Software Engineering II: Algorithms and Data Structures
Assignment 1

Aim:
The aim of the assignment is to test students' ability to operate with file I/O, dynamically allocate memory, use a **list structure** to store the data, apply a sorting algorithm on this structure, insert a new element, delete an element from this structure, and finally apply some processing on this structure.

The objective of the assignment is for the student to understand the advantages and disadvantages of using a list structure compared to the use of an array structure for manipulation of the data.

Please note that you should build up your data structure from scratch (do not use the existing list container of C++). This restriction applies only for the main list structure of your program (i.e. the data structure that holds your data). You can use the available containers if you wish for reading or writing the data to the files.

Details:
You have to write a program that operates on a list. Initially, each element of the list is a structure that contains two data fields. Field 1 is a **positive** integer taking values in the range [1, 150], and Field 2 is the pointer that points to the next element of the list.

The program should read in two files. The first file (*data_x.txt, where x is an integer*) contains the data (i.e. integers) and it is in ASCII format. The second file, whose filename should be passed as a command line argument, contains a list of operations that you have to perform on the data from the first file. Details of the two files are given below.

Data file (*data_x.txt*)
This file contains the data that you have to manipulate. It is an ASCII file, the data values are integers and are stored one per line. Instances of this file are *data_1.txt*, *data_2.txt*, …

Command file
This file contains the operations that your program should perform on the data from the data file. The commands are shown in Table 1. Please note that the commands are case-sensitive.

Command	Description
r	This command reads the data file *data_x.txt*. The command 'r' is followed by a number x indicating the exact file that has to be read. After the reading operation, no ordering is imposed to the resulting structure. Note that this also defines the number of the output file as well (see below)
s	This command applies the Bubble Sort algorithm to sort the elements of the structure in ascending order.
w	This command writes all the data in the structure to a file *output_x.txt,* where x is determined in the command file through the read command that has been preceded. The data should be stored as one data per line. If the file already exists, this command should append the new data to the end of the file.
i	This command inserts a new element in the structure **without** destroying the ordering. If no ordering exists, then this command just inserts the new element in the structure.
d	This command deletes an element from the structure **without** destroying the

	ordering. In the case where the structure has multiple elements of the same value, this command deletes **only** one (the first instance). In the case where this value is not in the structure, the command should not perform any operations.
e	This command calculates the number of entries in the list that have a value more than a threshold T and stores the result to the output file. The value of T is on the next line of the file. Your program should write the following line in the output file "Number of elements with value greater than T: x", where T is the provided threshold and x is the calculated number.
m	This command finds the maximum value the data in the list and stores it to the output file. Your program should write the following line in the output file "Maximum value: x", where x is the maximum value of the data in the list.
a	This command interpolates the linked list by adding an extra node between every two adjacent nodes of the list with a value equal to the average value of the data of the two nodes (the result should be converted to integer). For a given input list of with *N* elements, the resulted interpolated list will have *2N-1* elements. If the original list has no elements or a single element, then no action should be taken.

Table 1. The list of possible commands

An instance of the command file is given below:

```
r
1
s
i
45
i
50
d
50
w
m
```

The commands should be interpreted as follows:

1. Read file *data_1.txt* in the internal structure
2. Sort the elements in ascending order
3. Insert entry <45>
4. Insert entry <50>
5. Delete entry <50>
6. Write the current data of the structure to file *output_1.txt*
7. Calculate the maximum value of the data and write the result to file output_1.txt

The *data_x.txt* and *output_x.txt* files should have one entry per line. An example of an output file it is shown below, which is the output file that corresponds to the above example. It is assumed that the input file had one entry only (<10>):

10
45
Maximum value: 45

Valid Assumptions:

- You can assume that the input and output files are in the same directory as you executable.
- You can assume that the reading command will be performed only once in each execution of your program, and it will be always the first command.
- You can assume the files are properly formatted e.g. an integer follows the r command, as in the command file example given above.

Development issues:

1. You can develop your program in a C++ environment of your choice. You should ensure that your program can be executed taking the name of the command file as a command line argument. For instance on the Linux terminal:

   $ ./assignment1 commandFile.txt

   ,where assignment1 is the name of your executable and commandFile.txt is the name of the command file.

2. A set of input and anticipated output files are given for you to test your code. Please note that these tests are not exhaustive and during marking a larger and more comprehensive test set will be used.
3. Your program has to be compatible with **Linux**, as the markers will use the above environment to test your code.
4. Note that some compilers are stricter than other leading to developing of code that works fine under one system but fails in another. Make sure that your code is not prone to such issues.

Marking scheme:

Your program will be marked using the following criteria:

1. **Correct** and **efficient** operation for
    1. reading data from the file (10%)
    2. sorting the data (10%)
    3. inserting new data (10%)
    4. deleting an entry (10%)
    5. calculate the number of elements with value > T (10%)
    6. find the maximum value (10%)
    7. find the average value (10%)
    8. writing the data to the file (10%)
2. Clear and understandable code, use of functions (e.g. comments, proper name selection) (20%)

3. Assignment 1 has a 50% weight, where Assignment 2 has a 50% weight of the overall mark for the coursework.

Deliverables:
You should submit via Blackboard the source code, including your real, email and login name in a comment at the start of the file.

Submission deadline: 28th February at 23:00
If you delay the submission, the usual penalty will be applied
(http://www3.imperial.ac.uk/electricalengineering/teaching/undergraduate/assessment/guidance)