Tampere University of Applied Sciences

Project Report 3

Project report

November 2021

Software Engineering

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Software Engineering

Minh Dang
Contact detail: minh.dang@tuni.fi, 0402579988

Arming score: 5

Project report 3

Programing Language 3 report 13 pages
November 2021

The purpose of this report was to summary solutions for project 3.

Key words:

Table of Contents

INTRODUCTION

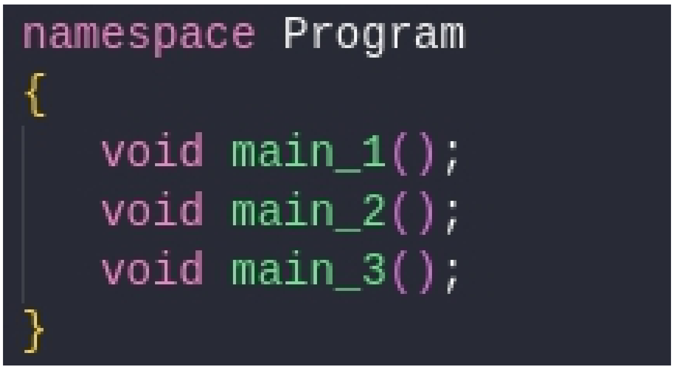
- The project was written by C++ and used GNU C++ compiler.

Building Project

- For running this project, there are 2 options:
    - Using GNU:
        - Installed g++ on your system first.
        - Command:
            - g++ *.cpp -o p && ./p
    - Using Meson[1]:
      The source code included Meson build configuration and tutorial on this GitHub repository: https://github.com/minhdangphuoc/Searching-and-Sorting-Testped-Program

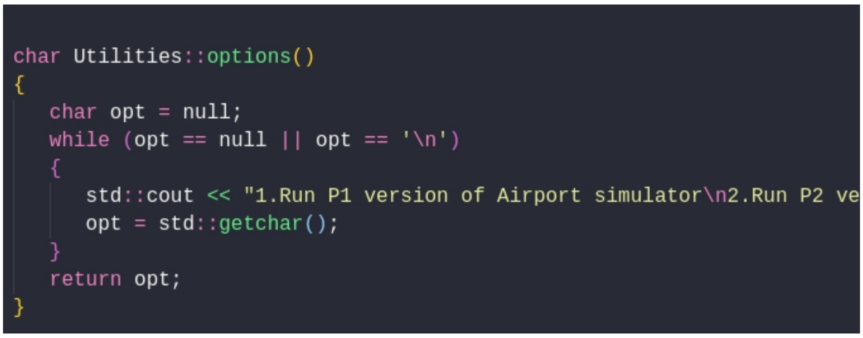
Program structure design

- I organizes Main() variations inside a namespace "Programs"



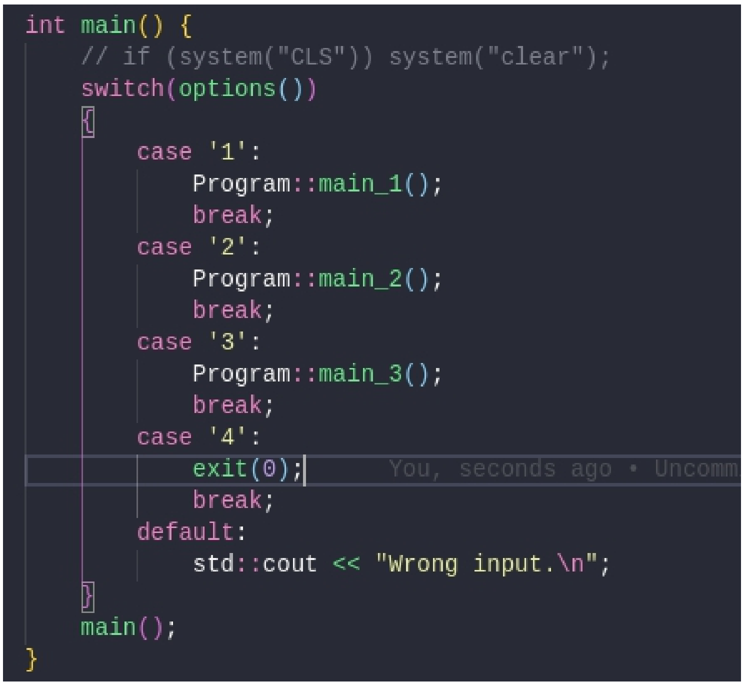
```
namespace Program
{
    void main_1();
    void main_2();
    void main_3();
}
```

- Option Picker



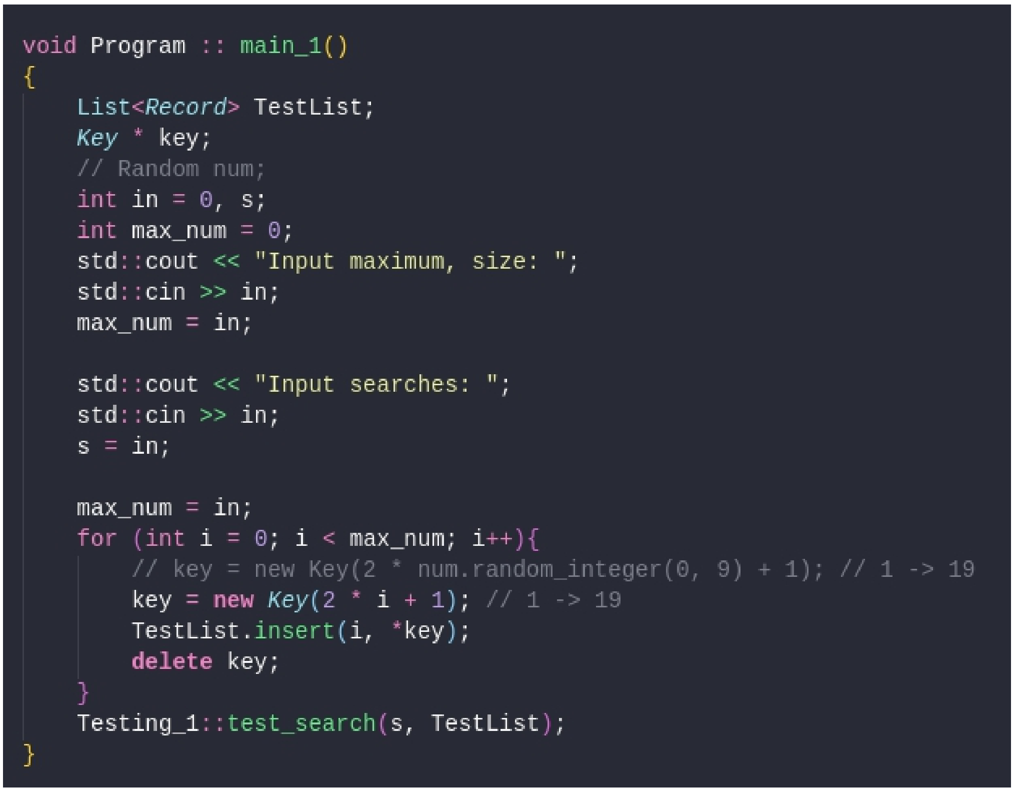
```
char Utilities::options()
{
    char opt = null;
    while (opt == null || opt == '\n')
    {
        std::cout << "1.Run P1 version of Airport simulator\n2.Run P2 ve
        opt = std::getchar();
    }
    return opt;
}
```

- Main() function



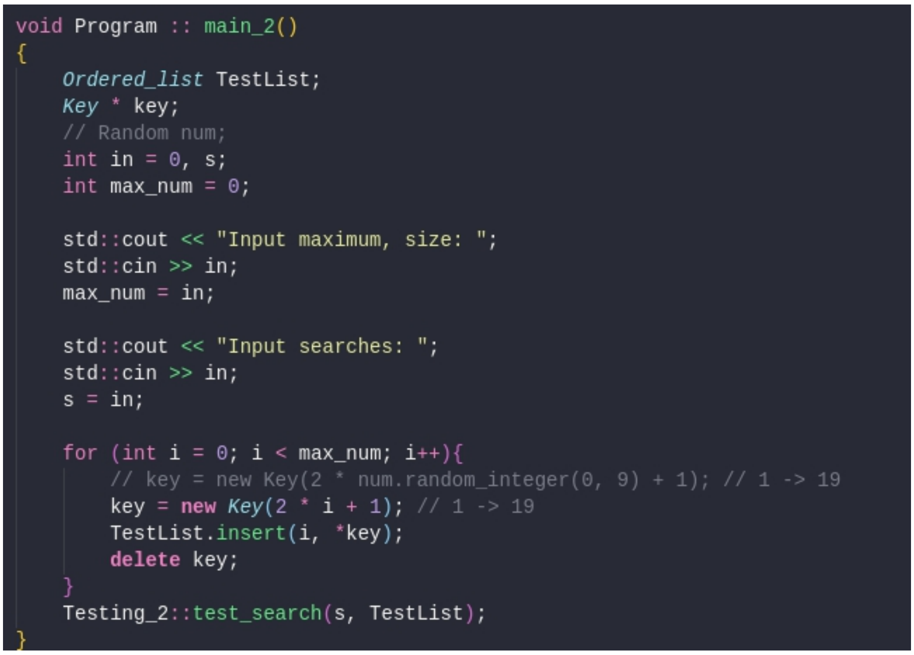
```
int main() {
    // if (system("CLS")) system("clear");
    switch(options())
    {
        case '1':
            Program::main_1();
            break;
        case '2':
            Program::main_2();
            break;
        case '3':
            Program::main_3();
            break;
        case '4':
            exit(0);      You, seconds ago • Uncomm
            break;
        default:
            std::cout << "Wrong input.\n";
    }
    main();
}
```

- Main_1() function



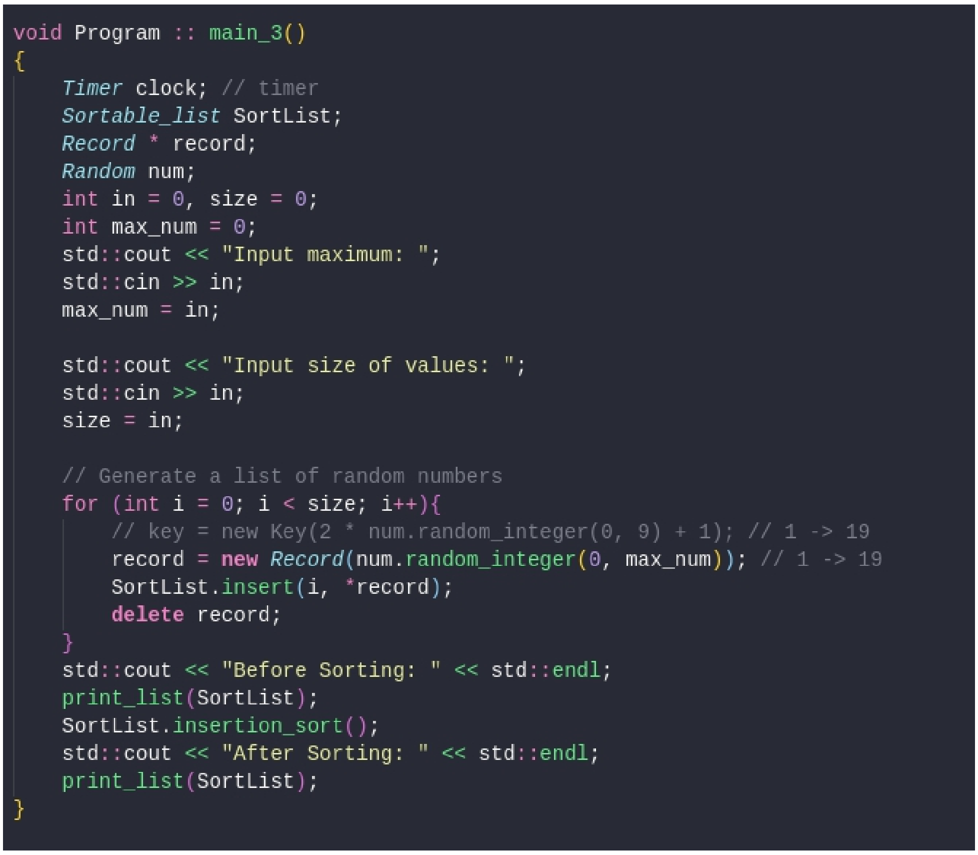
```
void Program :: main_1()
{
    List<Record> TestList;
    Key * key;
    // Random num;
    int in = 0, s;
    int max_num = 0;
    std::cout << "Input maximum, size: ";
    std::cin >> in;
    max_num = in;

    std::cout << "Input searches: ";
    std::cin >> in;
    s = in;

    max_num = in;
    for (int i = 0; i < max_num; i++){
        // key = new Key(2 * num.random_integer(0, 9) + 1); // 1 -> 19
        key = new Key(2 * i + 1); // 1 -> 19
        TestList.insert(i, *key);
        delete key;
    }
    Testing_1::test_search(s, TestList);
}
```

- Main_2() function



```
void Program :: main_2()
{
    Ordered_list TestList;
    Key * key;
    // Random num;
    int in = 0, s;
    int max_num = 0;

    std::cout << "Input maximum, size: ";
    std::cin >> in;
    max_num = in;

    std::cout << "Input searches: ";
    std::cin >> in;
    s = in;

    for (int i = 0; i < max_num; i++){
        // key = new Key(2 * num.random_integer(0, 9) + 1); // 1 -> 19
        key = new Key(2 * i + 1); // 1 -> 19
        TestList.insert(i, *key);
        delete key;
    }
    Testing_2::test_search(s, TestList);
}
```

- Main_3() function

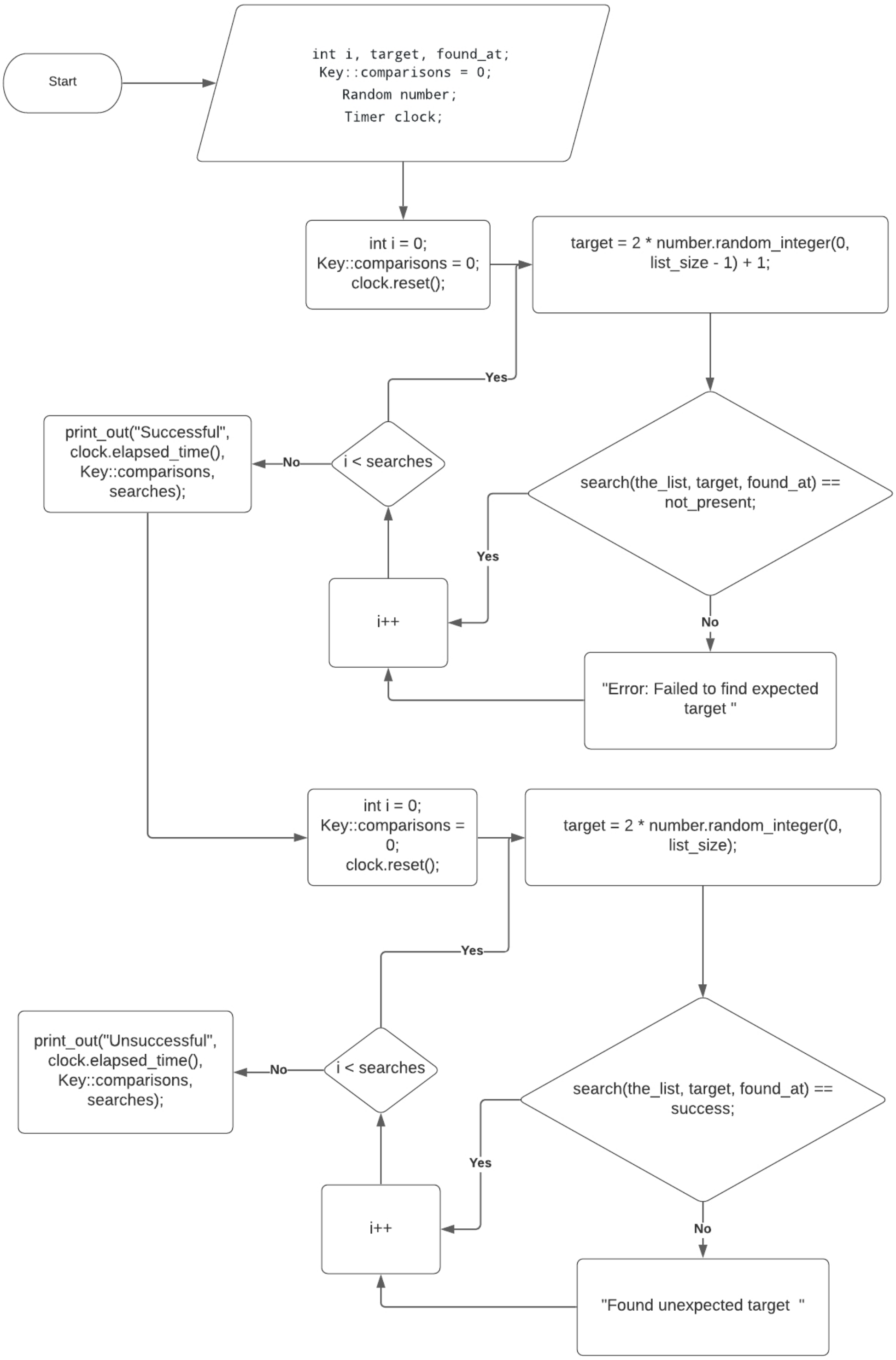


```
void Program :: main_3()
{
    Timer clock; // timer
    Sortable_list SortList;
    Record * record;
    Random num;
    int in = 0, size = 0;
    int max_num = 0;
    std::cout << "Input maximum: ";
    std::cin >> in;
    max_num = in;

    std::cout << "Input size of values: ";
    std::cin >> in;
    size = in;

    // Generate a list of random numbers
    for (int i = 0; i < size; i++){
        // key = new Key(2 * num.random_integer(0, 9) + 1); // 1 -> 19
        record = new Record(num.random_integer(0, max_num)); // 1 -> 19
        SortList.insert(i, *record);
        delete record;
    }
    std::cout << "Before Sorting: " << std::endl;
    print_list(SortList);
    SortList.insertion_sort();
    std::cout << "After Sorting: " << std::endl;
    print_list(SortList);
}
```

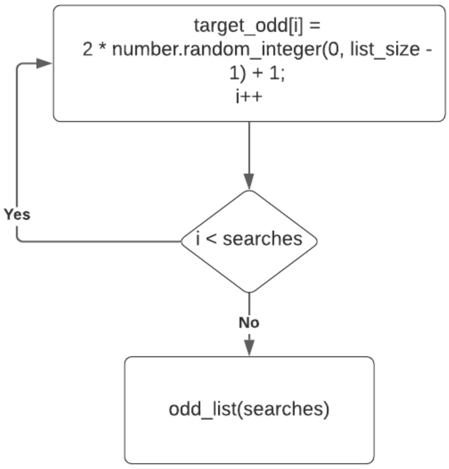
METHOD

Diagram

1. Group 1

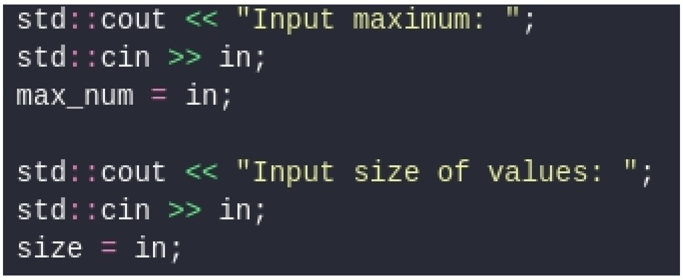

Start
int i, target, found_at;
Key::comparisons = 0;
Random number;
Timer clock;
int i = 0;
Key::comparisons = 0;
clock.reset();
target = 2 * number.random_integer(0, list_size - 1) + 1;
search(the_list, target, found_at) == not_present;
Yes
No
"Error: Failed to find expected target "
i++
i < searches
Yes
No
print_out("Successful", clock.elapsed_time(), Key::comparisons, searches);
int i = 0;
Key::comparisons = 0;
clock.reset();
target = 2 * number.random_integer(0, list_size);
search(the_list, target, found_at) == success;
Yes
No
"Found unexpected target "
i++
i < searches
Yes
No
print_out("Unsuccessful", clock.elapsed_time(), Key::comparisons, searches);

2. Group 2



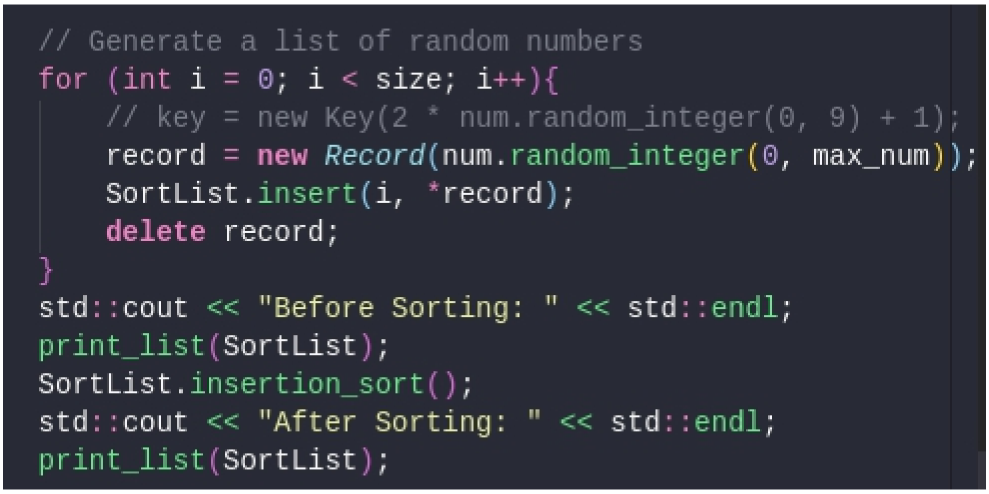
3. Group 3

a) Input maximum value and size of list



```
std::cout << "Input maximum: ";
std::cin >> in;
max_num = in;

std::cout << "Input size of values: ";
std::cin >> in;
size = in;
```

b) Processing part



```
// Generate a list of random numbers
for (int i = 0; i < size; i++){
    // key = new Key(2 * num.random_integer(0, 9) + 1);
    record = new Record(num.random_integer(0, max_num));
    SortList.insert(i, *record);
    delete record;
}
std::cout << "Before Sorting: " << std::endl;
print_list(SortList);
SortList.insertion_sort();
std::cout << "After Sorting: " << std::endl;
print_list(SortList);
```

Using test cases for solving and testing algorithms for each programs.

Test cases for Group 1:

	Sizes	Searches
Test 1	100	10
Test 2	1000	10
Test 3	10000	20

Test cases for Group 2:

	Sizes	Searches
Test 1	100	10
Test 2	1000	20
Test 3	10000	30

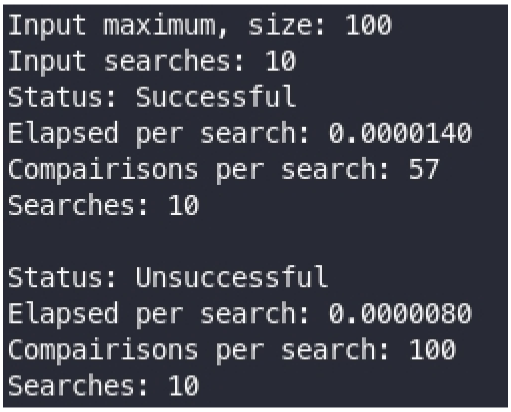
Test cases for Group 3:

	Sizes	Maximum
Test 1	500	10000
Test 2	1000	10000
Test 3	2000	10000

RESULTS

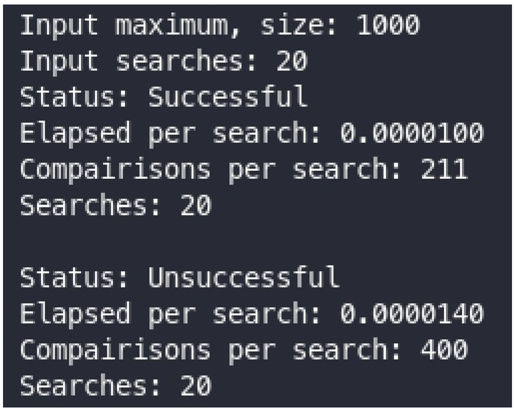
1. Group 1

a) Test 1



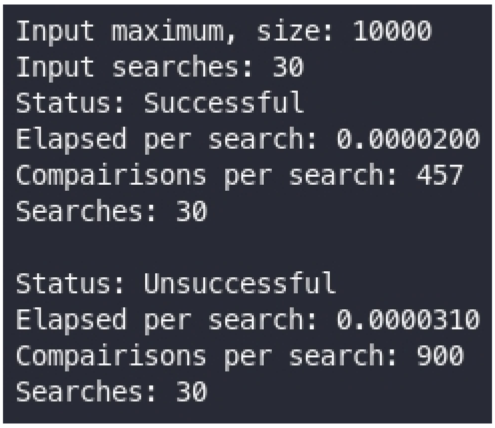
```
Input maximum, size: 100
Input searches: 10
Status: Successful
Elapsed per search: 0.0000140
Compairisons per search: 57
Searches: 10

Status: Unsuccessful
Elapsed per search: 0.0000080
Compairisons per search: 100
Searches: 10
```

b) Test 2



```
Input maximum, size: 1000
Input searches: 20
Status: Successful
Elapsed per search: 0.0000100
Compairisons per search: 211
Searches: 20

Status: Unsuccessful
Elapsed per search: 0.0000140
Compairisons per search: 400
Searches: 20
```

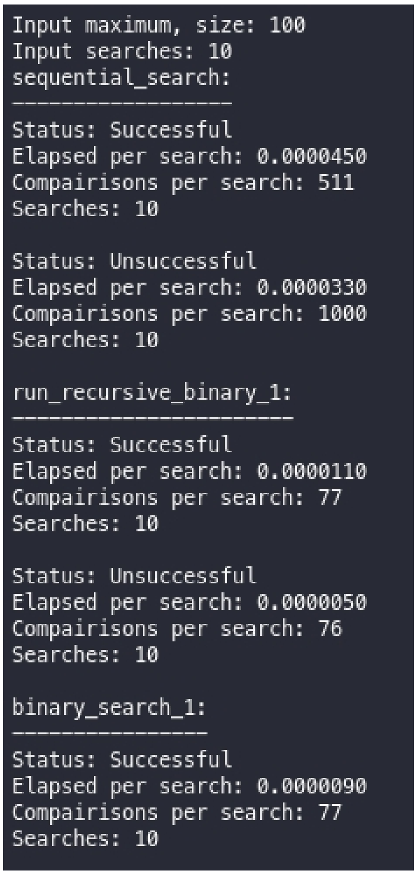
c) Test 3



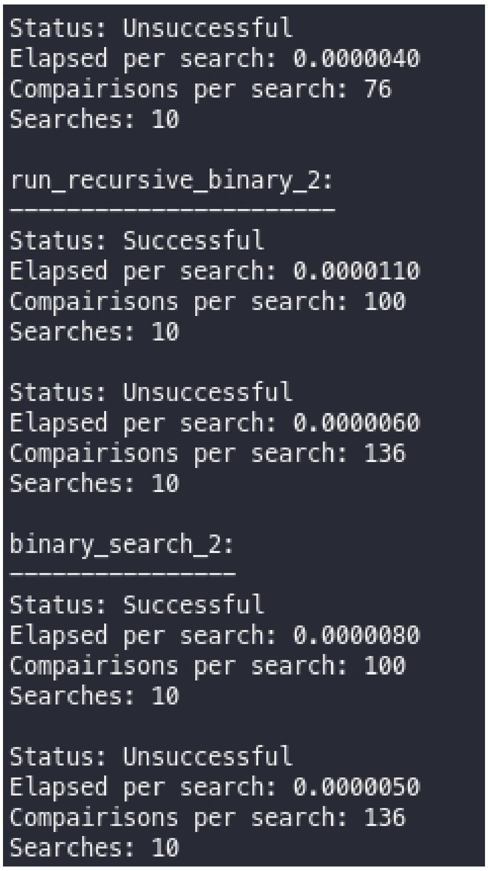
```
Input maximum, size: 10000
Input searches: 30
Status: Successful
Elapsed per search: 0.0000200
Compairisons per search: 457
Searches: 30

Status: Unsuccessful
Elapsed per search: 0.0000310
Compairisons per search: 900
Searches: 30
```

2. Group 2

a) Test 1

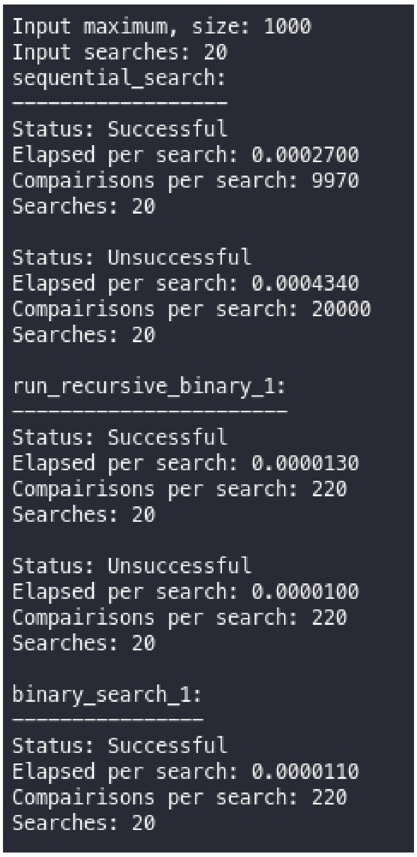


```
Input maximum, size: 100
Input searches: 10
sequential_search:
------------------
Status: Successful
Elapsed per search: 0.0000450
Compairisons per search: 511
Searches: 10

Status: Unsuccessful
Elapsed per search: 0.0000330
Compairisons per search: 1000
Searches: 10

run_recursive_binary_1:
-----------------------
Status: Successful
Elapsed per search: 0.0000110
Compairisons per search: 77
Searches: 10

Status: Unsuccessful
Elapsed per search: 0.0000050
Compairisons per search: 76
Searches: 10

binary_search_1:
----------------
Status: Successful
Elapsed per search: 0.0000090
Compairisons per search: 77
Searches: 10
```

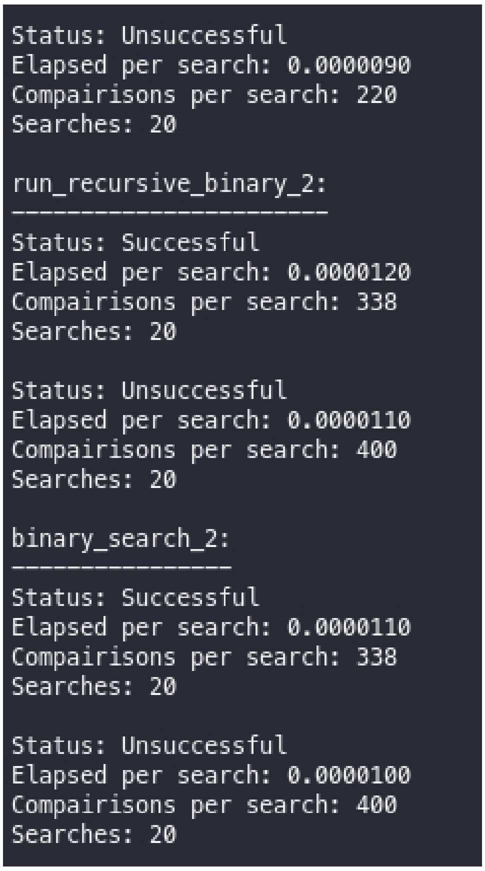


```
Status: Unsuccessful
Elapsed per search: 0.0000040
Compairisons per search: 76
Searches: 10

run_recursive_binary_2:
-----------------------
Status: Successful
Elapsed per search: 0.0000110
Compairisons per search: 100
Searches: 10

Status: Unsuccessful
Elapsed per search: 0.0000060
Compairisons per search: 136
Searches: 10

binary_search_2:
----------------
Status: Successful
Elapsed per search: 0.0000080
Compairisons per search: 100
Searches: 10

Status: Unsuccessful
Elapsed per search: 0.0000050
Compairisons per search: 136
Searches: 10
```

b) Test 2

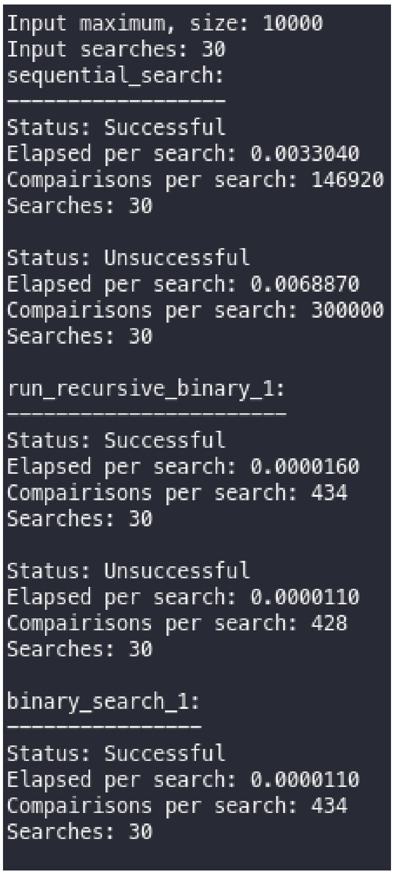


```
Input maximum, size: 1000
Input searches: 20
sequential_search:
------------------
Status: Successful
Elapsed per search: 0.0002700
Compairisons per search: 9970
Searches: 20

Status: Unsuccessful
Elapsed per search: 0.0004340
Compairisons per search: 20000
Searches: 20

run_recursive_binary_1:
-----------------------
Status: Successful
Elapsed per search: 0.0000130
Compairisons per search: 220
Searches: 20

Status: Unsuccessful
Elapsed per search: 0.0000100
Compairisons per search: 220
Searches: 20

binary_search_1:
----------------
Status: Successful
Elapsed per search: 0.0000110
Compairisons per search: 220
Searches: 20
```

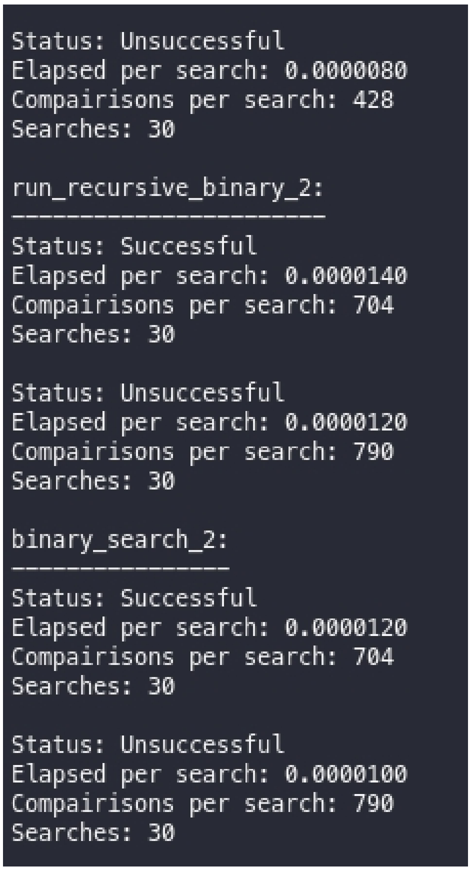


```
Status: Unsuccessful
Elapsed per search: 0.0000090
Compairisons per search: 220
Searches: 20

run_recursive_binary_2:
-----------------------
Status: Successful
Elapsed per search: 0.0000120
Compairisons per search: 338
Searches: 20

Status: Unsuccessful
Elapsed per search: 0.0000110
Compairisons per search: 400
Searches: 20

binary_search_2:
----------------
Status: Successful
Elapsed per search: 0.0000110
Compairisons per search: 338
Searches: 20

Status: Unsuccessful
Elapsed per search: 0.0000100
Compairisons per search: 400
Searches: 20
```

c) Test 3



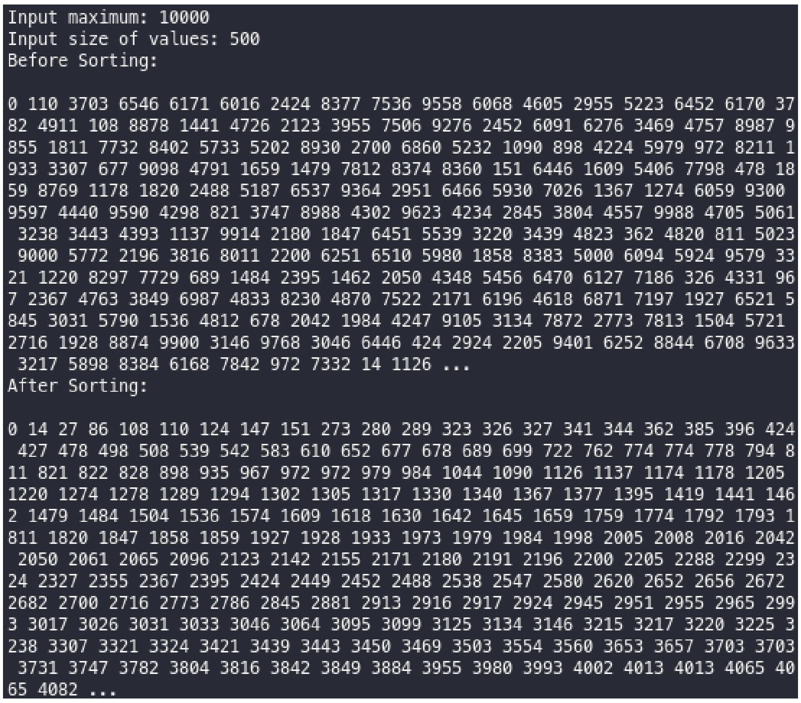
```
Input maximum, size: 10000
Input searches: 30
sequential_search:
------------------
Status: Successful
Elapsed per search: 0.0033040
Compairisons per search: 146920
Searches: 30

Status: Unsuccessful
Elapsed per search: 0.0068870
Compairisons per search: 300000
Searches: 30

run_recursive_binary_1:
-----------------------
Status: Successful
Elapsed per search: 0.0000160
Compairisons per search: 434
Searches: 30

Status: Unsuccessful
Elapsed per search: 0.0000110
Compairisons per search: 428
Searches: 30

binary_search_1:
----------------
Status: Successful
Elapsed per search: 0.0000110
Compairisons per search: 434
Searches: 30
```



```
Status: Unsuccessful
Elapsed per search: 0.0000080
Compairisons per search: 428
Searches: 30

run_recursive_binary_2:
-----------------------
Status: Successful
Elapsed per search: 0.0000140
Compairisons per search: 704
Searches: 30

Status: Unsuccessful
Elapsed per search: 0.0000120
Compairisons per search: 790
Searches: 30

binary_search_2:
----------------
Status: Successful
Elapsed per search: 0.0000120
Compairisons per search: 704
Searches: 30

Status: Unsuccessful
Elapsed per search: 0.0000100
Compairisons per search: 790
Searches: 30
```

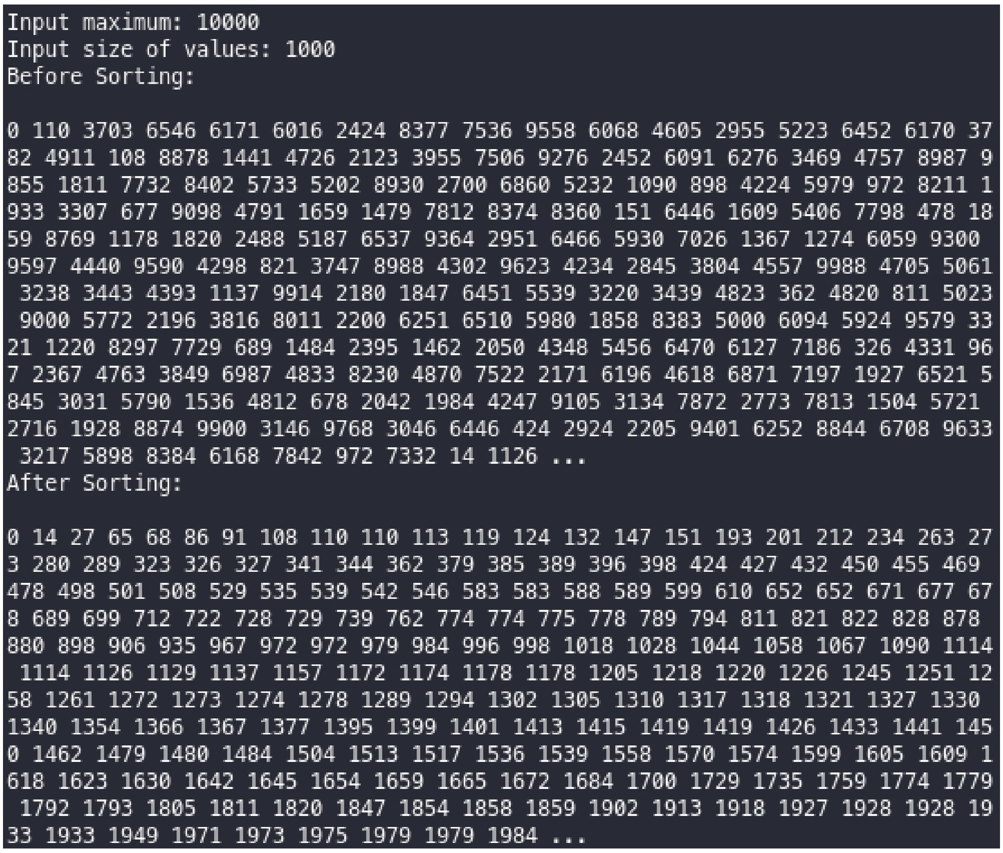
3. Group 3

a) Test 1



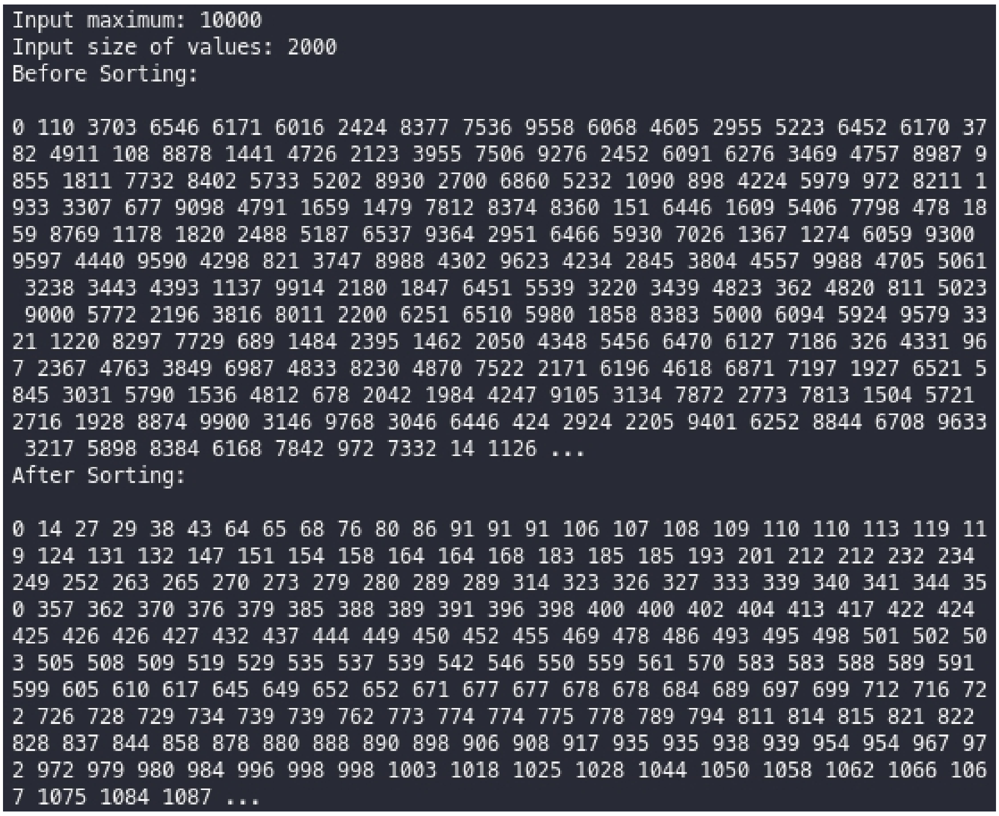
```
Input maximum: 10000
Input size of values: 500
Before Sorting:

0 110 3703 6546 6171 6016 2424 8377 7536 9558 6068 4605 2955 5223 6452 6170 37
82 4911 108 8878 1441 4726 2123 3955 7506 9276 2452 6091 6276 3469 4757 8987 9
855 1811 7732 8402 5733 5202 8930 2700 6860 5232 1090 898 4224 5979 972 8211 1
933 3307 677 9098 4791 1659 1479 7812 8374 8360 151 6446 1609 5406 7798 478 18
59 8769 1178 1820 2488 5187 6537 9364 2951 6466 5930 7026 1367 1274 6059 9300
9597 4440 9590 4298 821 3747 8988 4302 9623 4234 2845 3804 4557 9988 4705 5061
 3238 3443 4393 1137 9914 2180 1847 6451 5539 3220 3439 4823 362 4820 811 5023
 9000 5772 2196 3816 8011 2200 6251 6510 5980 1858 8383 5000 6094 5924 9579 33
21 1220 8297 7729 689 1484 2395 1462 2050 4348 5456 6470 6127 7186 326 4331 96
7 2367 4763 3849 6987 4833 8230 4870 7522 2171 6196 4618 6871 7197 1927 6521 5
845 3031 5790 1536 4812 678 2042 1984 4247 9105 3134 7872 2773 7813 1504 5721
2716 1928 8874 9900 3146 9768 3046 6446 424 2924 2205 9401 6252 8844 6708 9633
 3217 5898 8384 6168 7842 972 7332 14 1126 ...
After Sorting:

0 14 27 86 108 110 124 147 151 273 280 289 323 326 327 341 344 362 385 396 424
 427 478 498 508 539 542 583 610 652 677 678 689 699 722 762 774 774 778 794 8
11 821 822 828 898 935 967 972 972 979 984 1044 1090 1126 1137 1174 1178 1205
1220 1274 1278 1289 1294 1302 1305 1317 1330 1340 1367 1377 1395 1419 1441 146
2 1479 1484 1504 1536 1574 1609 1618 1630 1642 1645 1659 1759 1774 1792 1793 1
811 1820 1847 1858 1859 1927 1928 1933 1973 1979 1984 1998 2005 2008 2016 2042
 2050 2061 2065 2096 2123 2142 2155 2171 2180 2191 2196 2200 2205 2288 2299 23
24 2327 2355 2367 2395 2424 2449 2452 2488 2538 2547 2580 2620 2652 2656 2672
2682 2700 2716 2773 2786 2845 2881 2913 2916 2917 2924 2945 2951 2955 2965 299
3 3017 3026 3031 3033 3046 3064 3095 3099 3125 3134 3146 3215 3217 3220 3225 3
238 3307 3321 3324 3421 3439 3443 3450 3469 3503 3554 3560 3653 3657 3703 3703
 3731 3747 3782 3804 3816 3842 3849 3884 3955 3980 3993 4002 4013 4013 4065 40
65 4082 ...
```

b) Test 2



```
Input maximum: 10000
Input size of values: 1000
Before Sorting:

0 110 3703 6546 6171 6016 2424 8377 7536 9558 6068 4605 2955 5223 6452 6170 37
82 4911 108 8878 1441 4726 2123 3955 7506 9276 2452 6091 6276 3469 4757 8987 9
855 1811 7732 8402 5733 5202 8930 2700 6860 5232 1090 898 4224 5979 972 8211 1
933 3307 677 9098 4791 1659 1479 7812 8374 8360 151 6446 1609 5406 7798 478 18
59 8769 1178 1820 2488 5187 6537 9364 2951 6466 5930 7026 1367 1274 6059 9300
9597 4440 9590 4298 821 3747 8988 4302 9623 4234 2845 3804 4557 9988 4705 5061
 3238 3443 4393 1137 9914 2180 1847 6451 5539 3220 3439 4823 362 4820 811 5023
 9000 5772 2196 3816 8011 2200 6251 6510 5980 1858 8383 5000 6094 5924 9579 33
21 1220 8297 7729 689 1484 2395 1462 2050 4348 5456 6470 6127 7186 326 4331 96
7 2367 4763 3849 6987 4833 8230 4870 7522 2171 6196 4618 6871 7197 1927 6521 5
845 3031 5790 1536 4812 678 2042 1984 4247 9105 3134 7872 2773 7813 1504 5721
2716 1928 8874 9900 3146 9768 3046 6446 424 2924 2205 9401 6252 8844 6708 9633
 3217 5898 8384 6168 7842 972 7332 14 1126 ...
After Sorting:

0 14 27 65 68 86 91 108 110 110 113 119 124 132 147 151 193 201 212 234 263 27
3 280 289 323 326 327 341 344 362 379 385 389 396 398 424 427 432 450 455 469
478 498 501 508 529 535 539 542 546 583 583 588 589 599 610 652 652 671 677 67
8 689 699 712 722 728 729 739 762 774 774 775 778 789 794 811 821 822 828 878
880 898 906 935 967 972 972 979 984 996 998 1018 1028 1044 1058 1067 1090 1114
 1114 1126 1129 1137 1157 1172 1174 1178 1178 1205 1218 1220 1226 1245 1251 12
58 1261 1272 1273 1274 1278 1289 1294 1302 1305 1310 1317 1318 1321 1327 1330
1340 1354 1366 1367 1377 1395 1399 1401 1413 1415 1419 1419 1426 1433 1441 145
0 1462 1479 1480 1484 1504 1513 1517 1536 1539 1558 1570 1574 1599 1605 1609 1
618 1623 1630 1642 1645 1654 1659 1665 1672 1684 1700 1729 1735 1759 1774 1779
 1792 1793 1805 1811 1820 1847 1854 1858 1859 1902 1913 1918 1927 1928 1928 19
33 1933 1949 1971 1973 1975 1979 1979 1984 ...
```

c) Test 3



```
Input maximum: 10000
Input size of values: 2000
Before Sorting:

0 110 3703 6546 6171 6016 2424 8377 7536 9558 6068 4605 2955 5223 6452 6170 37
82 4911 108 8878 1441 4726 2123 3955 7506 9276 2452 6091 6276 3469 4757 8987 9
855 1811 7732 8402 5733 5202 8930 2700 6860 5232 1090 898 4224 5979 972 8211 1
933 3307 677 9098 4791 1659 1479 7812 8374 8360 151 6446 1609 5406 7798 478 18
59 8769 1178 1820 2488 5187 6537 9364 2951 6466 5930 7026 1367 1274 6059 9300
9597 4440 9590 4298 821 3747 8988 4302 9623 4234 2845 3804 4557 9988 4705 5061
 3238 3443 4393 1137 9914 2180 1847 6451 5539 3220 3439 4823 362 4820 811 5023
 9000 5772 2196 3816 8011 2200 6251 6510 5980 1858 8383 5000 6094 5924 9579 33
21 1220 8297 7729 689 1484 2395 1462 2050 4348 5456 6470 6127 7186 326 4331 96
7 2367 4763 3849 6987 4833 8230 4870 7522 2171 6196 4618 6871 7197 1927 6521 5
845 3031 5790 1536 4812 678 2042 1984 4247 9105 3134 7872 2773 7813 1504 5721
2716 1928 8874 9900 3146 9768 3046 6446 424 2924 2205 9401 6252 8844 6708 9633
 3217 5898 8384 6168 7842 972 7332 14 1126 ...
After Sorting:

0 14 27 29 38 43 64 65 68 76 80 86 91 91 91 106 107 108 109 110 110 113 119 11
9 124 131 132 147 151 154 158 164 164 168 183 185 185 193 201 212 212 232 234
249 252 263 265 270 273 279 280 289 289 314 323 326 327 333 339 340 341 344 35
0 357 362 370 376 379 385 388 389 391 396 398 400 400 402 404 413 417 422 424
425 426 426 427 432 437 444 449 450 452 455 469 478 486 493 495 498 501 502 50
3 505 508 509 519 529 535 537 539 542 546 550 559 561 570 583 583 588 589 591
599 605 610 617 645 649 652 652 671 677 677 678 678 684 689 697 699 712 716 72
2 726 728 729 734 739 739 762 773 774 774 775 778 789 794 811 814 815 821 822
828 837 844 858 878 880 888 890 898 906 908 917 935 935 938 939 954 954 967 97
2 972 979 980 984 996 998 998 1003 1018 1025 1028 1044 1050 1058 1062 1066 106
7 1075 1084 1087 ...
```

WORKING HOURS

Date	Task	Duration
29/11/2021	Download Source Code on Moodle	10m
29/11/2021	Edited base code and library for the first requirement.	1h
29/11/2021	Finished first requirement	20m
4/12/2021	Edited base code and library for the 2nd requirement.	30m
4/12/2021	Coding and add modifications	30m
4/12/2021	First search running	5m
5/12/2021	Finished second requirement	30m
12/12/2021	Edited base code and library for the 2nd requirement.	15m
12/12/2021	Finished third requirement	60m

REFERENCE

[1] Meson C++ Build