CT102 Algorithms	Assignment 1

Date: Friday 16th February 2024

Due: on or before Monday 4th March* 2024 via Canvas – SINGLE document including Plagiarism

Total Marks: 15

*unless you have a LENS report

***** Please include the following plagiarism declaration in your solution if applicable:

Plagiarism Declaration:

"I am aware of what plagiarism is and include this here to confirm that this work is my own"

Please note that any suspected cases of plagiarism or use of generative AI tools, or absence of a plagiarism declaration, will not receive a mark until assurances can be given as to the origins of the solution.

QUESTION 1 (4 MARKS)

Given the following function `check()`which is passed an integer array, `arrA[]`, and its associated size, `size`, and returns a Boolean (line numbers are included):

```
L1   bool check (int arrA[], int size)
L2   {
L3        bool checked = true;
L4        for (int i = 0; i < size - 1 && checked; i++) {
L5            if (arrA[i] > arrA[i + 1]) {
L6                checked = false;
L7            }
L8        }
L9        return(checked);
L10  }
```

(i) Explain, in your own words, and with reference to the appropriate line numbers, and some sample data, what the function `check()` does and how it works. *(1 mark)*

(ii) Perform a time step analysis of the function `check()` as a function of the size of `arrA[]`, in a best case and a worst case situation. Clearly explain any assumptions made. *(3 marks)*

QUESTION 2 (3 MARKS)

Given the following function `search()` which is passed the integer arrays, `arrA[]`, with non-unique, unsorted values of size, `size`, and an empty integer array, `arrOccur[]`, the same size as `arrA[]`. Also passed to the function is an integer value `item`. (Note line numbers are also included):

```
L1   int search(int arrA[], int size, int item, int arrOccur[])
L2   {
L3       int location = -1;
L4       for (int i = 0; i < size; i++) {
L5           if ( arrA[i] == item ) {
L6               ++location;
L7               arrOccur[location] = i;
L8           }
L9       }
L10      return (location);
L11  }
```

(i) Explain, in your own words, and with reference to the appropriate line numbers, and some sample data, what the function `search()` does and how it works. *(1 mark)*

(ii) Perform a time step analysis of the function `search()` as a function of the size of `arrA[]` in a worst case situation. Clearly explain any assumptions made. *(2 marks)*

QUESTION 3 (8 MARKS)

Given two test files, file1.txt and file2.txt, (both with 10,000 integers), perform a comparison of the four sorting techniques considered for different values of N (array size) in terms of:

a. time taken
b. number of swaps/data moves
c. number of comparisons

Present your results in a meaningful and clear way so that it is easy to see differences between the algorithms for the same value of N. For example you might want to use a graph or to use a table as follows:

N (array size)	Bubble Sort	Selection Sort	Insertion Sort	Count Sort
1000	0.001000	0.001000	0.001000	0.0000
2000	0.008000	0.004000	0.002000	0.0000
3000	0.012000	0.007000	0.004000	0.0000
etc.				

Note the following:

- You should read in different portions of the files for each test of N, for example the first 1000 integers, then the first 2000 integers, etc. The last test should use all, or nearly all, of the 10,000 integers. Make sure not to read in partially sorted data (i.e. from a previous run) to ensure fair comparisons.
- Modify `countSort()` to deal with negative integers and note that for different portions of N there will probably be a different maximum value (and potentially a different minimum value) so you will need to find this for each call, which you may decide to include as part of your analysis, i.e. the time to find the minimum and maximum. You may use Radix sort if you wish but both Radix and Count sort are not required.
- Ensure all functions have the correct code to count time, data movements and swaps (if used). Please note that "data movement, swaps, comparisons" should only refer to the data in the array we are sorting – no other comparisons or data assignments should be counted (e.g. do not count "`i < size`" as a comparison).
- Modify `main()` to read in different sizes of data and call the appropriate functions. You may want to write your results to a file (e.g., *algorithm, N, time taken, numSwaps, numComprs*) for each function call with each N.
- Note that countSort() may not (always/ever) return non-zero time.
- Note that you can decide:
  - what steps of N to use, but you should aim for at least 5 values of N.
  - what initial value of N to use, for example N>=1000 as a starting value or not.
  - how best to organise the flow of your tests in main()

In your answer:

- Explain your testing approach (values of N, what is counted and where, etc.)
- Present the results in a clear and tidy way (comparisons) and comment on/explain the trends that you see. Also, include some sample output (screenshots) to prove the results were obtained from your code.
- Include your `main()` code (screenshot) or wherever you have written the code to read in the data and call the appropriate functions. Ensure it can be easily followed/understood (include comments as needed).
- Include the `countSort()` function with the modifications for integer values.
- Include any of the sorting code if you have made changes to the code.