University of Nottingham
UK | CHINA | MALAYSIA

# CELEN086 Introduction to Algorithms

Semester One - Coursework Assignment

## Instructions: (*Please read carefully before getting started*)

1. This coursework contributes to **25%** of your overall CELEN086 module marks.
2. This coursework consists of THREE questions and maximum marks 50.
3. The deadline for submission of the completed coursework assignment is:
   **5 pm (China Time),Friday 6 December 2024**.
4. Penalties for late submission will be applied in accordance with university regulations.
5. No excuses such as problems with internet connectivity, etc. will be entertained; so you are advised to submit your work well in advance before the deadline.
6. This work must be completed on your own. **Plagiarism and collusion are regarded as very serious academic offences and will be treated as such.**
7. You must submit **ONLY ONE PDF file** consisting all of your work with the signed Coursework Submission Form on the front.
8. You must type your answers neatly using a text editor (e.g. MS Word). The style of algorithms you write must be similar to what you have seen in lectures and seminars. You must include all details of any sub-algorithms (or helper functions) used in your work.
   **Hand written submissions will not be accepted.**
9. Save your PDF file in the following format: **StudentID_N086CW.pdf**
   For example, `207XXXXX_N086CW`
10. Submit your final PDF file electronically to the Submission Box available on Moodle.
11. Note that your work will not be returned so you should keep a copy for reference.

**Question 1** (15 marks) **Removal of dull items from museum**

The management team of a well-known museum wants to get rid of some exhibition items that are classed as ‘dull’. Miriam, the senior manager, comes up with a plan to remove the most boring exhibition items. She gives each item a rating, and then removes the one with the lowest rating. She wishes to automate this rating procedure by designing applicable algorithms.

(a) Write a recursive algorithm **`"RemoveSmallest"`** that accepts a list of integers (which are ratings of items), removes the least rated item, and returns the list of remaining numbers (i.e. ratings).

  * If there are multiple elements with the same value, remove the one with a lower index.
  * If the input list empty return an empty list.
  * Do not alter the order of remaining elements.

For example,

- `RemoveSmallest([5, 3, 2, 1, 4])` returns [5, 3, 2, 4],
- `RemoveSmallest([2, 2, 1, 2, 1])` returns [2, 2, 2, 1].

(b) Demonstrate (Trace) how your algorithm works by using the list [5, 2, 3, 2, 4].

**Question 2** (15 marks) **Non-linear data structure**

We often need to determine whether a particular number exists in a given list of numbers. One way of course is to simply start at the beginning of the list and check every data point until we find what we’re looking for or exhaust the list.This is fine if we only need to search the list once.

The method above is very inefficient if we need to search the list many times for different numbers. In this case, a better method is to turn the list into a Binary Search Tree which can be searched much more quickly than a list.

(a) Given the list [2, 15, 7, 8, 23, 10, 12, 37, 4, 17, 18, 20, 9, 25, 14]

  (i) Turn this list into a binary search tree by inserting elements one by one into an empty tree. Carefully show the process of building this tree step by step.

  (ii) Highlight the path taken through the tree when searching for the number 18.

(b) Write a recursive algorithm `"SortedListToBST"` that accepts a sorted list and converts it into a balanced Binary Search Tree.

## **Question 3** (20 marks) **Perfect Square**

Perfect square, is a number made by squaring an integer.

Perfect squares have practical uses in real world scenarios, such as determining areas in construction, calculating distances in physics, and mathematical modeling.

(a) You might know some pretty large perfect squares. But what about the NEXT one? Write an algorithm **"FindNextSquare"** that gives the next perfect square number without using the square root function. If the input parameter itself is not a perfect square, then $-1$ should be returned. You may assume the input parameter is a positive integer.

For example,

- `FindNextSquare(`$9$`)` returns $16$ (since $16$ is the next perfect square after $9$).
- `FindNextSquare(`$8$`)` returns $-1$ (since $8$ is not a perfect square).

(b) Write an algorithm **"FindSum"** to find the sum of all perfect squares that are less than $n$, where $n$ is an input positive integer.

For example,

- **`FindSum(`**$12$**`)`** returns $14$ (because $1 + 4 + 9 = 14$),
- **`FindSum(`**$25$**`)`** returns $30$ (because $1 + 4 + 9 + 16 = 30$).

(c) Write an algorithm **"NoSquareList"** that accepts a positive integer $n$ and returns a list consisting of all integers less than $n$ that are not perfect squares.

For example,

- **`NoSquareList(`**$16$**`)`** returns the list $[2, 3, 5, 6, 7, 8, 10, 11, 12, 13, 14, 15]$.