ASSIGNMENT 2: SORTING

CS3D5A, Trinity College Dublin

Deadline: 23:00 17/11/2017
Grading: The assignment will be graded during the lab hours on 17/11/2017
Questions: You will able to ask question during the lab hours on 03/11/2017
Submission: Submit via blackboard. Include a separate .c file for each task

Goals:

- Implement some sorting algorithm
- Learn how to implement quicksort
- Learn how to evaluate the performance of a sorting algorithm
- Use sorting in a practical application

Task 1 - Quicksort

- Implement quicksort for sorting an array of integers. Your implementation should count the number of times values are swapped in the array and the number of times two values in the array are compared. These total number of swaps and comparisons should be reported at the end of your program's execution.
- Test your implementation on the following types of data:
    - A randomly shuffled array with no duplicate values e.g. [4, 3, 5, 1, 0, 2]
    - A randomly shuffled array with duplicate values e.g. [3, 3, 2, 1, 1, 4]
    - An ascending sorted array e.g. [0, 1, 2, 3, 4, 5]
    - A descending sorted array [5, 4, 3, 2, 1, 0]
    - An array where every value is the same (uniform) e.g. [3, 3, 3, 3, 3, 3]
- What do you observe about the performance of quicksort under each of these conditions? You can evaluate performance by looking at how many comparisons and how many swaps take place. How does that relate to what we covered in class?

Task 1 mark allocations	
Correct implementation of quicksort using any pivot selection/partitioning method, demonstrated on an array of 10 values	2 marks
Printing the number of swaps and the number of comparisons Quicksort performs when sorting an arbitrary array	2 marks
Demonstration of Quicksort's performance on each of the 5 types of data given above (random no duplicates, random with duplicates, ascending sorted, descending sorted and uniform). Demonstrate on an array of 10, 000 values. Comment on your results	1 mark

```
// Sample Output
Running profile tests with 10000 elements

TEST  : Unique random values
SORTED: Y
SWAPS : 30968
COMPS : 221706

TEST  : Random values
SORTED: Y
SWAPS : 32231
COMPS : 223552

TEST  : Ascending sorted list
SORTED: Y
SWAPS : 0
COMPS : 131343

TEST  : Descending sorted list
SORTED: Y
SWAPS : 5000
COMPS : 136359

TEST  : Uniform list
SORTED: Y
SWAPS : 61728
COMPS : 209757
```

Task 2 – Another sorting algorithm

- Implement a sorting algorithm of your choice and evaluate it in the same manner as you did quicksort (i.e. print swaps/comparisons and test for each of the 5 different types of data). Does the algorithm you picked perform better or worse under each of the five conditions?

Task 2 mark allocations	
Implementation of a sorting algorithm of your choice. The algorithm you choose should be useful from a computer science perspective e.g. Gnome sort was created to illustrate poor algorithm design and Bogosort is a joke (literally). Implementing these will not get you any marks.	2 marks
Demonstrate performance of implemented algorithm compared to quicksort	1 mark

Task 3 – Most popular games

- You have been provided with a dataset of game reviews which have been gathered from IGN over the last 20 years. Sort the reviews on the basis of game scores and find out what the most popular games of the last 20 years are.

  (You many need to make use of the atoi function in order to convert the scores from strings to ints. You can see examples of this in the Pokemon solution from Lab 1.)

- How would you get the top 5 games for each of the last 20 years (i.e. top ranked games for 2012, top ranked games for 2011 etc.)? (you might want to remember what we discussed about combining sorts!)

Task 3 mark allocations	
Load and sort IGN reviews and print top 10 most popular games of the last 20 years	1.5 marks
Discuss how you would get the top 5 games for each of the last 20 years	0.5 mark