VINUNIVERSITY

Lab Practice № 9: Sorting and Searching Algorithms

COMP1010 Introduction to Programming **Week 10**

Lab Practice Submission Instructions:

- This is an individual lab practice and will typically be assigned in the laboratory (computer lab).
- Your program should work correctly on all inputs. If there are any specifications about how the program should be written (or how the output should appear), those specifications should be followed.
- Your code and functions/modules should be appropriately commented. However, try to avoid making your code overly busy (e.g., include a comment on every line).
- Variables and functions should have meaningful names, and code should be organized into functions/methods where appropriate.
- Academic honesty is required in all work you submit to be graded. You should **NOT** copy or share your code with other students to avoid plagiarism issues.
- Use the template provided to prepare your solutions.
- You should upload your .py file(s) to the Canvas **before the end of the laboratory session** unless the instructor gave a specified deadline.
- Submit separate .py file for each Lab problem with the following naming format, for example: **V202000999_Lab9.py**. **Note:** If you are working on Jupiter Notebook, you need to download/convert it to Python .py file for submission.
- Late submission of lab practice without an approved extension will incur the following penalties:

  (a) No submission by the deadline will incur 0.25 point deduction for each problem (most of the problems are due at the end of the lab session).

  (b) The instructor will deduct an additional 0.25 point per problem for each day past the deadline.

  (c) The penalty will be deducted until the maximum possible score for the lab practice reaches zero (0%) unless otherwise specified by the instructor.

Problem 1 – Shorting participants

VinUni organized a programming competition in this summer. After evaluating all the submissions, they announced the Top N shortlisted participants for the next round. Write a program that allows the participant to check if they have been shortlisted or unsuccessful. An example of top 10 participant's ID are as follows:

122 332 234 55 401 102 331 234 224 387

Requirements:

1. Prompt user to enter the top participant's ID, stored them as a list with a variable named `winner_ID`.
2. Sort these winner ID using a built-in method and print out the sorted list.
3. Implement your own binary search to support the search function (using built-in or third-party package is not accepted for this task).
4. Prompt user to enter their ID.
5. Show the search result as follows:

Sample program:

```
Enter the top participant's ID: 122 332 234 55 401 102 331 234 224 387
Sorted top participant's ID: [55, 102, 122, 224, 234, 234, 331, 332, 387, 401]
Enter participant's ID: 102
Congratulations! You have been shortlisted in the Top 10 "Best Programmer 2020".

Enter the top participant's ID: 122 332 234 55 401 102 331 234 224 387
Sorted top participant's ID: [55, 102, 122, 224, 234, 234, 331, 332, 387, 401]
Enter participant's ID: 235
Unsuccessful!
```

Submit your solution to Canvas using the following sample program above.

Problem 2 – Sorting a list in ascending order

Write a selection sort program that accepts the user's input and then outputs a sorted list in ascending order. Prompt the user for the number of elements to be sorted. Submit your solution to Canvas using the following sample program:

```
Enter number of elements: 6
3
5
2
89
43
56
Sorted array:
2 3 5 43 56 89
```

Problem 3 – Partially sorted list

You have the following list of numbers to sort: A = [5, 4, 6, 3, 1, 2]. Write a program to show the partially sorted list after each complete passes of `selection sort`.

Sample program:

```
Partially Sorted List 1:
1 4 6 3 5 2

Partially Sorted List 2:
1 2 6 3 5 4

Partially Sorted List 3:
1 2 3 6 5 4

Partially Sorted List 4:
1 2 3 4 5 6

Partially Sorted List 5:
1 2 3 4 5 6

Partially Sorted List 6:
1 2 3 4 5 6

Sorted List:
1 2 3 4 5 6
```

Submit your solution to Canvas using the following sample program above.

Problem 4 – Linear search vs Binary search

Prompt user to enter a list of integer numbers (separated by a space), and an integer key to search. Complete the following search task:

1. Linear search: Write a function to perform linear search, print out the list of numbers which was checked, and the final search result (Found or Not Found).
2. Using built-in function to sort the list.
3. Binary search: Write a function to perform binary search, print out the list of numbers which was checked, and the final search result (Found or Not Found):

Using the example input and output below to submit your code to CMS.

Sample program #1:

```
Enter a list of integer numbers: 45 98 27 68 95 36 75
Enter a number to search: 27
Linear Search: List of checked numbers:  [45, 98, 27]
Found!
Sorted List: [27, 36, 45, 68, 75, 95, 98]
Binary Search: List of checked numbers:  [68, 36, 27]
Found!
```

Sample program #2:

```
Enter a list of integer numbers: 26 84 43 27 71 32 40 55 53
Enter a number to search: 6
Linear Search: List of checked numbers:  [26, 84, 43, 27, 71, 32, 40, 55, 53]
Not Found!
Sorted List: [26, 84, 43, 27, 71, 32, 40, 55, 53]
Binary Search: List of checked numbers:  [71, 84, 26]
Not Found!
```