Test bed for search and or sorting algorithms

# Introduction

## 1.Program compilation

* Visual Studio 2022 Preview
* C++ environment

## 2. Linking

https://moodle.tuni.fi/course/view.php?id=21010

## 3. Installation

* Install g++
* Download file airport simulation project from Moodle.
* Find the code from Moodle
* Set up the code from all file

## 4. Running

* Compile every file from the directory
* Type command g++ \*cpp – o p to begin the program
* There are two modes to run the program

1. Step-by step:

Type p -p

1. Directing according to the point requirements

The details will shown in the next part of this report.

## 5. Work hours

|  |  |  |
| --- | --- | --- |
| Date | Work name | Implementation time |
| 30/11/2021 | Write the initial program. Think about the requirement and make some ideas. | 7 hours |
| 1/12/2021 | Arrange all the file to do the first part of projec P1 | 6.5 hours |
| 3/12/2021 | Complete P2 | 3 hours |
| 3/12/2021 | Complete P3,P4 | 8 hours |
| 5/12/2021 | Testing program | 2 hour |
| 8/12/2021 | Finish the program and write Readme.docx | 3 hours |

Total hour: 29.5 hours

## 6.Contact number

Email: [hung.truong@tuni.fi](mailto:hung.truong@tuni.fi)

Tel: 0403252908

7. Aiming score: 5/5

# Testing- Compilation proof

* Type p.exe on CMD

Text

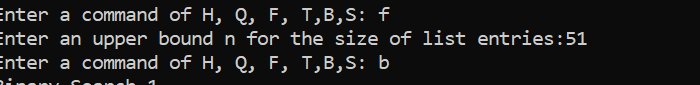
Description automatically generated

## 1.Group Number 1

Text

Description automatically generated

## 2.Group number 2



The results:

Text

Description automatically generated

## 3.Group number 3

Text

Description automatically generated

Graphical user interface, text

Description automatically generated

The results:

Text

Description automatically generated

## 4.Group number 4

Selection sort

A computer screen capture

Description automatically generated with medium confidence

Quick sort

A computer screen capture

Description automatically generated with medium confidence

The results:

Selection sort result

A computer screen capture

Description automatically generated with low confidence

Quick sort result:

A computer screen capture

Description automatically generated with low confidence

# Conclusion

1. There are 16 files in my project.

Graphical user interface, application

Description automatically generated

1. The main idea:

Diagram

Description automatically generated

Main\_1: Searching version

Main\_2: Sorting version

A screenshot of a computer

Description automatically generated with low confidence

Random,Timer are the files that imported into each main files.

Specific idea for each group:

#### Main\_1

1. Sequential\_search:

The main idea is that we will find a number in a list of length n by testing the all element of the list with for loop.

When the loop finish, if we don’t have suitable result, the comparision is n.

Moreover, when we find the suitable result in the list at any index in the list, we have the comparison is (index +1).

Text

Description automatically generated

1. Binary\_search

This algorithm works with a sort list. Therefore, we should create an order list before we start searching in this list.

We divide the list into 2 sublists and find the middle number of the list.

We will compare searched number with the middle number of the list to decide which sublist that the searched number belongs to.

We will continue these steps until we can’t implement the searching anymore. After that, we show the search results.

Text

Description automatically generated

#### Main\_2

#### 1.Insertion\_sort

Assume that we have 2 sublists in the list: one sorted list and one unsorted list.

We will check each element in the unsorted list and insert this element to the sort list until we have a new sorted list with the length equal to the length of original list.

We will compare the "current" with all elements in the sorted sublist and give it a suitable position. After implementing all members of the unsorted list. We done.

#### 2.Selection sort

In this algorithms, we will swap any two members of the list.

Firstly, we find the key of the biggest member of the list and swap this member with the last member of the list.

Secondly, we continue to find the biggest member of the list without the last member that we found before. We swap this member with the member which position is next to the last member that we found before.

We will continue until we have a new sorted list with a length equal to the length of the original list.

#### 3. Quick sort

Firstly, we find the member of the list such that we will partition the list into two sublists according to this member (key member). The value of the key member is in the middle of the list. For example, in the list '26 33 35 29 12 22', the key member is 26. One sublist with each member is smaller than the key member and one sublist with each member is bigger than the key member.

Secondly, we will implement each sublist by a similar method that we do above until we have a new sorted list.