We will use IntelliJ as our ‘IDE’ initially. Please see note on using IntelliJ.

You will be using GitHub to store your code as part of this module. Please see note on using GitHub. Make sure you save your work. You lecturer will want to examine it at a later stage.

You should create your own user defined methods in your solutions to the problems below. All the solutions to this problem Set should be in individual java classes, in one package, in one project.

**Problem Set 0**

1) A simple calculation. The canteen is offering a special start-of-year mineral-and-muffin snack deal at 2 euro per snack. Write a program which will ask you to enter your name, your class name and how many snacks you’d like, and which will calculate and display in a dialog your name and class, the number of snacks you asked for and what you will have to pay.

2) Calculations and if – else (the sponsored cyclist). A cyclist is sponsored at the rate of 7c for each km up to 10 km, and 10c for each km in excess of 10km. Write a program which will ask for the cyclist’s first name, initial and last name and the distance cycled. Your program should calculate the money due, then display a dialog containing the cylist’s full name (first name + initial + last name), the distance cycled, and the money due. Note here that the problem involves **2 different rates** if the number of km exceeds 10 so, for example, if the cyclist is sponsored for 15km then the first 10km are sponsored at 7c and the remaining 5 km are sponsored at 10c giving

3. If and else – if. Write a program which will ask for a person’s name and taxable income, calculate the tax due and the net income after tax, and display the gross income, the tax rate, the tax and the income after tax. Use the following table:

Taxable Income(€) Tax Rate (%)

----------------------- ---------------

0.00 – 20,000.00 0

20,000.01 – 36,000.00 20

36000.01 and upwards 41

Take the simple approach: if the person earns, say 25,000, tax him /her at 20% on all of it.

4. Write a program that uses a *while* loop. In each iteration of the loop, prompt the user to enter a number – positive, negative, or zero. Keep a running total of the numbers the user enters and also keep a count of the number of entries the user makes. The program should stop whenever the user enters “q” to quit. When the user has finished, print the grand total and the number of entries the user typed.

5. Switch case, and char variables. Write a program that reads in the following values:-

(a) operation desired - A (for addition)

S (for subtraction)

M (for multiplication)

D (for division)

(b) a number

(c) another number

The program should carry out the operation on the numbers e.g. if the input is M 1.5 3 then the output would be 1.5\*3 i.e. 4.500. Use a **switch statement** here and have a **default case** to handle any exceptional circumstances. Use dialogs for all input and output.

6. Loop with sentinel. Write a version of your solution to Q 5 which loops to allow you to process a number of calculations, stopping only when you enter ‘Q’ or ‘Quit. [Ambitious students (optional): use the showConfirmDialog() method from JOptionPane instead of the stopping code of Quit to control the loop.]

7. VOPC and class diagrams. Have a good look at the sample program DialogDemoC, then do the following:

(i) Draw a UML class diagram for DialogDemoC

(ii) Draw a UML VOPC diagram showing all the classes needed for DialogDemoC

(iii) List all variables and objects used within main, and their types

(iv) List all methods used within main, and the classes they belong to

(v) Look up the documentation for one of the classes used by DialogDemoC, find a useful method not invoked by the program, and add some code which uses it.

[Optional exercise] Equilibrium Index

An Equilibrium Index of a sequence of numbers is an index where the sum of all

elements with a lower index is equal to the sum of all elements with a higher index

(the number in the sequence at the index is not included in either sum).

In the sequence 1; 5;-7; 2; 3;-4; 0 the equilibrium indices are 3 and 6 since

1 + 5 + (-7) = 3 + (-4) + 0 and 1 + 5 + (-7) + 2 + 3 + (-4) = 0

Given a sequence of numbers, the task is to find the equilibrium indices of the

sequence

Input

First line of input contains a single integer N, 0 < N < 10; 000; 000. N is the

length of the sequence. The following line contains n integers between 1; 000; 000

and 1; 000; 000 inclusive, separated by spaces and followed by a newline character.

Output

The output should be a list of space separated equilibrium indices for the given list

sorted from lowest to highest index.

Sample Input 1

4

6 2 5 1

Sample Output 1

1

Sample Input 2

7

1 5 -7 2 3 -4 0

Sample Output 2

3 6

[Optional Exercise] Flipping Numbers

Given a list of N unique integers, 2 \_ N \_ 25000, produce a sequence of flips"

so that the end result is the list, sorted in ascending order. A k-flip takes the first k

numbers and reverses their order in the list. For example given the list (2; 6; 4; 1; 9),

a k-flip where k = 4 would result in the list (1; 4; 6; 2; 9).

Your task is to determine a list of k values which represent the sequence of flips

that will result in list being sorted.

Input

The first line of input contains the integer N which represents the number of integers

in the list. The next line of input contains N space-separated integers.

Output

Your program should output the sequence of k values which represent the flips that

turn the input list in to a sorted list. Each of these values should be separated by

a space and the output should be terminated by a new line character.

Sample Input 1

5

1 4 6 2 9

Sample Output 1

3 4 2

4

Iterations

14629

64129

21469

12469

Sample Input 2

5

2 7 5 6 3

Sample Output 1

2 5 2 4 2 3 2

8

Sequence

27563

72563

36527

63527

25367

52367

32567

23567