Programming Technique

Lab 3

1 Expected outcomes

- Be able to use standard libraries and built-in functions.
- Know how to execute simple operations to manage input/output.
- Understand control statements and be able to use them to solve required problems.
- Learn how to solve problems by decomposing them into smaller ones and not missing any cases.

2 Mandatory exercises

Exercise 1. Given a circle with center $O(x_o, y_o)$ and radius R. Write a program to check if a point $A(x, y)$ lies outside or inside the circle. The program should allow the user to:

- Enter O and R. If the user inputs a negative number for R, the program will print that it was invalid, prompt the user to press ENTER and stop.
- Enter A, perform a check to see the result.

Guidelines:

a) Data: the program needs variables to store the following data:
   - Coordinates x and y of center O: data type is double, name is either oX or OX.
   - Coordinates x and y of point A: data type is double, name is either aX or aY.
   - Radius R: double.

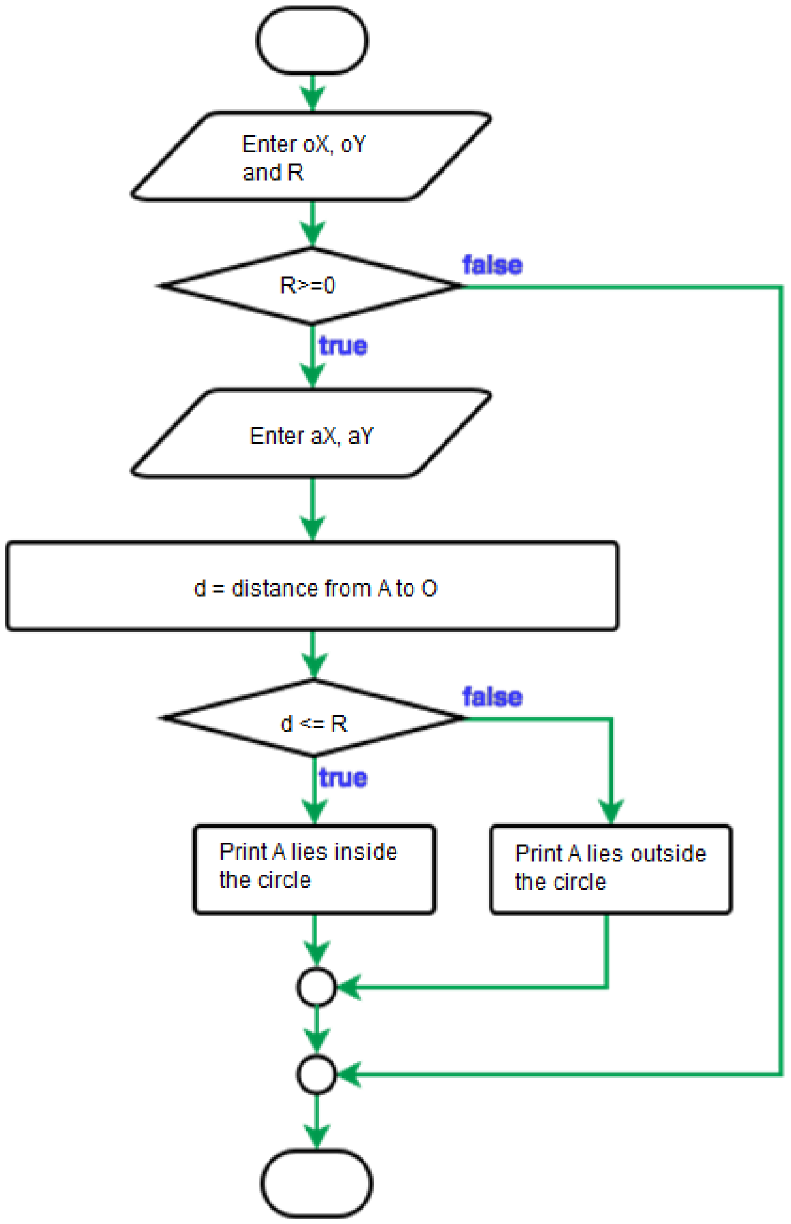
b) Algorithm:

   Point A lies inside a circle O and radius R if the distance between A and O is less than or equal to R. Otherwise, A lies outside the circle. Therefore, we need to do the following:

   ⇨ Calculate the distance between A and O, call it d.

⇨ We need the function `sqrt` to do this.
⇨ We need to include `<math.h>` to use the function.

There are only two cases to consider: $R \leq d$ and $R > d$. Here is the flow chart of the program (drawn using www.draw.io):



Exercise 2. Assume that today is Sunday, write a program to receive a number X and check what day it is after X days from today.

Guidelines:

a) Data:

- X: non-negative integer → data type should be `unsigned int` or `unsigned long long`.

b) Algorithm:

With the following observation:

• X = 0, it's the same day as today	→ Sunday.
• X = 1, it's tomorrow	→ Monday.
• X = 2	→ Tuesday.
• X = 3	→ Wednesday.
• X = 4	→ Thursday.
• X = 5	→ Friday.
• X = 6	→ Saturday.
• X = 7	→ Sunday.
• X = 8	→ Monday.
• And so on	

We can conclude that after 6 days, the day of the week will repeat (the seventh day will be the same as today):

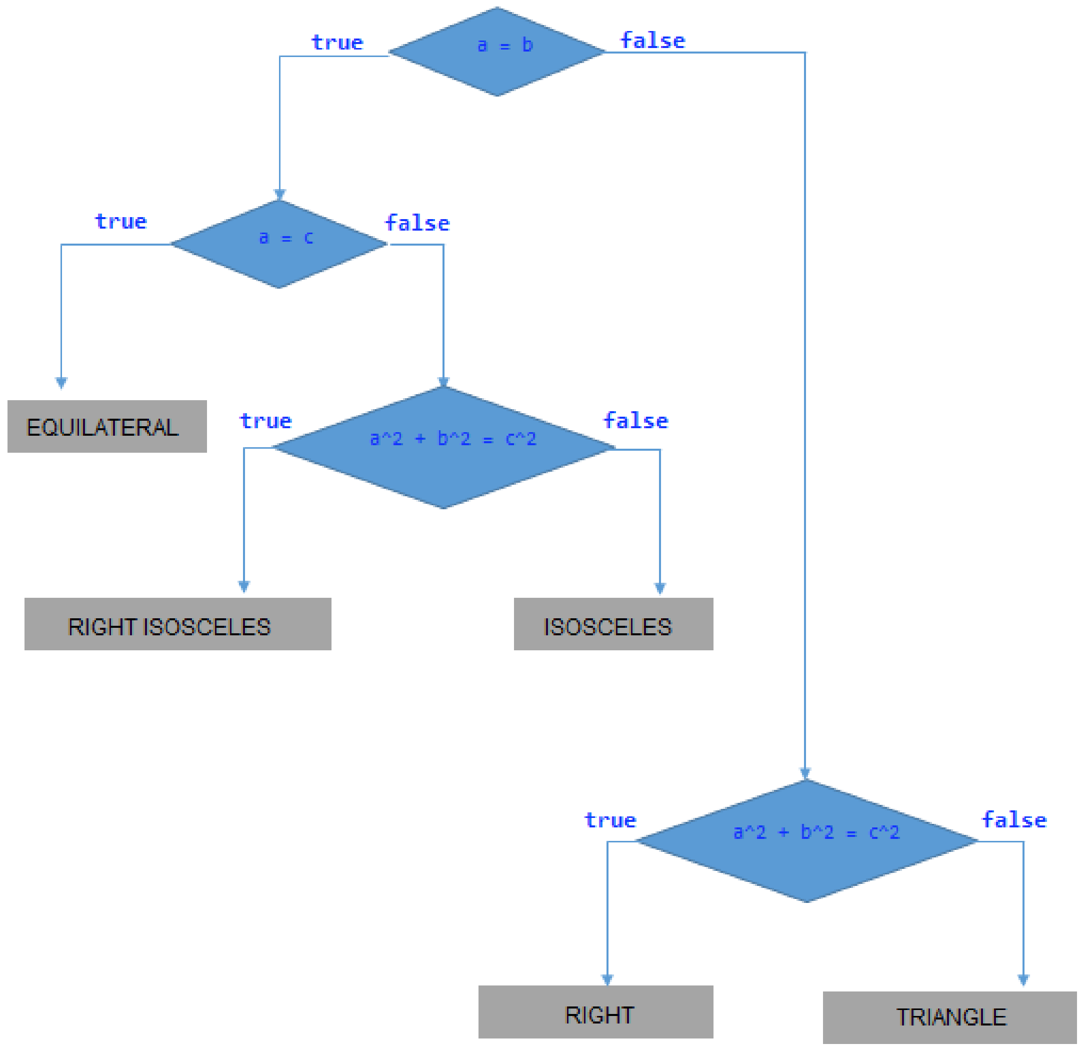
⇨ Calculate the remainder of X divided by 7 (X % 7). If the remainder is 0 then the day is Sunday, if it is 1 then the day is Monday, etc.
⇨ You can use either if...else or switch-case. Switch-case is more suitable here because the number of cases is countable.
⇨ Student should practice this problem using both if...else and switch-case.

Exercise 3. Write a program to take a, b and c and verify if they are the lengths of an isosceles triangle, an equilateral triangle, a right triangle or an isosceles right triangle.

Guidelines:

a) Data: a, b and c should be either float or double.
b) Algorithm:
   - Verify if a, b and c are lengths of a triangle or not.
   - List 3 pairs of lengths: (a, b), (b, c) and (c, a).
   - For each pair, i.e. (a, b), we must consider all possible cases.
     - First we can start checking from (a, b), if the triangle comes out as equilateral, isosceles, right isosceles or right, we don't have to check on (b, c) and (c, a) anymore (because they are mutually exclusive).
     - However, when the triangle comes out as a normal one with (a, b), we must perform further check on (b, c) and then (c, a). Note that,

when checking using (b, c) and (c, a), the triangle can't be equilateral anymore. Why?



c) The problem with real number comparison:

When working with real-valued data (float, double, long double), we **should avoid** equality comparison (==) at all costs. The reason is that our computers cannot exactly represent a real-valued number for most of the times, for example: π, $\sqrt{3}$, etc. To see this clearer, try to paste the following codes into your IDE and see for yourself:

```
#include <iostream>
#include <iomanip>
#include <math.h>
using namespace std;

void main() {
	double a = sqrt(2.0), b = 2.0;
	if (a*a == b) cout << "sqrt(2.0)*sqrt(2.0) = 2.0";
	else {
		cout << "sqrt(2.0)*sqrt(2.0) != 2.0\n";
		cout << "sqrt(2.0)*sqrt(2.0) == " << setprecision(20) << a*a << endl;
	}
}
```

However, if we have to check if a and b are equal, we mitigate the above problem somewhat by comparing (abs(a-b)) to a small enough number called EPSILON. If (abs(a-b)) is less than EPSILON, we acknowledge that a = b and vice versa.

For float numbers, we can use the following EPSILON:

```
#define EPSILON 1.0E-6
```

For double numbers, we can use the following:

```
#define EPSILON 1.0E-13
```

Note:

- Students should define macro-function to check if two numbers are equal then use the macro to perform comparison between real-valued variables and expressions (refer to the equality operators in the flow chart).
- Students should use `enum` to define all possible cases of a triangle. For instance, EQUILATERAL = 0, RIGHT = 1, etc.

Exercise 4. Write a program that allows user to enter the coefficients a, b and c of a quadratic equation: $ax^2 + bx + c = 0$. Solve the equation and print the result.

Exercise 5. Write a program to receive:

a) The date your electricity meter was read (dd:mm:yyyy) and the amount of electricity usage (in Kilowatt-hour or KWh) last month.
b) Number of households share the same electricity meter (1, 2, 3, ...).

The program will print the electricity bill using the below information:

- Cost/KWh column, consisting 6 different costs depending on the level of usage.
- Standard rate column (SR), is the maximum KWh for each level of usage.
- Customer rate column (CR), is the rate for each customer (meter). This column is computed from the standard rate column as follow:
    - CR[i] = SR[i]*(N/T)*h, i = {1, 2, 3, 4, 5, 6}.

- N: number of days since the last day the meter was read up until today (excluding today).
- T: number of days of the previous month. For example, if the latest meter read day is in March => The number of days of the previous month (Februrary) => T = 28 or 29.
- h: The number of households share the same meter.

- Usage column, is calculated from each respective "Customer rate", the values of this column must never exceed the respective "Customer rate" values.
- Cost column, is calculated from the other columns, accordingly to each level.

For example:

- Previous read date: 11/01/2016, meter read: 1000 KWh.
- Latest read date: 03/02/2016, meter read: 1236 KWh.
- Number of households: 1
  - ⇨ N = 24.
  - ⇨ T = 31 (January).

	Cost/KWh	Standard rate	Customer rate	Usage	Cost (VND)
1	1.484	50	39 = ceil(50 * (24/31) * 1)	39	57.876
2	1.533	50	39	39	59.787
3	1.786	100	77	77	137.522
4	2.242	100	77	77	172.634
5	2.503	100	77	4	10.012
6	2.587	∞	∞	0	0
				236 kWh (1236 – 100)	437.831
VAT (10%)					43.783
Total					481.614

Guidelines:

a) Data:

The program should have the following variables:

- The date the meter was read in previous month: int (or unsigned char).
- The previous month in which the meter was read: int (or unsigned char).
- The year of the previous month: int (or unsigned char).
- The date the meter was read in the latest month: int.
- The latest month in which the meter was read: int.
- The year of the latest month: int.

- Previous meter read: long long.
- Latest meter read: long long.
- Number of days from the last read to the latest read (N): int.
- Number of days of the previous month (T): int.
- Cost/KWh or standard rates can be defined by #define.

b) Algorithm:

This exercise is a typical case to use nested if-else:

- First, we need to calculate CR[1] = SR[1] * (N/T) * h, (i = 1, first row).
- If TotalUsage <= CR[1] then Usage[1] = TotalUsage, Cost[1] = Cost/KWh[1] * Usage[1]. End calculation.
- Otherwise, when TotalUsage > CR[1], Cost[1] = Cost/KWh[1] * CR[1]. Usage[2] = TotalUsage – CR[1].
- If Usage[2] <= CR[2] then Usage[2] stays the same, Cost[2] = Cost/KWh[2] * Usage[2]. End calculation.
- Otherwise, when Usage[2] > CR[2], Cost[2] = Cost/KWh[2] * CR[2]. Usage[3] = TotalUsage – CR[2].
- ...
- Continue until we reach row 6. Print the amount of money the households need to pay.

Exercise 6. Write a program to:

a) Print a menu with the following options:

```
1. Enter product
2. Find product
3. Print the list of products
4. Remove product
5. Update product
6. Save data
7. Load data
8. Quit

Please choose an option:
```

b) Read the option entered by the user. Print the chosen option and wait for the inputs from user. When the user press ENTER, check if the input is correct. If it is, proceed normally. Otherwise, print an error and prompt the user to press ENTER to end the program.

Guidelines:

a) Data type:

```
Typedef enum menu {
     ENTER_PRODUCT = 0,
     FIND_PRODUCT,
     LIST_PRODUCT,
     // ...
}
```

b) Data: the program should have a variable to store the choice of user (unsigned char).

c) Algorithm:
   - Print menu → using printf. Each line corresponds to a choice. The alignment should be left.
   - Print the line prompting the user to enter.
   - Read the choice from user.
   - Use switch-case to execute the respective choice. Use `break;` to escape the break scope.

Exercise 7. Write a program that receives an expression of a string follows the rule `value1 op value2`. Here, value1 and value 2 are two numbers, op is one of +, -, * or /. The program must evaluate and print the result of the expression. For example:

- Enter: 13 + 45.
- Print: 58.

Guidelines:

a) Data type:

Use `<string>` and `<sstream>` to work with strings.

Use `getline()` to receive the string from keyboard.

Use `stringstream` to decompose the string into smaller pieces:

- Operand_1: first operand → int
- Operand_2: second operand → int
- Operator → char
- Result → int

b) Algorithm:
   - Use switch-case to cover all valid inputted cases.

- Execute the operation in each case, assign the calculated result to the result variable.
- Print the result.