SECTION A: STRUCTURED QUESTIONS [70 MARKS]

Question 1 **[11 marks]**

(a) Complete **`Program 1`** based on questions (i), (ii) and (iii) as commentaries in the given code.

```
// Program 1
#include <iostream>
using namespace std;

class MyClass {
public:
  int x;
  int y;

  //(i) Define an overloaded default constructor.
  ________________________ { ________________________ } //1 mark

  //(ii) Define a constructor with one parameter.

  ________________________ { ________________________ } //1 mark

  //(iii) Define a constructor with two parameters.

  ________________________ { ________________________ } //1 mark
};

int main()
{
  MyClass t;
  MyClass t1(5);
  MyClass t2(9, 10);
  cout << "t.x: " << t.x << ", t.y: " << t.y << "\n";
  cout << "t1.x: " << t1.x << ", t1.y: " << t1.y << "\n";
  cout << "t2.x: " << t2.x << ", t2.y: " << t2.y << "\n";

  return 0;
}


```

(iv) What is the output printed by **`Program 1.`**

Answer: (3 marks)

(b) Complete **`Program 2`** based on questions (i), (ii) and (iii) stated as commentaries in the given code.

```
// Program 2
#include <iostream>
using namespace std;

class Point
{
private:
    int x, y;
public:
    Point(int x1, int y1)
    {   x = x1; y = y1;   }

    // (i) Define a copy constructor to inialize the another
    //     object's data

    ___________________________________________

    { ________________________ ;

      ________________________ ;
    }

    int getX() {  return x;  }
    int getY() {  return y;  }
};

int main()
{
    //(ii) Call the constructor to create an object, p1 and
    //     inital the parameters with the value 10 and 15.

    _____________________________________;

    //(iii) Call copy constructor to create an object p2, and
    //      assign p2 to p1 object

    _____________________________________;

    // Accessing values assigned by constructors
    cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY();
    cout << "\np2.x = " << p2.getX() << ", p2.y = " << p2.getY();
    return 0;
}
```

(iv) What is the output printed by **`Program 2.`**

Answer:	(2 marks)

Question 2 **[16 marks]**

Consider the following **Game** class definition. Answer questions (a), (b) and (c) below.

```
class Game
{
  private:
    int val;
    static int game;

  public:
    Game(int v) : val(v)  // constructor with default parameters
    { }
    void setVal(int v)
    {   val=v;  }
    void prntVal() const ;   // Function to output val
    static void prntGame();  // Function to output game
    static void increGame(); // Function to increment game++;
} ;

```

(a) Write a C++ statement that initializes the member variable **game** to **0**. (1 mark)

(b) Define all the member functions of **Game** as specified in the class definition above. (3 marks)

(c) What is the output if the following **main()** function which relates to the **Game** class as defined above? (7 marks)

```
int main()
{
    Game g1(11), g2(67);
    g1.prntGame();
    g1.increGame();
    g2.prntGame();
    g1.prntVal();
    g2.prntVal();
    Game::increGame();
    g1.prntGame();
    g2.prntGame();
    g2.increGame();
    Game::prntGame();

    return 0;
}
```

(d) Study **`Program 3`** given below and answer the questions as commented in (i), (ii) and (iii) in the program.

```
// Program 3
#include <iostream>
using namespace std;

//(i) forward declaration: Declare a class named Second

_____________________________________ //1 mark

class First {
    int data;

  public:
    First(): data(12){ }
    //(ii) Declare a friend function

    _________________________________ //1 mark
    int data;

  public:
    Second(): data(1){ }
    //(iii) Declare a friend function

    _________________________________ //1 mark
};

int func(First d1,Second d2)
/* Function func() is the friend function of both classes
   First and Second. The private data of both class can
   be accessed from this function. */
{   return (d1.data+d2.data);   }

int main()
{
    First a;
    Second b;
    cout << "Data: " << func(a,b);
    return 0;
}
```

(iv) What is the output printed by **`Program 3`**? (2 marks)

Question 3 **[12 marks]**

Given the following class **`Rect`** that calculates the **`area`** and the **`perimeter`** of a rectangle. Answer questions (a), (b) and (c) below.

```
class Rect {
  private:
    double width,length;

  public:
    Rect ();
    double area, perimeter;
    void setRect();   // function to set width & length
                      // from keyboard input
    void calcArea();  // calculate area
    void calcPeri();  // calculate perimeter
    void print();     // print all attributes in the class
};
```

(a) Write an overloaded **`operator ==`** function using the following function definition. This function returns value **`true`** if the area and perimeter of two rectangles are the same. (2 marks)

```
bool operator == (rect rec1,  rect rec2)
```

(b) If variables **`area`** and **`perimeter`** are defined as **`private`** attributes, explain how the function overloaded **`operator ==`** can be implemented. (1 mark)

(c) Write the **`main`** function which defines an array of object **`Rect`** called **`recs[]`** with the size of 4. The function must abide to the following rules:

(i) for each element of **`recs`**, set the values of **`width`** and **`length`**, and calculate **`area`** and **`perimeter`** (4 marks)

(ii) check which rectangles are of the same size using overloaded **`operator ==`** and print a message to the screen if their sizes are the same. Example of the screen output if **`recs[0]`**, **`recs[1]`** and **`recs[2]`** are the same is as followed: (5 marks)

```
Same sized rectangles at index 0 and 2
Same sized rectangles at index 0 and 1
Same sized rectangles at index 1 and 2
```

Question 4 **[12 marks]**

Given **`Program 4`** as shown below. Answer questions (a) and (b) following it.

```cpp
// Program 4
#include <string>
#include <iostream>
using namespace std;

class Name
{
  private:
    string firstname, lastname;

  public:
    Name(string fname, string lname)
    {
      firstname = fname;
      lastname = lname;
    }

    string getFullName()
    {
      return firstname + " " + lastname;
    }
};

class Lecturer
{
  private:
    Name name;
    string staffId;

  public:
    Lecturer(string fname, string lname, string sId):
    name(fname, lname)
    {
      staffId = sId;
    }

    string getLecturer()
    {
      return name.getFullName() + "\nLecturer id  : " +
             staffId;
    }
};

class Department
{
  private:
    Lecturer *lecturerDepart;

  public:
    Department(Lecturer *lectDepart)
    {
      lecturerDepart = lectDepart;
    }

    void printDepartment()
    {
      cout << "Lecturer name: " <<
           lecturerDepart->getLecturer() << endl;
    }
};

int main()
{
  Lecturer *lect = new Lecturer("Abdullah", "Hamid", "124");
  Department department(lect);
  department.printDepartment();

  return 0;
}
```

(a) Based on **`Program 4`** given above, draw the UML class diagram that shows the relationship between classes. (5 marks)

(b) What is the output of **`Program 4`**? (2 marks)

(c) Write the class definitions that contain aggregation relationship for the following classes as depicted in **Figure 1**. (5 marks)

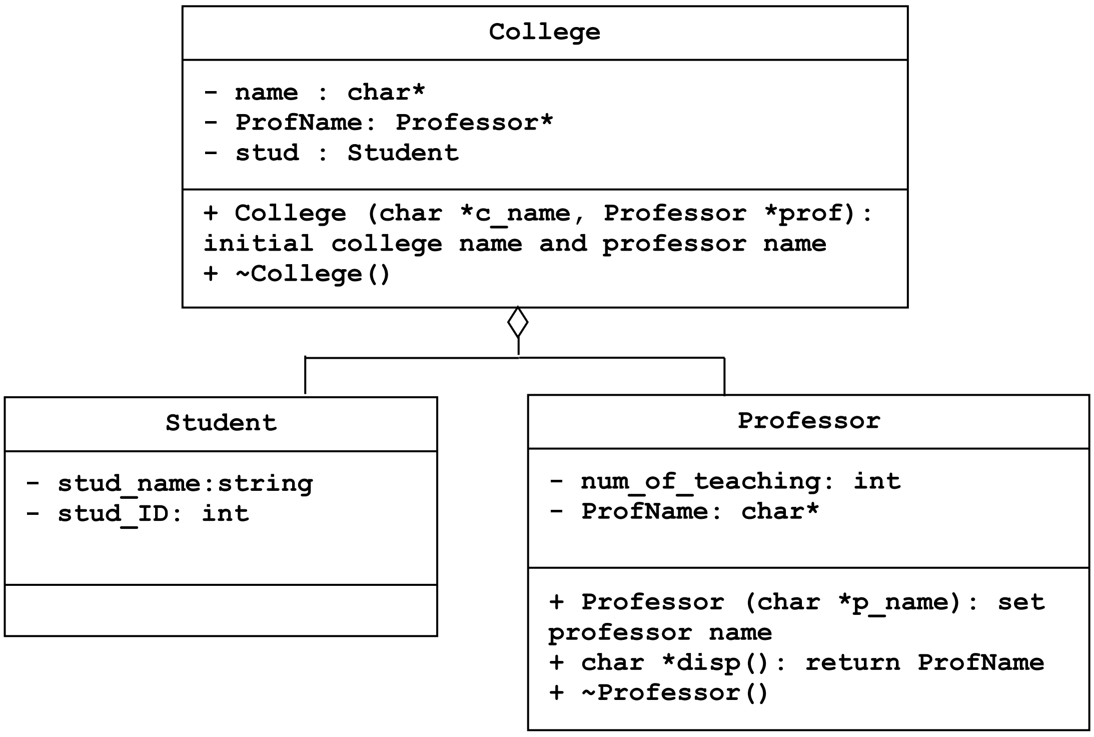

Figure 1

Question 5 **[19 marks]**

Given three classes named **`Automobile`**, **`Car`** and **`Truck`** respectively, and some objects of these classes as shown in **`Program 5`**, answer questions (a) to (c).

```
// Program 5
#include <iomanip>
#include <iostream>
using namespace std;

class Automobile
{
  private:
    string make;

  protected:
    int model;

  public:
    double price;

    Automobile()
    {
      make = "";
      model = 0;
      price = 0.0;
      cout << "An Automobile object has been created but "
           << "not yet have details." << endl;
    }

    Automobile(string a, int b, double c)
    {
      make = a;
      model = b;
      price = c;
      cout << "Automobile object: " << make << " makes in "
           << model << ". The price is RM " << price << "."
           << endl;
    }

    void printInfo(){ }
}; //End class Automobile

class Car: private Automobile
{
  private:
    int doors;

  public:
    Car(): Automobile("BMW", 2010, 150000.0)
    {
      doors = 0;
48        cout << "Car with car's make, year model and price "
49             << "(\"BMW\", 2010, 150000.0) accordingly."
50             << endl;
51      }
52  
53      Car(string a, int b, double c, int d): Automobile(a, b,
54      c)
55      {
56        doors = d;
57        cout << a << "'s car has " << d << " doors." << endl;
58      }
59  
60      void printCar() { }
61  }; //End class Car
62  
63  
64  class Truck: protected Car
65  {
67    protected:
68      string driveType;
69  
70    public:
71      Truck(string d): Car("Toyota", 2014, 45000.0, 4)
72      {
73        driveType = d;
74        cout << "This is the truck with " << d
75             << " drive type." << endl;
76      }
77  
78      void printTruck() { }
79  }; //End class Truck
80  
81  
82  int main()
83  {
84      cout << fixed << setprecision(2);
85  
86      Car car;
87      Truck truck("4WD");
88      Automobile automobile;
89  
90      return 0;
91  }
```

(a) What will the lines 86 to 88 print onto the screen when the program runs? State your answer by completing the following table:

Line	**Statement**	**Output**
86	`Car car;`	

		(2 marks)
87	`Truck truck("4WD");`	(3 marks)
88	`Automobile automobile;`	(1 marks)

(b) Based on the objects created in the **`main`** function of the program (line 86 to 88), determine the member variables that each object owns. Write (**Own**) in the corresponding blank cells, if the object own or has a copy of the variable; and write (**No**) if otherwise. State your answer by completing the following table: (6 marks)

	Member Variables				
Object	**`make`**	**`model`**	**`price`**	**`doors`**	**`driveType`**
`car`	(i)	(ii)	(iii)	(iv)	(v)
`truck`	(vi)	(vii)	(viii)	(ix)	(x)
`automobile`	■	■	■	(xi)	(xii)

(c) With reference to the same member variables in (a), determine whether they are accessible inside the methods and by the object listed below. Write (**Accessible**) in the corresponding blank cells, if it is accessible; and write (**No**) if otherwise. State your answer by completing the following table:

(7 marks)

	Member Variables				
Object	**`make`**	**`model`**	**`price`**	**`doors`**	**`driveType`**
`printInfo()`	(i)	■	■	(ii)	(iii)
`printCar()`	(iv)	(v)	■	■	(vi)
`printTruck()`	■	(vii)	(viii)	(ix)	(x)
`automobile`	(xi)	(xii)	(xiii)	(xiv)	■

SECTION B: PROGRAMMING QUESTION TOTAL – 30 MARKS

This section consists of **ONE (1)** question only.

Question

There are two types of courses offered at the Faculty of Computing of Universiti Teknologi Malaysia, lectured and non-lectured. A lectured course is a regular course which has a lecturer teaching the class; whereas a non-lectured course does not require a lecturer, for example final year undergraduate projects, and internship courses. Besides that, a non-lectured course has a prerequisite minimum credit hour, indicating that the student needs to earn at least the minimum credit hour before he or she can enrol for the course. Each course, regardless lectured or non-lectured, is assigned with a code and name.

(a) Draw the UML class diagram for the above problem. Your design should include the classes and their attributes and methods as specified in Table B1; as well as relationships between the classes. Then, write the C++ code to implement the design. Your implementation should apply the concept of data hiding in which all the attributes are inaccessible from outside of the class.

(7 marks)

(b) Finally, use the classes to create a program that helps the faculty to manage two types of courses, subject-based and project-based courses. You need to create two arrays named `subjects` which is of type `LecturedCourse` and `projects` of type `NonLecturedCourse`, respectively. Also, the program should provide the user a menu-driven interaction with the options as described in Table B2. The program repeats until the user chooses to exit. Refer to Figure B1 for the example run of the program.

(23 marks)

Table B1: The classes

Class	Attribute	Method
`Course`	The course's code and name.	• The method `getName` that accesses the course's name.
`LecturedCourse`	The lecturer as well as the course's code and name.	• The method `getLecturer` that accesses the course's lecturer. • The method `setLecturer` that sets the lecturer for the course. • The method `read` that reads in input from the keyboard for the attributes.
`NonLecturedCourse`	The minimum credit hours as well as the course's code and name.	• The method `getMinCredit` that accesses the course's pre-requisite minimum credit hour. • The method `read` that reads in input from the keyboard for the attributes.
`Lecturer`	The lecturer's name.	• The method `getName` that accesses the lecturer's name. • The method `setName` that sets the lecturer's name.

Table B2: The menu of the program

Menu Options	Description
1. Add subject	To insert a new subject course into the array `subjects.`
2. Add project	To insert a new project course into the array `projects.`
3. Print subjects	To print the subject's name and the lecturer for all the subject courses.
4. Print projects	To print the subject's name and the minimum credit hours for all the project courses.
5. Exit	To end the program.

```
--------- Menu------------
1. Add subject
2. Add project
3. Print subjects
4. Print projects
5. Exit

Enter your choice => 1

Adding a new subject course:
Code => SCSJ1013
Name => Programming Technique I
Lecturer => Dr Muhammad Razali


--------- Menu------------
1. Add subject
2. Add project
3. Print subjects
4. Print projects
5. Exit

Enter your choice => 1

Adding a new subject course:
Code => SCSJ1023
Name => Programming Technique II
Lecturer => Mr Abdul Hakim Abdullah


--------- Menu------------
1. Add subject
2. Add project
3. Print subjects
4. Print projects
5. Exit

Enter your choice => 2

Adding a new project course:
Code => SCSV3032
Name => Graphics and Multimedia Software Project I
Min Credit => 80

--------- Menu------------
1. Add subject
2. Add project
3. Print subjects
4. Print projects
5. Exit

Enter your choice => 2
```

```
Adding a new project course:
Code => SCSJ3032
Name => Software Engineering Project I
Min Credit => 80


--------- Menu------------
1. Add subject
2. Add project
3. Print subjects
4. Print projects
5. Exit

Enter your choice => 2

Adding a new project course:
Code => SCSV4044
Name => Graphics and Multimedia Software Project II
Min Credit => 90



--------- Menu------------
1. Add subject
2. Add project
3. Print subjects
4. Print projects
5. Exit

Enter your choice => 3

The list of subject courses:
Course Name                Lecturer

Programming Technique I    Dr Muhammad Razali
Programming Technique II   Mr Abdul Hakim Abdullah


--------- Menu------------
1. Add subject
2. Add project
3. Print subjects
4. Print projects
5. Exit

Enter your choice => 4

The list of project courses:
Course Name                                 Min Credit Hour
Graphics and Multimedia Software Project I       80
Software Engineering Project I                   80
Graphics and Multimedia Software Project II      90


--------- Menu------------
```

```
1. Add subject
2. Add project
3. Print subjects
4. Print projects
5. Exit

Enter your choice => 5

Program ends!
```

Figure B1: An example run of the program. Note that, the bold texts indicate input entered by the user.