CONFIDENTIAL

UNIVERSITI TEKNOLOGI MALAYSIA

MID TERM TEST

SEMESTER II 2017/2018

SUBJECT CODE	**: SCSJ10123**
SUBJECT NAME	**: PROGRAMMING TECHNIQUE II**
YEAR/COURSE	**: 1 (SCSB/SCSJ /SCSP/SCSR / SCSV)** **2 (SCSR/SCSV)**
TIME	**: 9:00 AM – 11:00 AM (2 HOURS)**
DATE	**: 30 MARCH 2018**
VENUE	**: BK1-7, N28, FC**

INSTRUCTIONS:

SECTION A: THREE (3) STRUCTURED QUESTIONS	(60 MARKS)
SECTION B: ONE (1) PROGRAMMING QUESTION	(40 MARKS)
TOTAL	(100 MARKS)

ANSWER ALL QUESTIONS IN THIS BOOKLET AT THE SPACES PROVIDED.

Name	
I/C No.	
Year/Course	
Section	
Lecturer's Name	

(This question booklet consists of 19 pages including this page.)

SECTION A: STRUCTURED QUESTIONS [60 MARKS]

Question 1 **[15 Marks]**

a. Object-Oriented Programming (OOP) comes with many principles. Choose only FOUR (4) of them and briefly describe the key concept of each principle with an appropriate example. (10 Marks)

Answer space:

b. Design a UML class diagram to model an employee of a company. Each employee has a name and salary rate (given as per hour). You must use proper notations for each element of the design including notations for private and public members and data types for each member data (attribute) and parameter. The class should also include the following operations:

- A default constructor.
- An appropriate overloaded constructor.
- A mutator for each attribute.
- An accessor for each attribute.
- A method to calculate the total salary earned. The method will take the total hours of work as a parameter and return the total salary.

(5 marks)

Answer space:

Question 2 **[25 Marks]**

a. Consider the class `Data` and functions `increaseDataValue` and `decreaseDataValue` in **Program 1** below.

```
 1  // Program 1
 2
 3  #include<iostream>
 4  using namespace std;
 5
 6  class Data{
 7        private:
 8              int value;
 9              int *link;
10
11        public:
12              Data( int _value, int * _link){
13                    value = _value;
14                    link = _link;
15              }
16
17              Data(const Data &data){
18                    value = data.value;
19                    link  = data.link;
20              }
21
22              ~Data(){
23                  cout << "Data with value " << value
24                       << " is being destroyed" << endl;
25              }
26
27              int getValue() const { return value;}
28
29              void setValue(int _value) { value =_value;}
30
31              void setLinkContent(int number) { *link = number;}
32
33              void print() const{
34                 cout << "value contains "<< value <<endl;
35                 cout << "link points to "<< *link <<endl<< endl;
36              }
37  };// End of class
38
39  void increaseDataValue(Data data){
40        int val = data.getValue();
41        val = val + 10;
42        data.setValue(val);
43  }
44
45  void decreaseDataValue(Data& data){
46        data.setValue( data.getValue() - 5 );
47  }
```

i. Assume the main function of the program is as below. What is the output of the program as printed by each of the following lines? (4 marks)

```
51  int main(){
52       int number = 5;
53       Data data1(100, &number);
54       data1.print();
55
56       Data data2 = data1;
57       data2.print();
58
59       data1.setLinkContent(9);
60       data2.setLinkContent(11);
61       data2.setValue(88);
62
63       data1.print();
64
65       data2.print();
66
67       return 0;
68  }
```

Answer space:

Line	Output
54	
57	
63	
65	

ii. If the main function is changed to as follows, determine the value of variables **a**, **b** and **c**. (3 marks)

```
51  int main(){
52       int a, b, c;
53
54       Data data(100, NULL);
55       a = data.getValue();
56
57       increaseDataValue(data);
```

```
58        b = data.getValue();
59
60        decreaseDataValue(data);
61        c = data.getValue();
62
63        return 0;
64  }
```

Answer space:

a = _____ b =_____ c =_____

iii. If the main function is changed to as follows, determine the sequence of the destruction of the objects by writing the output printed from the destructor in order.

(3 marks)

```
51  int main(){
52        int x = 777;
53
54        Data *ptrData;
55        Data data(1,&x);
56
57        ptrData = new Data(7, &x);
58
59        if ( ptrData->getValue() > data.getValue() ){
60              Data subData(11,&x);
61              // do nothing here
62        }
63
64        delete ptrData;
65
66        return 0;
67  }
```

Answer space:

b. A rectangle can be represented by its bottom-left corner point (x_1, y_1) and top-right corner point (x_2, y_2), as shown in Figure 1. The width and height are then determined from these coordinates.

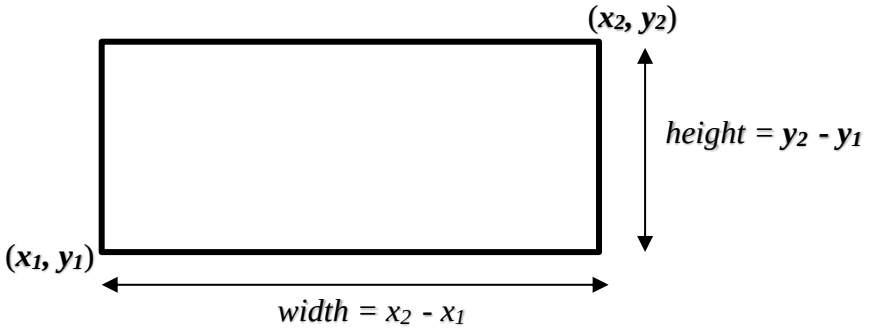

Figure 1

Complete **Program 2** below based on the tasks **(i)** to **(viii)**. Write your answer in the program.

Answer space:

```
// Program 2

#include<iostream>
using namespace std;

class Rectangle{
  private:
      int x1,y1;    // bottom-left corner
      int x2,y2;    // top-right corner
  public:
      // The default constructor
      Rectangle(){ x1=y1=x2=y2=0; }

      // The constructor that sets the bottom-left and top-right corners
      Rectangle(int a, int b, int c, int d){
            x1=a; y1=b;
            x2=c; y2=d;
      }

      // (i). Define a constructor to create a rectangle with the bottom-left corner (0,0) and the
      //      width and height set as w and h, respectively.          ( 1 mark)
```

```
        // (ii). Define a constructor to create a square with the bottom-left corner (x, y),
        //       and the side length of s.                                    (1.5 marks)









        // (iii). Define a member function (method) that determines the width of the rectangle.
        // /                                                                  (1.5 marks)









        // (iv). Define a member function (method) that determines the height of the rectangle.
        //                                                                    (1.5 marks)









        friend void inputRectangle(Rectangle&);

};  // End of class



// (v). Write the definition code for the function inputRectangle to read the coordinates of a
//       rectangle from the keyboard.                                        (1.5 marks)
```

```
int main()
{
   // (vi). Declare an array to hold 4 objects of Rectangle, and initialize the array as follow:
   //       the first element is a rectangle    with bottom-left and top-right corners of (1,2) and
   //       (3,4), respectively.
   //
   //       the second element is a rectangle with bottom-left corner at (0,0), and width is 5 and height
   //       is 10 .
   //
   //       the third element is a square with bottom-left corner at (3,4), and the side length is 5
   //                                                                                    (2 marks)
```

```
   // (vii). Read the data for the fourth element of the array from the keyboard.      (1 mark)
```

```
   // (viii). Using a loop, calculate the total area of the rectangles and print the result.   (5 marks)
```

```
      return 0;
}
```

Question 3 **[20 Marks]**

Given **Program 3** below which is intended to read a list of cars from a binary file and calculate the total price of all the cars. The user will need to enter the file's name that contains the list of cars. Write the correct C++ code to perform the following tasks. Write your answer in Table 1.

a. Specify an appropriate parameter for the method **`fileCheck.`** This method will take a file as its parameter. (1 mark)

b. Write the condition to check whether the file is opened successfully. (1 mark)

c. The file's name used in your computer system is in small letters. However, the user might be entering the name in capital letters or mixed cases. Thus, convert the file's name to small letters. (3 marks)

d. Open the file as a binary input file. (2 marks)

e. Invoke the method **`fileCheck`** to check the input file has been opened successfully. (2 marks)

f. Determine the size of the input file in bytes. (3 marks)

g. Determine the size of each `Car` object. (1 mark)

h. Determine how many cars in the input file. (1 mark)

i. Read all the `Car` objects from the input file into the array `cars` (4 marks)

j. Calculate the total price. (2 marks)

```
 1  //Program 3
 2
 3  #include <iostream>
 4  #include <cstdlib>
 5  #include <fstream>
 6  using namespace std;
 7
 8  class Car{
 9     private:
10        int year;
11        double price;
12
13     public:
14        Car(int year=0, double price=0.0){
15           this->year = year;
16           this->price = price;
17        }
18
19        double getPrice() const{ return price; }
20
21        static void fileCheck(   (a)   ){
22
```

```
23          if (    (b)    ) {
24             cout << "Error! File not found..";
25             exit(1);
26          }
27       }
28 }; // End of class Car
29
30 int main()
31 {
32     Car cars[100];       // To store the list of cars read from the file
33     fstream fin;
34     string fileName;
35     int fileSize;        // The size of file (number of bytes)
36     int objectSize;      // The size of each Car object (in number of bytes)
37     int nCars;           // The number of Car objects in the file.
38
39     cout << "Enter the input file's name => ";
40     cin >> fileName;
41
42     _________________(c)_____________
43     ____________________________________
44
45
46     _________________(d)_____________
47
48     _________________(e)_____________
49
50     _________________(f)_____________
51     ____________________________________
52
53
54     _________________(g)_____________
55
56     _________________(h)_____________
57
58     _________________(i)_____________
59     ____________________________________
60
61     fin.close();
62
63     double totalPrice = 0;
64     _________________(j)_____________
65     ____________________________________
66
67     cout << "Total car price: " << totalPrice << endl;
68
69    return 0;
70 }
71
```

.

Answer space:

Table 1

Question	Answer (C++ Code)
(a)	
(b)	
(c)	
(d)	
(e)	
(f)	
(g)	
(h)	
(i)	
(j)	

SECTION B: PROGRAMMING QUESTION TOTAL – 40 MARKS

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

Question

Particle collision is the fundamental principle behind the invention of cathode ray tube (CRT) televisions (TVs) and computer monitors. The CRT takes particles, called electrons, from the cathode, speeds them up using electromagnets and then smashes them into phosphor particles on the screen. The collision between the electron and phosphor particles results in a lighted spot, or pixel on your TV or computer monitor.

Develop a program to be used as a **particle collision detector**. This program determines whether two particles are colliding. For simplicity, particles are assumed to be sphere in shape. Thus, each particle can be represented by its center point and radius.

Your program should define a class named **`Particle`** for representing a particle. (1 mark)
The class has the following members:

a. Private member data (attributes):
   i. The coordinates of center point, **`x`**, **`y`** and **`z`**, respectively and the radius, **`r`**. (1 mark)

b. Public member functions (methods):
   i. A default constructor which sets all the attributes to zero. 1.5 marks)
   ii. A member function (method) named **`print`** to display all the attributes onto the screen.
   (1.5 marks)
   iii. An overloaded minus operator (**`-`**) that will be used for operations like:

```
p1 - p2
```

where **`p1`** and **`p1`** are of type **`Particle`**. The above operation results in the distance between those two particles, measured between their center points, as given in the following equation.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$$

where (x_1, y_1, z_1) and (x_2, y_2, z_2) are the coordinates of the center points for the two particles respectively. (3.5 marks)

iv. An overloaded greater than operator (>) to compare whether a particle is larger than another particle based on their radius. This operator returns a `true` value if the left operand is larger. (2.5 marks)

v. A member function (method) named **`read`** to read input for the attributes from the keyboard in a flexible manner. Flexible here means that, the user can enter input for an attribute without the need to follow the sequence. For example, the user can enter the radius first, then followed by the coordinates x, y and z, or the user can enter the value for z, then followed by r, x and y. The user can even enter only for selected attributes such as only the radius. However, the input for each attribute should be entered one at a time and must follow a specific pattern as shown in the following examples (Note that the bold texts indicate user inputs). Note also that, this method should use **C++ strings** for string manipulations.

Attribute and value => **`x:10`**

Attribute and value => **`r:5`**

(10.5 marks)

Your program should also define a standalone function named **`collision`** to test whether two particles are colliding. A collision is detected when the distance between two particles is less than or equal to the sum of their radius. The following figures illustrate this scenario.

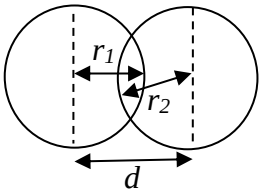

Figure 1: Collision, $d \leq (r_1 + r_2)$

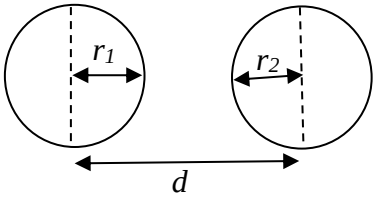

Figure 2: No collision, $d > (r_1 + r_2)$

where r_1 and r_2 are the radius of two particles respectively, and d is the distance between the particles from their centers. A collision is detected in Figure 1 and there is no collision in Figure 2.

The function **`collision`** receives two parameters of type **`Particle`** representing the two particles to be tested for a collision. The function returns a **`true`** value if a collision is detected. Furthermore, this function must be specified such that it has direct access to the attributes of **`Particle`**. (6.5 marks)

```
Particle 1:

Attribute and value=> x:1[Enter]
Attribute and value=> y:2[Enter]
Attribute and value=> z:3[Enter]
Attribute and value=> r:2[Enter]
Attribute and value=> [Enter]

Particle 2:

Attribute and value=> z:5[Enter]
Attribute and value=> x:3[Enter]
Attribute and value=> y:4[Enter]
Attribute and value=> r:3[Enter]
Attribute and value=> [Enter]


** Collision detected **

Particle 1:
Center: (1,2,3)   Radius: 2

Particle 2:
Center: (3,4,5)   Radius: 3

Larger: Particle 2
```

(a)

```
Particle 1:

Attribute and value=> r:3[Enter]
Attribute and value=> [Enter]

Particle 2:

Attribute and value=> r:2[Enter]
Attribute and value=> x:10[Enter]
Attribute and value=> y:20[Enter]
Attribute and value=> [Enter]


** No collision **

Particle 1:
Center: (0,0,0)     Radius: 3

Particle 2:
Center: (10,20,0)     Radius: 2

Larger: Particle 1
```

(b)

Figure 3: Example runs of the program with (a) a collision is detected and (b) no collision detected. Note that the bold texts are user input.

Finally, utilize the class **`Particle`** and the function **`collision`** into the main function to develop the particle collision detection program.

Figure 3 shows sample runs of how the program should look like. Note that, bold texts indicate inputs from the user. The program should perform the following operations:

i. Create two `Particle` objects. 1 mark)
ii. Ask the user to enter the information of the particles. (2 marks)
iii. Determine whether the particles are colliding with each other. (4 marks)
iv. Print the information of both particles. (2 marks)
v. Print which particle is larger. (3 marks)

Answer spaces for Section B

(Notes: Please use the back of the page if space is not enough)

Answer spaces for Section B

(Notes: Please use the back of the page if space is not enough)

Answer spaces for Section B

(Notes: Please use the back of the page if space is not enough)

Answer spaces for Section B

(Notes: Please use the back of the page if space is not enough)