Vietnamese German University
Computer Science Program

Practice Test

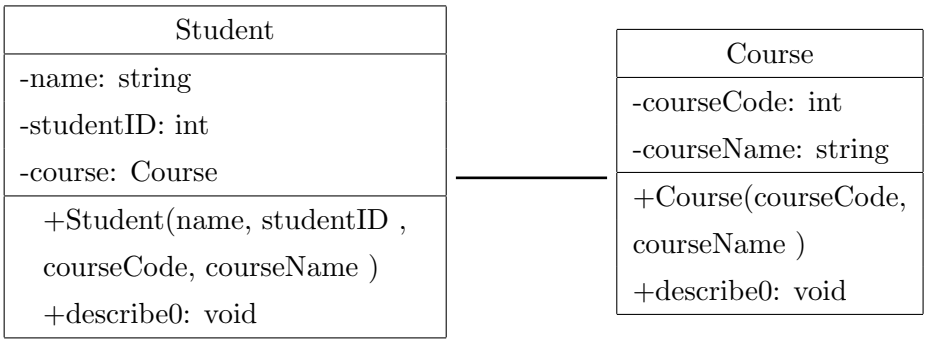
Course: Object Oriented Programming - C++
Duration: 100 minutes

Submit your code (.cpp files) to the classroom
During the exam, only a C++ editor is opened (run).
Do not open any kind of media, such as Facebook, Skype, etc., during the exam.

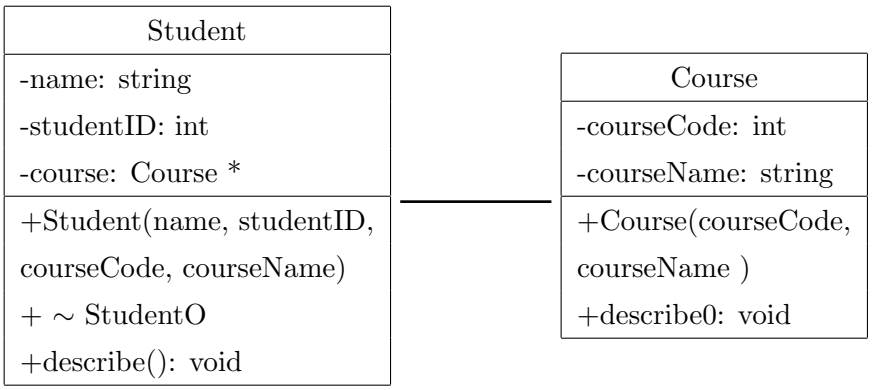
---

1. Implement the following class diagrams. Please note that you need to implement exactly what is described in the class diagrams. You cannot create a constructor with a different number of arguments or something like that.

a) Class diagram 1



b) Class diagram 2



Test sample for both (a) and (b)

```cpp
int main() {
    //Create a student with
    //name = "David" and Student ID = 101
    //Course code = 111 and name = "C++"

    return 0;
}

//OUTPUT
//Student name: David - Student ID: 101
//Course code: 111 - Course name: C++
```

Code Listing 1: Input, Output Example

2. Write a program to create a class called "Product" which is used to represent the properties of a product. The class will have three private float data members (price, weight, and discount). Implement the following member methods:

a) A default constructor (set the value of data members to zero).

b) A constructor with parameters.

c) A copy constructor.

d) An overloaded plus operator (+), that adds the price, weight, and discount of two products and returns a new product with the updated values.

e) Overload assignment operator (=), that assigns the price, weight, and discount of one product to another.

f) Overloaded comparison operator (==), that compares two products. Two products are considered equal if their price, weight, and discount are equal.

Note: You need to implement the describe() method to display the information about the products.

Product
- price: float - weight: float - discount: float
+Product() +Product(price, weight, discount) +Product(other_product) +describe():void +operator+(other_product):Product +operator=(other_product):void +operator==(other_product):bool

```
int main() {
    //Create Product m0 with default contructor
    m0.describe();

    //Create Product m1 with price = 10.5
    //weight = 2.3, discount = 1.2
    m1.describe();

    //Create Product m2 with price = 15
    //weight = 2.2, discount = 0.5
    m2.describe();

    //Create Product m3 using copy contructor
    // copy m1 into m3
    m3.describe();

    //overloaded plus operator (+)
    //m3 = m1 + m2;
    m3.describe();

    cout << "Is m3 = m1? " << (m3 == m1) << endl;

    //assign m1 to m3
    //Is m3 == m1?
    m3 = m1;
    m3.describe();

    cout << "Is m3 = m1? " << (m3 == m1) << endl;
    return 0;
}

//OUTPUT
```

```
/*
Price: 0 Weight: 0 Discount: 0
Price: 10.5 Weight: 2.3 Discount: 1.2
Price: 15 Weight: 2.2 Discount: 0.5
Price: 10.5 Weight: 2.3 Discount: 1.2
Price: 25.5 Weight: 4.5 Discount: 1.7
Is m3 = m1? 0
Price: 10.5 Weight: 2.3 Discount: 1.2
Is m3 = m1? 1
*/
```

Code Listing 2: Input, Output Example

3. A Society consists of a name (string) and a list of members. A member of this system can be a participant, a mentor, or a coordinator. A participant comprises a name (string), year of birth (int), and a field (string). A mentor consists of a name (string), year of birth (int), and a subject (string). A coordinator includes a name (string), year of birth (int), and a division (string). You need to use a vector to store a list of members.

a) Implement classes for the above description. Implement the describe() method to print out the content of the objects for all classes.

b) Write the addMember(Member* member) method in the Society class to add a new member to the society (list of members). Create a society object, then add one participant, two mentors, and two coordinators. Implement the describe() method to print out the society name and the information of members in the society.

c) Write the countCoordinators() method to count the number of coordinators in society.

d) Write the sortBirthYear() method to sort members in the society by their year of birth in increasing order. Please remember to include $< algorithm >$ library.

e) Write the aveMentorYearOfBirth() method to calculate the average year of birth of mentors.

```
int main () {
   // Create a participant with
   // name = "Alice", yearOfBirth = 1995, field = "Art"
   Participant participant("Alice", 1995, "Art");
   participant.describe();

   // Create a mentor1 with
   // name = "Bob", yearOfBirth = 1980, subject = "Math"
   Mentor mentor1("Bob", 1980, "Math");
   mentor1.describe();
```

```
11
12      // Create a mentor2 with
13      // name = "Carol", yearOfBirth = 1970, subject = "Science"
14      Mentor mentor2("Carol", 1970, "Science");
15      mentor2.describe();
16
17      // Create a Coordinator1 with
18      // name = "David", yearOfBirth = 1985, division = "Events"
19      Coordinator  coordinator1("David", 1985, "Events");
20
21      // Create a coordinator2 with
22      // name = "Thomas", yearOfBirth = 1980, division = "Logistics"
23      Coordinator  coordinator2("Thomas", 1980, "Logistics");
24
25      // Create a society with name = "society1"
26      Society society("society1");
27      // add 1 participant, 2 mentor, 2 coordinator
28      society.addMember(&participant);
29      society.addMember(&mentor1);
30      society.addMember(&mentor2);
31      society.addMember(&coordinator1);
32      society.addMember(&coordinator2);
33
34      // describe
35      society.describe();
36
37      // Use countCoordinator() method to count
38      // number of people and store result
39      // in numberOfCoordinators variable
40      int numberOfCoordinators = society.countCoordinator();
41      cout << "numberOfCoordinators: " << numberOfCoordinators << endl;
42
43      //Sort people in the society by their age
44      //use sortAge() method to sort
45      society.describe();
46
47      // Use aveMentorYearOfBirth() method to calculate average year of birth of Mentors
48      //and store the result
49      //in aveMentorYearOfBirth variable
50      cout << "Average year of birth(Mentors): " << aveMentorYearOfBirth << endl;
51
52      return 0;
53  }
54
55  //OUTPUT
56  /*
```

```
Participant - Name: Alice - yearOfBirth: 1995 - Field: Art
Mentor - Name: Bob - yearOfBirth: 1980 - Subject: Math
Mentor - Name: Carol - yearOfBirth: 1970 - Subject: Science
Coordinator  - Name: David - yearOfBirth: 1985 - Division : Events
Coordinator  - Name: Thomas - yearOfBirth: 1980 - Division : Logistics
Society Name: society1
Participant - Name: Alice - yearOfBirth: 1995 - Field: Art
Mentor - Name: Bob - yearOfBirth: 1980 - Subject: Math
Mentor - Name: Carol - yearOfBirth: 1970 - Subject: Science
Coordinator  - Name: David - yearOfBirth: 1985 - Division : Events
Coordinator  - Name: Thomas - yearOfBirth: 1980 - Division : Logistics
numberOfCoordinators: 2
Society Name: society1
Participant - Name: Alice - yearOfBirth: 1995 - Field: Art
Coordinator  - Name: David - yearOfBirth: 1985 - Division : Events
Mentor - Name: Bob - yearOfBirth: 1980 - Subject: Math
Coordinator  - Name: Thomas - yearOfBirth: 1980 - Division : Logistics
Mentor - Name: Carol - yearOfBirth: 1970 - Subject: Science
Average year of birth(Mentors): 1975
*/
```

Code Listing 3: Input, Output Example