10 LABS x 2 hours 15 minutes

- Lab 1: OOP Reviews & Arrays
- Lab 2: Simple sorting
- Lab 3: Stacks & Queues
- Lab 4: Linked List
- Lab 5: Recursion
- Lab 6: Trees
- Lab 7: Hash Tables
- Lab 8: Graph
- Lab 9: Exam
- Lab 10: Project Presentation

There are 8 practical labs (30%):

- Select 3 random submissions to mark
- If you miss a lab or a submission: that lab will be selected to mark

Lab 9 will be a practical exam (35%)

- You can use your laptop to code
- You are only allow to use the following IDE:
    - NetBeans
    - VS Code
    - BlueJ
    - IntelliJ
- You must DISCONNECT your laptop from the Internet

Lab 10 is the project presentation (35%)

Deadline to submit your work on Blackboard: 3 days from the lab day

- i.e., Lab day is Monday => deadline is Wednesday (mid-night)

Assignments submission guide

- Create the folder with a name like: ***StudentID_Name_Lab#***, (e.g. ***01245_VCThanh_Lab1***) to contain your assignment with subfolders:
    - Problem_01 (sometimes Problem_i or Problem_Array)
    - Problem_02 (sometimes Problem_ii or Problem_Queue)
    - etc.
- Compress (.zip) and Submit the whole folder with the same name (i.e., ***01245_ VCThanh_Lab1.zip***) to Blackboard
- Students **not** following this rule **will get their marks deducted**

1. Lab 1: OOP Reviews & Arrays

1.1. Objectives

i. Review basic OOP and Java skills
ii. Arrays and operations with arrays: find, insert, delete
iii. Ordered arrays, binary search
iv. Implementing arrays is Java
v. Arrays of objects
vi. Efficiency of aray operations, big O notation

1.2. Problem 1: Simple application

i. Write a function to convert array to number. (10pts)

Suppose we have loaded an array with the digits of an integer, where the highest power is kept in position 0, next highest in position 1, and so on.

The ones position is always at position array.Length - 1:

```
int[] digits = { 2, 0, 1, 8 };
```

ii. Write a function to input a list of integer numbers and return the median of that list (10pts).

iii. Find the min-gap (10pts)

Write a method named minGap that accepts an integer array and a number of elements as parameters and returns the minimum 'gap' between adjacent values in the array. The gap between two adjacent values in an array is defined as the second value minus the first value.

For example, suppose a variable called array is an array of integers that stores the following sequence of values:

```
int[] array = {1, 3, 6, 7, 12};
```

The first gap is 2 (3 - 1), the second gap is 3 (6 - 3), the third gap is 1 (7 - 6) and the fourth gap is 5 (12 - 7).

Thus, the call of minGap(array, n) should return 1 because that is the smallest gap in the array. If you are passed an array with fewer than 2 elements, you should return 0.

iv. BankApp.java

Move the BankAccount class into a separate file. Note the class structure in BlueJ.

For each transaction print its type and the balance after the transaction

Add a method to class BankAccount that retruns the balance. Use it to print the current balance instead of using display().

Create a second bank account and move the entire balance of the first account to the second one

v. Student.java, Students.java, students.txt (text files, loops, decision making, access modifiers)

Add public/private to the declaration of student names and print students with System.out.println(st.lname), explain.

Change the while-loop with a for-loop

Use grade to determine the type of the student: excellent (> 89), ok [60,89], and failure (< 60)

Do counting and averaging within each student type (excellent, ok, and failure)

1.3. Problem 2

i. Compile and Run the Example Programs

- Array, LowArray, HighArray
- OrderedArray
- ClassDataArray

ii. Programming Projects 2.1 in Text-Book (array.java)

Modify the method in Programming Project 2.1 so that the item with the highest key is not only returned by the method, but also removed from the array. Call the method removeMax().

Read the array items from a text file (one array item per line) and read the item to find from user input (see GasMileage.java)

Generate random numbers ([0,99]) to fill the aray with 100 items and find and delete a random item ([0,99]).

Print the number of comparisons to find an item and the number of memory moves to delete an item.

Change the range of random numbers (use nextInt()) and explain the change in the number of comparisons.

iii. Programming Projects 2.2 in Text-Book (lowArray.java)

iv. Programming Projects 2.5 in Text-Book (classDataArray.java)

Add a merge() method to the OrderedArray class in the orderedArray.java program (Listing 2.4) so that you can merge two ordered source arrays into an ordered destination array. Write code in main() that inserts some random numbers into the two source arrays, invokes merge(), and displays the contents of the resulting destination array. The source arrays may hold different numbers of data items. In your algorithm you will need to compare the keys of the source arrays, picking the smallest one to copy to the destination. You'll also need to handle the situation when one source array exhausts its contents before the other.