important Note: Please ensure that you have read and understood the university's policies on plagiarism and collusion available at `http://www.monash.edu.au/students/policies/academic-integrity.html`. You will be required to agree to these policies when you submit your assignment.

Marks: This assignment has a total of 30 marks and contributes to 10% of your final mark. Late submission will have 5% off the total assignment marks per day (including weekends) deducted from your assignment mark. (In the case of Assignment 2, this means that a late assignment will lose 1.5 marks for each day (including weekends)). Assignments submitted 7 days after the due date will normally not be accepted.

Marking Guide:

Task 1: 10 marks

(a) Program performs correctly. - 3 marks

(b) Demonstrate the ability to use the recursive function. - 5 marks

(c) Decomposition - 1 mark

(d) Readability (including meaningful variable names and non-trivial comments on code). - 1 mark

Task 2: 20 marks

(a) Program performs correctly.

- Sorting – 5 marks
- Searching (Genre/Title/Artist) – 4 marks
- Searching (Price) – 3 marks
- Print the list in user-friendly readable format and provide description of each field – 2 marks

- Program Control Flow – 2 marks

(c) Decomposition - 3 marks

(d) Readability (including meaningful variable names and non-trivial comments on code). - 1 mark

Task 1

Overview

In this task you will write a recursive program that takes as input a non-negative integer n and prints out the first $n+1$ rows of Pascal's triangle. Your program should be decomposed into functions. The triangle should be printed as a right angle triangle.

For Example: On input $n = 0$, your program may output:

```
Number n: 0
1
```

and on input $n = 7$, your program may output:

```
Number n: 7
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
1 5 10 10 5 1
1 6 15 20 15 6 1
1 7 21 35 35 21 7 1
```

Note: The values for row i can be obtained from row $i-1$ as follows:

- The first and last entries are 1
- The entry at position j (apart from the first and last entries) in row i is the sum of the entries at position $j-1$ and j in the previous row.

Your program should use a function `printPascal`.

Function printPascal

```
Input: List L containing the previous line in Pascal's triangle
         or the empty list if there is no previous line.
         n, the number of lines that still need to be printed in the triangle
         (excluding the line printed by this call of the function).
Output: Updates list L to contain the next line in Pascal's triangle and prints this line.
```

Task 2

Overview

In this task you will write a program for an on-line CD store. Your program should create a list of records about CDs by reading the data from a file. The program should then display a menu that allows the user to **sort** the list by different attributes of the CDs or **search** for CDs based on these attributes until the user elects to **quit**.

Your program should be decomposed into functions. Some of the main functions are specified in this document, but you should use further functions when necessary.

Note: You cannot use inbuilt functions for searching or sorting the list: you must write your own functions.

Function createDatabase

```
Input: None
Output: Returns a list L containing the records in CD_Store.txt
```

Description: Function creates the database by reading from the file CD_Store.txt. Each line in the file is a comma-separated record containing the Title, Artist, Genre and Price of a CD. (Note: Title, Artist and Genre are strings (and do not contain commas), but Price is a number.) The database is stored as a list of lists.

For example: If CD_Store.txt contained:

```
Gangnam Style,Psy,K Pop,8.28
The Piano Guys Live,The Piano Guys,Classical,19.99
Live on Earth,Cat Empire,Jazz,21.99
```

your program should store store the database as the following list:

```
[['Gangnam Style', 'Psy', 'K Pop', 8.28], ['The Piano Guys Live', 'The Piano Guys',
'Classical', 19.99], ['Live on Earth', 'Cat Empire', 'Jazz', 21.99]]
```

Function DisplayMenu

```
Input: none
Output: Prints menu.
```

Description: Program displays a menu with the following options:

1. Print List of CDs
2. Sort CDs by Title
3. Sort CDs by Artist
4. Sort CDs by Genre
5. Sort CDs by Price
6. Find All CDs by Title
7. Find All CDs by Artist
8. Find All CDs by Genre
9. Find All CDs with Price at Most X.
10. Quit

Function PrintList

```
Input: List of CDs
Output: Prints list in readable format.
```

Description: Program prints the list of CDs in a format that is readable for users and includes a identifies what each field is (Hint: you have written similar functions in Workshops.)

Function SortByTitle

```
Input: List of CDs
Output: Updates the list of CDs so that elements are sorted in ascending order by title.
```

Description: Program sorts the list of CDs by the title attribute.

Function SortByGenre

```
Input: List of CDs
Output: Updates list of CDs so that elements are sorted in ascending order by genre.
```

Description: Program sorts the list of CDs by the genre attribute.

Function SortByArtist

```
Input: List of CDs
Output: Updates the list of CDs so that elements are sorted in ascending order by artist.
```

Description: Program sorts the list CDs by the artist attribute.

Function SortByPrice

```
Input: List of CDs
Output: Updates the list of CDs so that elements are sorted in ascending order by price.
```

Description: Program sorts the list of CDs by the price attribute.

Function FindByTitle

```
Input:  a target string and a list of CDs
Output: Prints all CDs in the list of CDs that have the title target.
```

Description: Program should print all elements in the list of CDs that have a title that matches target.

Function FindByGenre

```
Input:  a target string and a list of CDs
Output: Prints all CDs in the list of CDs that have the genre target.
```

Description: Program should print all elements in the list of CDs that have the genre given in target.

Function FindByArtist

```
Input: a target string and a list of CDs
Output: Prints all CDs in the list of CDs that have target listed as the artist.
```

Description: Program should print all elements in the list of CDs that have the artist that matches target.

Function FindByPrice

```
Input: the price (a decimal number) and a list of CDs
Output: Prints all CDs in the list of CDs that cost at most the given price.
```

Description: Program finds all CDs that cost at most the amount specified by price.

Testing

Remember to test your program is working correctly. For example, you can print the list after it has been sorted to confirm that each sort function is working correctly.