Exercise 3 of Programming Language II (CSE, NTOU) ID: _____

11:10 – 12:00, 8 May 2026; Room INS101 Name: _____

Note: Cell phones and any smart device or calculator are forbidden.

1. (15%) Complete the program so that `increase()` can add 10 to the private data member `value`

```
#include <iostream>
using namespace std;

class Score {
private:
    int value;
public:
    Score(int v = 0) : value(v) {}
    // write one line here
};

int increase(Score s) {
    s.value += 10;
    return s.value;
}

int main() {
    Score s(25);
    cout << increase(s) << endl;
    return 0;
}
```

2. (15%) **Fix** the following program and **write** a friend function `sum()` that can access private data from both classes.

```
#include <iostream>
using namespace std;


class A {
private:
    int x;
public:
    A(int v = 3) : x(v) {}
    // declare friend function here
};

class B {
private:
    int y;
public:
    B(int v = 7) : y(v) {}
    // declare friend function here
};

// write sum() here

int main() {
    A a(8);
    B b(5);
    cout << sum(a, b) << endl;
    return 0;
}
```

3. (15%) Define `operator+` and `operator<<` for the following class so that two vectors are added coordinate-wise and a vector (*x*, *y*) can be printed as "`(x, y)`".

```
#include <iostream>
using namespace std;

class Vect {
private:
    int x, y;
public:
    Vect(int a = 0, int b = 0) : x(a), y(b) {}
    // write operator+ and operator<< here
};

int main() {
    Vect o1(2, 3), o2(4, 5), o3;
    o3 = o1 + o2;
    cout << o3; // (6, 8)
    return 0;
}
```

4. (15%) Suppose that the class below already has a constructor `Vect(int)` that turns an integer into `(x,y) = (a,a)` and a member `operator+(const Vect&)`. Please check whether `line A` and `line B` in the following program can compile or not.

```
Vect o1(1, 2), o2;
o2 = o1 + 7;   // line A
o2 = 7 + o1;   // line B
```

5. (15%) Please modify the following program so that the main function works correctly.

```
#include <iostream>

class Polygon {  // modify this
  protected:
    int width, height;
  public:
    void set_values (int a, int b) { width=a; height=b; }
};
class Rectangle: public Polygon { // modify this
  public:
    int area() { return width * height; }
};
class Triangle: public Polygon { // modify this
  public:
    int area() { return width * height/2; }
};

int main () { // DO NOT MODIFY THIS MAIN FUNCTION
  double a, b;
  std::cin >> a >> b;
  Rectangle<double> rect;
  Triangle<double> trgl;
  Polygon<double>* ppoly1 = &rect;
  Polygon<double>* ppoly2 = &trgl;
  ppoly1->set_values (a,b);
  ppoly2->set_values (a,b);
  std::cout << ppoly1->area() << endl;
  std::cout << ppoly2->area() << endl;
  return 0;
}
```

6. (15%) Write a function template `sum` that can compute the sum of **zero or more elements of the same type stored in consecutive memory locations.** The function template should have **one template parameter**, namely the type of the elements to be processed. The function `sum` takes **two parameters**:
   (a). a pointer to the **first element** in the range to be processed;
   (b). a pointer to the **last element** in the range to be processed.

   Please complete the function template so that the following `main` function works correctly

```
int main() {   // Do NOT modify main()
    int i[3] = {1, 2, 3};
    double d[5] = {1.0, 2.0, 3.0, 4.0, 5.0};

    std::cout << sum<int>(&i[0], &i[2]) << endl;
    std::cout << sum<double>(&d[0], &d[4]) << endl;

    return 0;
}
```

7. (15%) What happens when the following program is compiled? Explain the reasons of your answers.

```
#include <iostream>
using namespace std;

class MyClass {
private:
    MyClass() {}
};

int main() {
    MyClass obj;
    return 0;
}
```

8. (15%) Please modify the line in bold face by using smart pointers.

```
class Logger {
private:
    Logger() {
        cout << "[Logger created]" << endl;
    }
    Logger(const Logger&) = delete;
    Logger& operator=(const Logger&) = delete;
    static Logger* instance;

public:
    static Logger* getInstance() {
        if (instance == nullptr) {
            instance = new Logger();  // Lazy initialization
        }
        return instance;
    }
    void log(const string& message) {
        cout << "[LOG]: " << message << endl;
    }
};
```

9. (15%) Please correct the following class declaration it has errors.

```
class Vect {
     int x, y;
public:
     Vect() = default;
     ~Vect() = default;
     friend ostream& operator<<(ostream& os, const Vect r);
     friend istream& operator>>(istream& is, const Vect r);
};
```