5.5

RAJARATA UNIVERSITY OF SRI LANKA
FACULTY OF APPLIED SCIENCES

B.Sc. in Applied Sciences
First Year - Semester II Examination –January / February 2023

COM 1305 – OBJECT ORIENTED PROGRAMMING

Time: Three (03) hours

INSTRUCTIONS TO CANDIDATES

- This paper consists of **Nine (09)** pages including this page.
- This paper contains of **Five (05)** questions.
- Answer **ALL** questions.
- This examination accounts for 40% of the course assessment. The total maximum mark attainable is 100. The marks assigned for each question and section, thereof are indicated in brackets.
- This is a closed book examination.
- Mobile phones or any other communication devices are not permitted.

---

1. a) Consider the following *Calcu* class. Write the main method that will
   - create an object of *Calcu* class,
   - take two integers as user inputs,
   - call the *add* function and pass those two user inputted integers to the *add* function.

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

class Calcu
{
        public:
        void add(int a, int b)
        {
                std::cout<<a+b;
        }
};
```

(05 marks)

b) Write the C++ code for the following class displayed in UML class diagram notation.

User
-name: String #userID: String
+setName() +getname() +setID() +getID()

(05 marks)

c) Consider the following code. Write a concrete child class named *B* for class *A*.

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

class A
{
        int i;

        public:
        void setI(int b)
        {
                i=b;
        }
```

```
        void printI()
        {
                cout<<i;
        }

        virtual void say()=0;
};
```

(05 marks)

d) Consider question 1 c). Write a main method that will create an object of the class *B* that you wrote, assign the value "10" through function *setI()* and call function *printI()*.

(05 marks)

2. a) Describe why the following code produces an error.

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

class A
{
        void say(){cout<<"say A";}
        virtual void saySpecial()=0;
};

class B
{
        void say(){cout<<"say B";}
        virtual void sayNothing()=0;
};

class C:public A, B
{
        void saySpecial(){}
        void sayNothing(){}
};

int main()
{
  C c;
  c.say();
}
```

(02 marks)

b) Re-write the code segment that needs to be modified in question 2 a) to correctly call the inherited *say()* functions in class *C*. Do not use pointers.

(03 marks)

c) Write the output of the following program.

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

class Parent
{
        private:
        string name;

        public:
        Parent(){cout<<"Parent()"<<endl;}
        Parent(string nme)
        {
                name = nme;
                cout<<"Parent(string)"<<endl;
        }
};

class Child: public Parent
{
        public:
        Child(){cout<<"Child()"<<endl;}
        Child(string s){cout<<"Child(string)"<<endl;}
};

int main()
{
        Child chld1;
        Child chld2("G");
}
```

(05 marks)

d) Write a C++ class that can be used to find the area of various shapes. It should have necessary *findArea()* functions (consider polymorphism principle) to find the area of different shapes when called. Following are the formulas for finding the area of certain shapes. Use them when writing your code.
Triangle = base x vertical height x ½
Square = length of a side x length of a side
Circle = π x radius x radius
π = 3.14 (05 marks)

e) Describe the function of the observer design pattern with a usage example. (05 marks)

3. a) Write down the output in the following program.

```
#include <iostream>
using namespace std;
class Person
{
        private:
        string name;
        int nic;

        public:
        Person():name("nil"), nic(0)
        {}
        Person(string nme, int nicn):name(nme), nic(nicn)
        {}

        void printPerson()
        {
                cout<<name<<endl;
                cout<<nic<<endl;
        }
};

int main()
{
        Person p1;
        Person p2("kamal",983212);
        Person p3(p2);
        p1.printPerson();
        p2.printPerson();
        p3.printPerson();
}
```

(03 marks)

b) In question 3 a), what is the built-in constructor that is being used in the expression "*Person p3(p2);*"? Explain the function of that built-in constructor.

(02 marks)

c) Describe the types of abstraction with examples.

(05 marks)

d) Explain the usage of Getter and Setter functions with the principle of data encapsulation.

(03 marks)

e) Describe the advantage of design patterns.

(02 marks)

f) Explain what is known as dynamic binding. (02 marks)

g) Describe the differences between "include" and "extend" relationships in UML use cases diagrams.

(03 marks)

4 a) Consider the following description. Read and draw a simple class diagram to reflect this. There is no need to identify all possible attributes and functions but need to identify the most important ones. Also use the OOP concepts such as abstraction and inheritance when drawing the class diagram.

"There is a student registration system for a university. This should allow the students to register for courses, academics to view registered students and view results, management staff to view registered students and view and enter results and administrators to create courses, create accounts and update them. All users need to login to the system to use it"

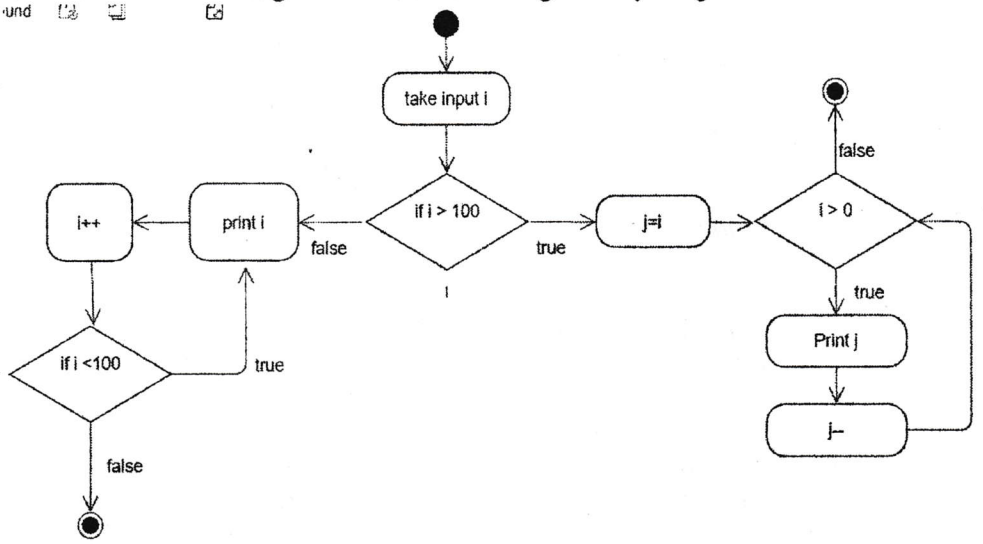
(05 marks)

b) Consider the following steps taken for a login scenario. Draw an activity diagram for that. You do not need to display activity partitions or swim lanes.

1. Users goes to the login page.
2. User enters the user name and password and press enter.
3. The system checks the matching user name and the password in the database.
4. If the login details are correct, user is logged in and moved to the homepage.
5. If the username exists and if the password is incorrect, message is displayed saying password is incorrect.
6. if the username is not in the database, then a message is displayed saying the username does not exist.
7. User has three attempts to correctly enter the login details.
8. after three failed attempts, the account is locked for 20 minutes.

(05 marks)

c) Write the C++ code segment for the following activity diagram.



(05 marks)

d) Draw the activity diagram for the following code.

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

int main()
{
   int i;
        cin>>i;

        int result=(i<10 || i > 4)?i+2/2:i-2*2/5;

        cout<<result;


}
```

(03 marks)

e) Considering the program in question 4 d) write down the outputs when we give the following values separately as inputs.

2
6
4
11

(02 marks)

5 a) Write down the output of the following program.

```
#include <iostream>
using namespace std;
enum months_of_year { Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov,
Dec };

int main() {
months_of_year mon1, mon2;
mon1 = May;
mon2 = Jun;
cout<<mon1*mon2;
cout<<mon1-mon2;
}
```

(02 marks)

b) Consider the following code. Write the code segment that should be inside the *main*() function that will call the *say()* function in the *Parent* class.

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

class Parent
{
        public:
        Parent(){cout<<"Parent()"<<endl;}

        void say(){cout<<"sayParent()"<<endl;}

        virtual void saySpecial()=0;
};

class Child: public Parent
{
        public:
        Child(){cout<<"Child()"<<endl;}

        void say(){cout<<"sayChild()"<<endl;}

        void saySpecial(){cout<<"saySpecial()"<<endl;}
};

int main()
{//your code here
}
```

(05 marks)

c) Describe the difference between instance variables and static variables in C++.

(03 marks)

d) Name operators that cannot be overloaded in C++. (02 marks)

e) Explain whether constructors get inherited in C++. (02 marks)

f) Write the output of the following code.

```
#include <iostream>
using namespace std;
class User
{
        private:
        string name;

        public:
        User():name("null"){cout<<name<<endl;}
        User(string nme):name(nme){cout<<name<<endl;}
        void setName(string nme){name=nme;}
        void getName(){cout<<name<<endl;}

};
class Staff: public User
{
        private:
        int staffID;

        public:
        Staff():staffID(0){cout<<"empty"<<endl;}
        Staff(int stfid):staffID(stfid){cout<<staffID<<endl;}

};
int main()
{
        Staff stf1;
        Staff stf2(1203243);
        stf1.setName("vijitha");
        stf1.getName();
        stf2.getName();
}
```

(06 marks)

---END---