(SECJ1013) PROGRAMMING TECHNIQUE I

SEM 1, SESSION 2023/2024

LAB EXERCISE 2

SECTION 03

Name	Matric Number
NG YU HIN	A23CS0148

Lab Exercise 2

Chapter 2 Elementary Programming

Chapter 3 Control Structures

INSTRUCTIONS TO THE STUDENTS

- This exercise must be done **individually**.
- Any form of plagiarism is **NOT ALLOWED**. Students who copied other students' assignments will get **ZERO** marks (both parties, students who copied, and students who shared their work).
- Please insert your **name and matric number** as a comment in your solution.

SUBMISSION PROCEDURE

- Please submit this exercise no later than **November 23, 2023, Thursday (11.59 PM MYT)**.
- Only one file is required for the submission (the file with the extension **.pdf**).
- Submit it via the UTM's e-learning system (https://elearning.utm.my/23241/).

WRITE A C++ PROGRAM
based on the tasks below:

- Set the values:
  x1 = 1; y1 = 3; x2 = 2; y2 = 6; x3 = 5; y3 = 4;
- Find the distance between every pair of points A(1, 3), B(2, 6), and C(5, 4) using Euclidean Distance Formula.
- The output of the program:

A(1, 3), B(2, 6), and C(5, 4)

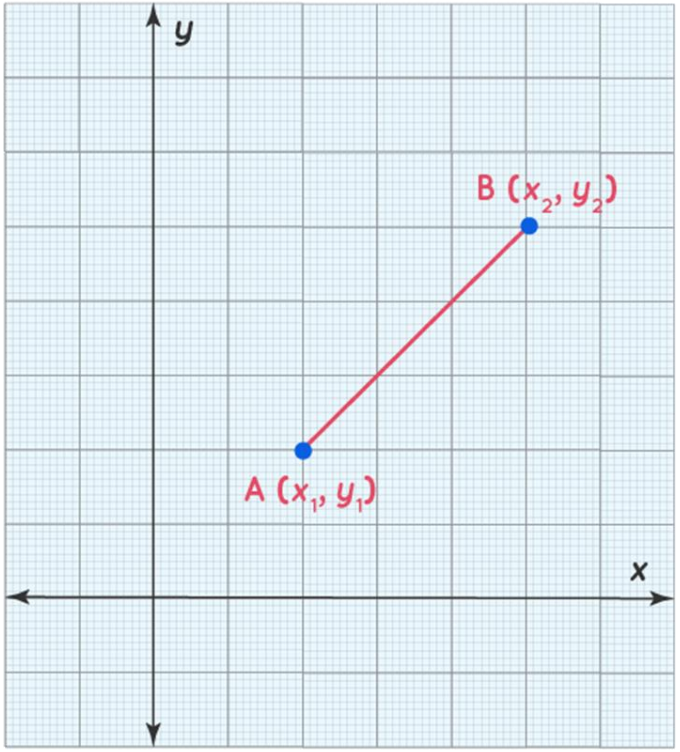
	x	y
A	1 x_1	3 y_1
B	2 x_2	6 y_2
C	5 x_3	4 y_3

AB = $\sqrt{(2-1)^2+(6-3)^2}$ = 3.1623

AC = $\sqrt{(5-1)^2+(4-3)^2}$ = 4.1231

BC = $\sqrt{(5-2)^2+(4-6)^2}$ = 3.6055

Euclidean Distance Formula



$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

```
//LAB EXERCISE 2
//NG YU HIN
//A23CS0148
#include <iostream>
using namespace std;

// Custom square root function
double Sqrt(double x) {
   double decimalplce;
   if (x < 0) {
      cout<< "Math ERROR" <<endl;
      return 0.0;
   }
   else{ decimalplce = x / 2.0;
      for (int i = 0; i < 10; ++i) {
         decimalplce = 0.5 * (decimalplce + x / decimalplce);
      }
   }
   return decimalplce;
}

int main(){

   double x1=1,x2=2,x3=5;
   double y1=3,y2=6,y3=4;

   //Display all points & their coordinates//
   cout<<"A("<<x1<<","<<y1<<"),"<<" B("<<x2<<","<<y2<<"), and"<<" C("<<x3<<","<<y3<<")"<<"\n"<<endl;
   cout<<"\t x"<<"\t y"<<endl;

   cout<<"A"<<"\t "<< x1 <<"\t "<< y1 <<endl;
   cout<<"B"<<"\t "<< x2 <<"\t "<< y2 <<endl;
   cout<<"C"<<"\t "<< x3 <<"\t "<< y3 <<"\n"<<endl;

   //Calculation of distance between every pair of points using Euclidean Distance Formula//
   double distanceAB_squared = ((x2 – x1) * (x2 – x1) + (y2 – y1) * (y2 – y1));
   double distanceAC_squared = ((x3 – x1) * (x3 – x1) + (y3 – y1) * (y3 – y1));
   double distanceBC_squared = ((x3 – x2) * (x3 – x2) + (y3 – y2) * (y3 – y2));

   double distanceAB = Sqrt(distanceAB_squared);
   double distanceAC = Sqrt(distanceAC_squared);
   double distanceBC = Sqrt(distanceBC_squared);

   cout << "AB: " << distanceAB <<  endl;
   cout << "AC: " << distanceAC << endl;
   cout << "BC: " << distanceBC << endl;

   system ("pause");
   return 0;
}
```

ERCISE 2 › lab_exercise.cpp › ...

```cpp
//LAB EXERCISE 2
//NG YU HIN
//A23CS0148
#include <iostream>
using namespace std;

// Custom square root function
double Sqrt(double x) {
    double decimalplce;
    if (x < 0) {
        cout<< "Math ERROR" <<endl;
        return 0.0;
    }
    else{ decimalplce = x / 2.0;
        for (int i = 0; i < 10; ++i) {
            decimalplce = 0.5 * (decimalplce + x / decimalplce);
            }
    }
    return decimalplce;
}

int main(){

    double x1=1,x2=2,x3=5;
    double y1=3,y2=6,y3=4;

    //Display all points & their coordinates//
    cout<<"A("<<x1<<","<<y1<<"),"<<" B("<<x2<<","<<y2<<"), and"<<" C("<<x3<<","<<y3<<")"<<"\n"<<endl;
    cout<<"\t x"<<"\t y"<<endl;

    cout<<"A"<<"\t "<< x1 <<"\t "<< y1 <<endl;
    cout<<"B"<<"\t "<< x2 <<"\t "<< y2 <<endl;
    cout<<"C"<<"\t "<< x3 <<"\t "<< y3 <<"\n"<<endl;

    //Calculation of distance between every pair of points using Euclidean Distance Formula//
    double distanceAB_squared = ((x2 - x1) * (x2 - x1) + (y2 - y1) * (y2 - y1));
    double distanceAC_squared = ((x3 - x1) * (x3 - x1) + (y3 - y1) * (y3 - y1));
```

C:\LAB EXERCISE 2\lab_exerci

```
A(1,3), B(2,6), and C(5,4)

        x       y
A       1       3
B       2       6
C       5       4

AB: 3.16228
AC: 4.12311
BC: 3.60555
Press any key to continue . . .
```