# Test bed for search and / or sorting algorithms

In this project a menu driven test bed program for searching and / or sorting algorithms is made. The algorithms are from teaching sessions. The points you get from this project depends on how much work you do for it. The more functions you include in the program the more points you get. Don’t remove any working functionalities when you made a new function; all functions must work (already made and new ones) at any point of time.

You get 0 – 5 points out of this project.

We will use some time in the last teaching sessions to give support for this project. We will also combine making home exercises and this project in the end of this course. Basically you can utilize some code from home exercises in this project.

Program features are divided into five groups; groups 1 and 2 involve searching algorithms and groups 3 – 5 involve sorting algorithms. If you make search functions do first function 1 and then function 2 (there are dependencies between these two parts). If you make sorting functions do them in the order 3, 4 and 5. There are no dependencies between searching and sorting parts of the project; you can do solely groups 1 and 2 and solely groups 3 – 5 or all of them.

|  |  |
| --- | --- |
| Group number | Properties / functions to be made for the group |
| 1 | The program makes a sequential\_search from the data given to it based on some key value. Result of the search is output to the screen.  More details can be found below. |
| 2 | The program makes a binary\_search from the data given to it based on some key value. Any binary\_search variant handled in the lessons can be used. Result of the search is output to the screen. After the search some performance data is output also to the screen.  More details can be found below. |
| 3 | The program makes insertion sort to the randomized data.  More details can be found below. |
| 4 | The program makes sorting of performance to the randomized data. Merge sort and quicksort are this kind of algorithms; you just pick one candidate.  More details can be found below. |
| 5 | Add to your program one sorting algorithm of class and one sorting algorithm of class (performance) not added before. In addition to output sorting results also a comparison must be output of performance of all used sorting algorithms.  More details can be found below. |

## Grading

The basic rule is that you implement the functions in this order 1 -> 2 -> 3 -> 4 -> 5. Basically the number of steps you do in this order is your grade. However you can make the searching activities only (1 -> 2) and sorting actitivies only (3 -> 4 -> 5) and here again the number of steps yoy make is your grade. Of course you can stop doing the steps in the middle, for example 1 -> 2 -> 3 and then you get 3.

Below are the steps and some possible routes (which functions he/she does) the student can do. The path arrow shows the order of things done and the balls the array goes through show the steps done. The numbers tell which grade the student gets by following that path.



## More details on the requirements of Group Number 1

In this alternative, the data to be searched is generated as shown on page 9 of the lecture slide tr7.pdf; the user only enters the data size here. For example, if 10 is entered as the data size, then the following data is generated: 1, 3, 5,…, 19. The data to be searched contains only integers both at this function and at other functions of the application of the search algorithms. Similarly, in sorting, we only sort integers.

As a result of the search, the program prints the success information of it and if the key value to be searched was found (= success) then in the list its location is also printed.

With regard to entering the key value to be searched, there are two possibilities: the user enters it directly; for example, 11, or the key value to be searched for is generated as a random number as described on page 9 in the slide set tr7.pdf. You can choose either way or if you are interested also doing both of them is ok.

## More details on the requirements of Group Number 2

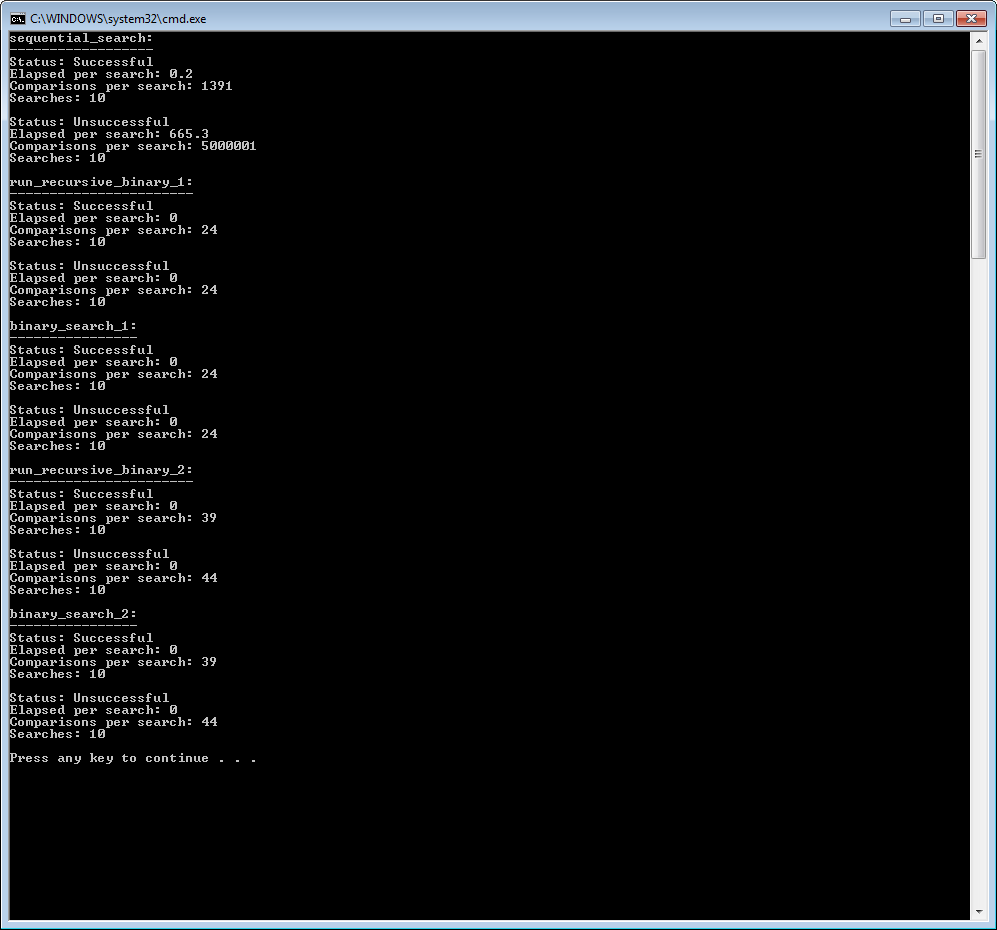
In binary search, the data to be searched and the key value to be searched are generated in the same way as presented in previous section (1.1). Similarly, the presentation of the results is done as described in Chapter 1.1.

If the user selects the performance comparison function for this function group, the sequential search and binary search algorithms are run on the exact same data with the same search keys. During the search, the number of comparisons as well as the elapsed wall clock time are observed; these are both printed for both search algorithms. In the performance comparison, both the data to be searched and the key to be searched are randomly generated following the instructions on page 9 of the tr7.pdf slide series.

In order for the results of the performance comparison to be more reliable, both search algorithms must be called many times (e.g. 10 times) with different search keys during the test run in order to make the results more reliable. Otherwise, for example in a binary search, the value found in the middle of the data may be generated as a random key, and the search ends immediately and the results are distorted. The user can also enter the number of repetitions.

In addition, a performance comparison is even more reliable if about half of the search ends in success and half does not. When we generate a search key at random this requirement is met enough. Look more instructions on slideset tr7.pdf!

Below is a suggestion for output of a performance run - here all the search algorithm variants that have been covered in the lessons have been run through; all you have to do is to run only sequence search and some binary search variant; not all binary search variants:



## More on the requirements of Group Number 3

Here, user enters the size of the data to be sorted with insertion sort; for example, input 1000 here means that the program generates 1000 random integer values in some array that will be sorted. In the example situation, the limits for generating each random number could be from 0 to 10,000, so that the generation of the same numbers in the data would be reasonably rare. Thus, the possible value range of each individual number to be generated is ten times the number of numbers to be generated.

The program prints out some small number of items in the beginning of a table to be sorted before sorting; e.g., 200 numbers from the beginning, and the same area of the table is printed after sorting to ensure that the algorithm works correctly for at least this slice area of the sorted array. The size of the area is asked from the user; if desired, the entire material is also printed before and after sorting, thus ensuring the success of the entire sorting.

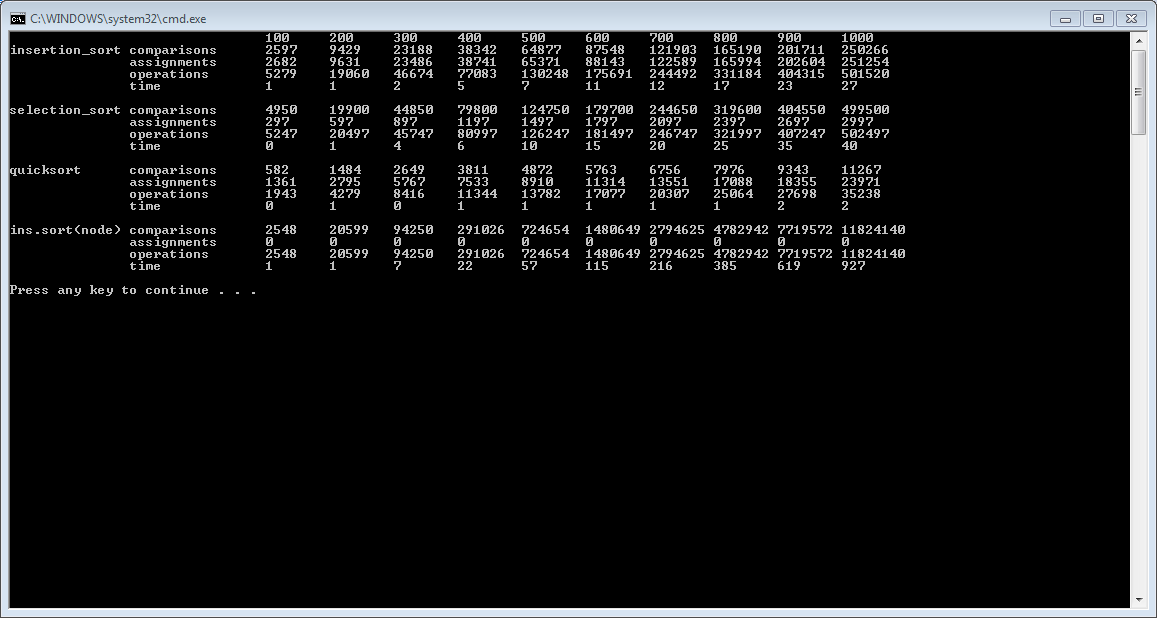
## More details on the requirements of Group Number 4

For inputs and outputs, the same requirements apply here as in section 1.3. for the new implemented sorting algorithm.

## More details on the requirements of Group Number 5

Here again, the inputs and results for the new sorting algorithms are similar to those presented in Section 1.3. In addition, the user selects here which of the two added sorting algorithms is run.

The performance comparison function could aim for the following result which is printed out (just an example; you can do it better!):



, where the 1st line tells the sizes of the material to be sorted (this is how many integer values were sorted). In operations numbers, the values of comparisons and assignments have been added together. Time is told in milliseconds. Three type sorting algorithms were run in this performance test; in your project just run the ones implemented so far. Note: in order to calculate the number of assignments while running the algorithm, you must override the assignment operator of the Key class (and at the same time the Record class); in it you record the number of assignments.

For the performance run, the results should still be plotted in the following style (example only where milliseconds are inspected):



The goal here is that based on the image, you will be able to rank the sorting algorithms based on execution time or some other criterion seen on your own tests.

## Main menu user interface

The program's user interface is text-based (run in cmd window) menu-driven interface, where it is possible to select function groups 1 to 5 from the main menu (displayed as soon as the program starts). For example, after selecting option 1, the user wants to perform a sequential search and the program asks the required input information. You can design the submenus yourself but follow a clear hierarchical data inputs for the program. Number the submenus in letters: a), b),…, etc. Other inputs to the program are given through question-and-answer dialogs; e.g:

**Give the amount of the sorted items: 1000**

, where the user input is marked with a gray background color.

## Requirements for return

* The return is made as a compressed file in place indicated in moodle course page, the name of the file must include the name of the author of the program, the project number in the course and the points target the student has done for the project; e.g. TIRA\_project\_3\_point\_target\_4\_Sanna\_Marin.zip.
* Return either the entire programming project in a format understood by Microsoft Visual Studio IDE or the necessary source code files to make it a working program by simply creating an empty Windows Console project in IDE to which the source files are added and thus it can be compiled into a working program. You must make portable C++ code so that you can compile it with g++ with the command: g++ \*.cpp -o testped for example in Linux; in here all source code must be in the directory where you make the complation (also .h files!). I must be able to compile all your returns with the command: g++ \*.cpp -o testped in Linux environment for my tests.
* The README.DOCX file (note: Word document) is also included in the return file which tells:
  + Program compilation / linking / installation / run instructions so that the teacher can test the program by following these instructions.
  + Test cases run in each level (1-5) of the program to ensure the operation of the program. In practice, you document the inputs and the outputs printed by the program. A handy way to do this is to use some snipping tool, which lets you point to a window on the screen and then it’s contents goes to the clipboard. Include these screenshots from clipboard into your document and write in the document how they prove the correct operation of the program (as text below the image). You can indicate the details in the picture with arrows or the like.
  + If you performed the activities required for group number 5 above, include the performance results of the performance comparison and a graphical diagram of them in the README.DOCX file. Just copy paste areas from corresponding Excel document containing the data and diagrams to Word document.
  + Report what score (= points) do you aim for with the returned exercise work. This information must be realistic and correct; if you aim for, for example, five points, the teacher must find all functions required for that score in your returned program.
  + Returner contact information: email address and phone number for further queries if they are needed
  + Work done in hours for this project in the format of date of working, hours done in that day, what you did inside these hours. You should collect such rows, for example, in an Excel file, the contents of which you add to the README.DOCX file.
  + Briefly explain verbally the design solution you use in each function group. For example, how do you handle data generation, sorting algorithm running, results and performance indicators generation, and so on.
* Note: The source code of the program must compile with g++ compiler without error messages and warnings like told above. The user interface of the resulting program should be as described in these instructions (no different user interface is reported than the one the program actually produces!).

README.DOCX file must comply with TAMK's written work reporting guidelines (you can find this with the intra search function; a ready-made Word template has been made for this).

NB! copying the project solutions is forbidden and this will be checked! Copying is an abuse and will be dealt with the process created for the abuse treatment of TAMK.