Practice 8

Problem 1: Array Comparison

- Write a program that tests whether two arrays are equal or not.
- Declare two integer arrays with 7 elements, and fill up the arrays using user's input
- Test if every elements of one array is equal to its corresponding elements in the other array. For example, the program should check if A[0] = B[0], A[1] = B[1], and so forth.
- Example output

```
Element 1 in A: 5
Element 2 in A: 3
…
Element 7 in B: 15
Two arrays are the same. (or are not the same)
```

Problem 2: Random Numbers without Duplication

- Write a program that generates 100 random integers (1~200) and store them into an array.
- The numbers in the array must not be duplicated (refer to Program 8-5)
- After the generation, save the array into a txt file (white space for number separator)
- File output example

```
23 184 1 29 ... 99 158 37
```

Problem 3: Filling up Array Using File Input

- Write a program to fill up an 100-integer array using file input.
  - Declare an integer array with 100 elements.
  - Read the file you created in Problem 2, and fill up the array with the data in the file.
- Sort the 100 numbers and calculate median
  - Use your favorite sorting algorithm.
  - Display both the original array and result of sorting
  - Find the median of the numbers and display it.

Problem 4: Number of Exchange: Selection and Bubble Sort

- Read the file you created in Problem 2 and fill up an 100-integer array.
- Sort the array twice. Use first the selection sort (Program 8-6, 8-7) and then bubble sort (Program 8-8, 8-9) algorithms
- During the sorting, count the number of exchanges needed to order an array.
- For each sorting algorithm, display the array before and after the sort and the total exchanges needed to sort the array.
- Example output

```
For the selection sort
Before sorting: 23 184 1 29 … 99 158 37
After sorting: 1 3 4 … 194 197 199
Number of exchanges: xxx

For the bubble sort
Before sorting: 23 184 1 29 … 99 158 37
After sorting: 1 3 4 … 194 197 199
Number of exchanges: xxx
```

Problem 5: Binary Search

- Read the file you created in Problem 2 and fill up an 100-integer array.
- Sort the array using your favorite sorting algorithm.
- Build a user interface to repeatedly get a user's input for a key to search.
- Find the key in the array using the binary search algorithm (Problem 8-14)
- Display the result of the search (the existence of the key in the array and the location if it exist).
- For each search, count the number of comparison for the search and display it.
- Example

```
Complete reading the file and sorting the array.
Enter a key to find: 41
There is 41 at 25
Number of comparison for the search: xxx
Enter a key to find: 5
Not found
Number of comparison for the search: xxx
Enter a key to find: 99999
Bye~
```

Problem 6: Student Ranking

- Write a student grade management program
- Read a text file containing students' name and scores
  - File format

```
[number of student]
[student name] [score1] [score2] [score3] [score4]
…
[student name] [score1] [score2] [score3] [score4]
```

- Calculate the grade
  - Use different weight for each score
  - score1:score2:score3:score4 = 10:40:20:30
  - Grading based on weighted average
  - 100~90:A, 89.9~80:B, 79.9~70:C, 69.9~60:D, 59.9~0:F
  - Calculate the ranking among the students
- Make an output file and write the following information on it
  - File format

```
[student name] [grade] [ranking]
…
[student name] [grade] [ranking]
```

Input file example

```
5
Sam 70 58 49 56
Mark 90 80 90 90
Jane 78 95 90 100
Minsu 20 100 100 20
Paul 80 60 70 80
```

Output file example

```
Sam F 5
Mark B 2
Jane A 1
Minsu D 4
Paul C 3
```