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**

2. Lab 2: Simple sorting

2.1. Objectives

i. Know how, in reality, three simple sorting methods work.
ii. Know how to use analysis tool to compare performance of sorting algorithms

2.2. Problem 1: BubbleSortApp.java

- Trace the algorithm (display the array inside after inner or outer loop)
- Display the number of swaps after the inner loop
- Display the number of comparisons after the inner loop and the total number of comparisons, and estimate the algorithms' complexity (n*(n-1)/2, O(n^2))

2.3. Problem 2: SelectSortApp.java

- Trace the algorithm (display the array after the inner loop)
- Print the items that are swapped. Are swaps always needed?
- Display the number of comparisons after the inner loop and the total number of comparisons, and estimate the algorithms' complexity (n*(n-1)/2, O(n^2))

2.4. Problem 3: InsertSortApp.java

- Trace the algorithm (display the array after each pass of the outer loop)
- Display the number of passes of the inner loop and total number of passes, and estimate the algorithms' complexity (n*(n-1)/4, O(n^2))

2.5. Problem 4

Create an array of integer numbers, fill the array with random data and print the number of **comparisons, copies, and swaps** made for sorting 10000, 15000, 20000, 25000, 30000, 35000, 40000, 45000 and 50000 items and fill in the table below. Analyze the trend for the three different algorithms.

COPIES/ COMPARISONS/ SWAPS			
	Bubble Sort	**Selection Sort**	**Insertion Sort**
10000			
15000			
20000			
25000			
30000			
35000			
40000			
45000			
50000			

2.6. Problem 5: ObjectSortApp.java (sort the array by first name or by age)

(Option 1) Given the class Student.java that has variables of first name, last name, grade

- Add a main() method and add create an array of 10 students

- Add methods to sort the array by first name, last name, and by grade.