SCS214: Data Structures
Assignment-1: Sorting

Instructions

1- The assignment is submitted in groups of **maximum 3** students from the same **lab OR** same **TA.**
2- Deadline of submission is **Wed. Mar.13th at 11:55 pm.**
3- Submission will be on Google classroom.
4- Your submission should include a single **cpp file**, named by LabGroup_ID1_ID2_ID3 (ex.: S13_20220022_20220023_20220024.cpp).
5- No late submission is allowed.
6- No submission through e-mails.
7- No rar files
8- No exe file submission.
9- **In case of Cheating you will get a negative grade whether you give the code to someone, take the code from someone/internet, or even send it to someone for any reason.**
10- You have to write clean code and follow a good coding style including choosing meaningful variable names.
11- In case of wrong submission, wrong file extension/type, missing files, plagiarism, extra submitted files, wrong naming, the assignment will not be accepted and no correction for these mistakes is allowed and you will lose your grade.

Task 1: Insertion sort and Binary Search

Insertion sort uses linear search to find the right place for the next item to insert. Would it be faster to find the place using binary search (reduce number of comparisons)?

We still have to shift 1 item at a time from the largest till the right place. Use binary search on the already sorted items to find the place where the new element should go and then shift the exact number of items that need to be shifted and place the new item in its place.

The algorithm works the same, except that instead of comparing and shifting item by item, it will compare quickly using binary search but it will still shift one by one till the right place.

SCS214: Data Structures
Assignment-1: Sorting

Requirements:

- **Input (From Console):**
  - Size of an array.
  - The Array to perform the sort on.
- **Implement** insertion sort algorithm using **linear search** and **binary search to compare their results.**
- **Calculate** the number of comparisons and shifts done in both cases.
- **Print:**

  (1) The sorted array for each case.

  (2) The number of comparisons for each case.

  (3) The number of shifts for each case.

Task 2

Implement a program that reads input from a file, sorts each array in the file using three different algorithms (Selection, Shell, and Merge Sort), and calculates the number of moves and comparisons performed by each algorithm.

Input:

- The program should read the input from a text file ("arrays.txt").
- Each line in the file represents a record, the **first** element of the record represents the **size of an array**, and **the rest** represents the **values of the array** separated by spaces.

SCS214: Data Structures
Assignment-1: Sorting

Implement the following sorting algorithms:

- Selection Sort
- Shell Sort
- Merge Sort

Note: Follow the algorithms written in the lecture.

Output:

- For each array:
  - Print the original array.
  - For each sorting algorithm:
    - Print the sorted array.
    - Print the number of comparisons performed by the algorithm
    - Print the number moves performed by the algorithm.
      (note that a swap involves 3 moves)

SCS214: Data Structures
Assignment-1: Sorting

Sample input:

```
4 4 2 1 8
5 10 5 7 3 6
4 2 9 6 1
3 8 4 1
```

Sample Output:

```
Original array 4 2 1 8
Selection Sort:
   Sorted array: 1 2 4 8
   Comparisons: X
   Moves: Y
Shell Sort:
   Sorted array: 1 2 4 8
   Comparisons: X
   Moves: Y
Merge Sort:
   Sorted array: 1 2 4 8
   Comparisons: X
   Moves: Y

Original array: 10 5 7 3 6
(Similar output for each sorting algorithm, with different comparison and move
counts)
```

SCS214: Data Structures
Assignment-1: Sorting

Grading Criteria:

Task 1:

Main	**5**
Insertion sort with Linear search	**10**
Insertion sort with Binary search	**15**
Shifts for both cases	**5**
Comparisons for both cases	**5**
Sorted Array for both cases	**5**
Total	**40**

Task 2:

Main	**2.5**
Reading input from File	**5**
Selection Sort	**10**
Shell Sort	**10**
Merge Sort	**10**
Comparisons for the 3 Algorithms	**7.5**
Moves for the 3 Algorithms	**7.5**
Sorted Array for the 3 Algorithms	**7.5**
Total	**60**