CSCI 340 Computer Assignment 1 Spring 2019
(10 points)

Search Algorithms

For this computer assignment, you are to write and implement a C++ program to implement two search algorithms (*linear search* and *binary search*) on randomly generated integers stored in vectors.

The program is partially implemented. You can obtain the source file `assignment1.cc` at `/home/turing/mhou/public/csci340spring2019.`

In this file, several routines are already provided for you. They include the `main()` routine, the `average_comparisons()` routine and the `random_number()` routine. You are required to implement the following routines in your program:

- `int linear_search(const vector<int>& inputVec, const int x, int& comparisons)` : A linear search algorithm, where `x` is the searched item in vector `inputVec.` It simply starts searching for `x` from the beginning of vector to the end, but it stops searching when there is a match. If the search is successful, it returns the position of the found element; otherwise, it returns `-1`. You need to save the number of comparisons (between `x` and a vector element) conducted in this search in the parameter `comparisons`.

- `int binary_search ( const vector < int >& inputVec, const int x, int& comparisons )` : A binary search algorithm, where `x` is the searched item in vector `inputVec`. If the search is successful, it returns the position of the found element; otherwise, it returns `-1`. The same as above, you need to save the number of comparisons in the parameter `comparisons`. **Note that only equivalence comparisons are counted.**

- `void print_vec ( const vector < int >& vec )` : This routine displays the contents of vector `vec` on standard output, printing exactly `NO_ITEMS = 8` numbers on a single line, except perhaps the last line. The sorted numbers need to be properly aligned on the output. For each printed number, allocate `ITEM_W = 4` spaces on standard output.

Programming Notes:

- Do not change existing implementation in `assignment1.cc`. But you need to include any necessary headers and add necessary global constants.

- You are not allowed to use any I/O functions from the C library, such as `scanf` or `printf.` Instead, use the I/O functions from the C++ library, such as `cin` or `cout.`

- To compile the source file, execute "`g++ -Wall assignment1.cc –o assignment1.exe`". This will create the executable file `assignment1.exe`. To test your program, execute "./`assignment1.exe &> assignment1.out`", which will put the output (including any error messages) in file `assignment1.out`. Depending on the implementation of binary search, there may be correct outputs of slight difference. You can find the correct output files `assignment1.out1` and `assignment1.out2` in the directory shown in the last page.

- Add documentation to your source file.

- Prepare your Makefile so that the TA only needs to invoke the command "`make`" to compile your source file and produce the executable file.

- When your program is ready, submit your source file and Makefile to your TA by following the Assignment Submission Instructions.