T2 Examination 2022
Semester-II

Course Title: **Software Development Fundamentals-II** Max. Hours:**1 Hr**
Course Code:**15B11CI211** Max. Marks:**20**

T2 Solution

1. Write a C++ code having the following details:

a) A base class Person having two pure virtual functions getData() and isoutstanding().
b) Your parent class should also include two member functions to input() and display() the name of person.
c) Your code includes two derived classes, Student and Instructor respectively.
d) The derived classes each contain a function called getData () and isoutstanding (). getData() function of Student class should input name of person and asks user to input GPA whereas isoutstanding() function determines either the GPA>3 (outstanding) or not.
Similarly, getData() function of Instructor class should input name of person and asks user to input no. of publications whereas isoutstanding() function determines either the no. of publications >50 (outstanding) or not.
e) Student and Instructor objects are casted into the person class type through array of pointers. Ask user first either he is student/instructor, then get his data using getData() and your program continues to ask to enter the data until the last personnel enters the data. Once the data is entered, print the names of all personnel along with their outstanding or not. [**5 Marks**] **[CO-3]**

Sol

a) A base class Person having two pure virtual functions getData() and isoutstanding(). **(.5 marks)**

b) Your parent class should also include two member functions to input() and display() the name of the person**.(.5 marks)**

c) Your code includes two derived classes, student and instructor respectively.

d) The derived classes each contain a function called getData () and isoutstanding (). getData() function of Student class should input name of person and asks user to input GPA whereas isoutstanding() function determines either the GPA>3 (outstanding) or not. **(1 marks)**

Similarly, getData() function of instructor class should input name of person and asks user to input no. of publications whereas isoutstanding() function determines either the no. of publications >50 (outstanding) or not.**(1 marks)**

e) Student and instructor objects are casted into the person class type through array of pointers. Ask user first either he is student/instructor, then get his data using getData() and your program continues to ask to enter the data until the last personnel enters the data. Once the data is entered, print names of all personnel along with their outstanding or not. **(2 marks)**

[Strict Marking]

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

// abstract class
class Person
{
private:
   string name;

public:
   //pure virtual function
   virtual void getData() = 0;
   virtual void isoutstanding() = 0;

   //function to read name
   void input()
   {
      cout << "Enter name of personnel: ";
      cin >> name;
```

```
int main()
{
   int ch;    // to store choice
   int i = 0; // loop variable
   //creating an array of pointers
   Person *arr[10];

   //reading inputs
   while (i < 10)
   {
      cout << "\nEnter Data for Person" << i + 1 << endl;
      cout << "Enter 1 for Student\nEnter 2 for
Instructor\nEnter your response: ";
      cin >> ch;
      if (ch == 1)
      {
         //creating object for student class
         arr[i] = new Student();
         arr[i]->getData();
```

```
    }

    //function to display name
    void display()
    {
        cout << "Name: " << name;
    }
};

class Student : public Person
{
private:
    double gpa;

public:
    //method to read data
    void getData()
    {
        input();
        cout << "Enter GPA: ";
        cin >> gpa;
    }

    //method to display outstanding or not
    void isoutstanding()
    {
        if (gpa > 3)
            cout << "Outstanding";
        else
            cout << "Not Outstanding";
    }
};

class Instructor : public Person
{
private:
    int publications;

public:
    //method to read data
    void getData()
    {
        input();
        cout << "Enter Publications: ";
        cin >> publications;
    }

    //method to display outstanding or not
    void isoutstanding()
    {
        if (publications > 50)
            cout << "Outstanding";
        else
            cout << "Not Outstanding";
    }
};
```

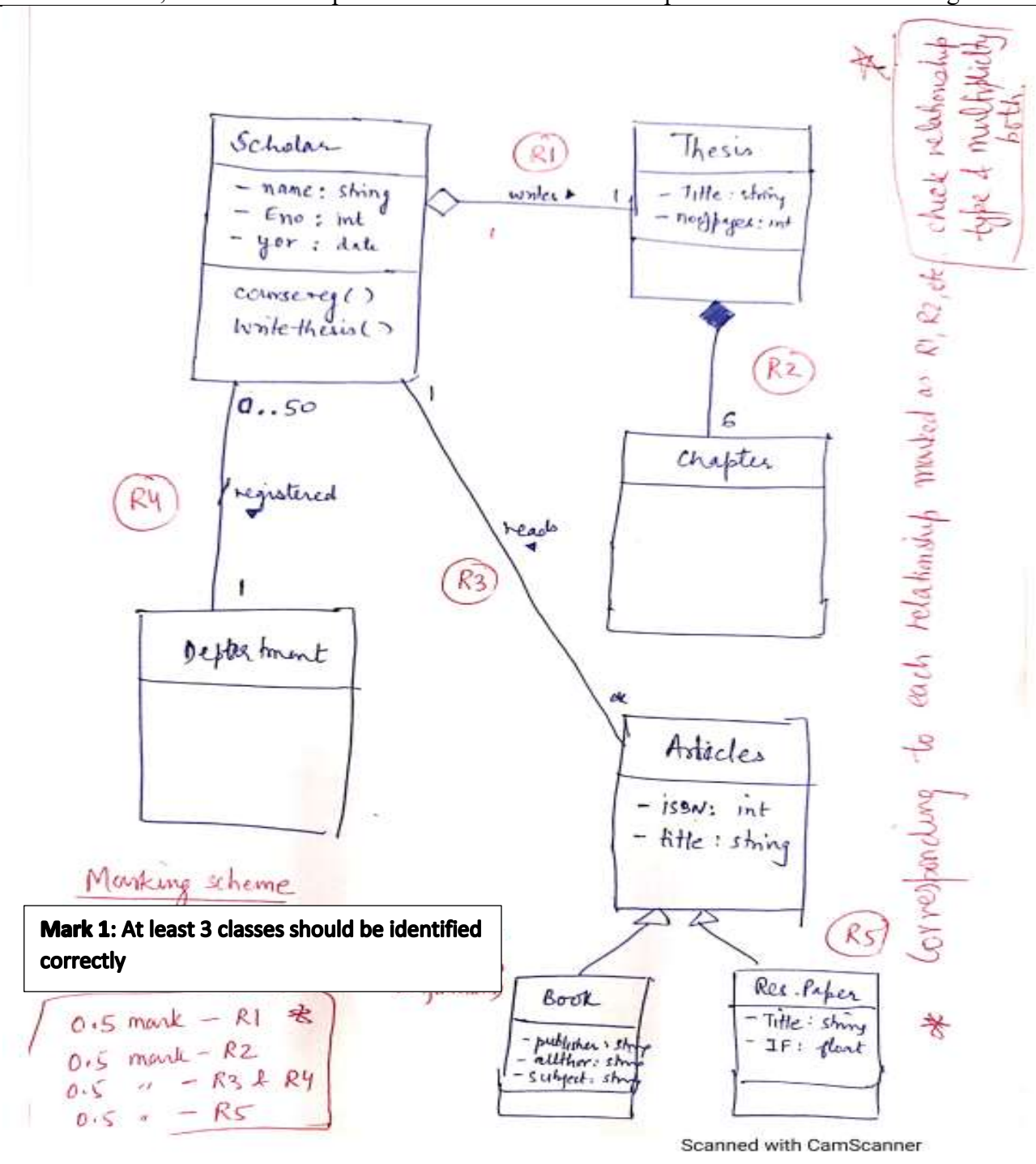
```
        }
        else if (ch == 2)
        {
            //creating object for instructor class
            arr[i] = new Instructor();
            arr[i]->getData();
        }
        else
        {
            cout << "Invalid input. Try again.\n";
            continue;
        }
        i++;
    }

    cout << "\n\nALL PERSON DETAILS\n";
    cout << "-----------------------------------------\n";
    for (i = 0; i < 10; i++)
    {
        arr[i]->display();
        cout << "\t";
        arr[i]->isoutstanding();
        cout << endl;
    }
}
```

2. A scholar studying in a department is identified by his name, enrollment number and year of registration. Each scholar can perform some functionalities like course registration, write thesis etc. Further, it is compulsory for each scholar to write his/her one thesis and submit it for degree allotment. The scholar's thesis must consist of six chapters. Also, a thesis is identified by its title and number of pages. To write the thesis, a scholar needs to read different articles (or study material). Each article has an ISBN number and title. Further, each article can be categorized as a book or a research paper. A book has a publisher, author, and subject name. A research paper has attributes like title, impact factor etc. For awarding

degree, each scholar must be registered in any one department. However, a department can enroll any number of students with maximum seats as 50.
With respect to the above case study perform the following: **[CO-2]**

a) [**3 Marks**] Represent the above scenario with the help of a complete class diagram. Draw necessary classes with their attributes and functionalities. Also, mention the relationships and their multiplicity, direction and roles properly.
b) [**3 Marks**] Write a C++ program to just show the implementation of the relationship between Scholar, Thesis and Chapter class. You don't need to implement the entire class diagram.

SOL



Implementation

Thesis & Chapter : Composition

```
class Chapter
{
    int no of pages;
    int title;
    string author;
public:
    Chapter()
    {
    }
    writechapter()
    {
    }
};
```

```
class Thesis
{
    string title;
    int no.of pages;
    Chapter C[6];
public:
    Thesis()
    {
    }
    :
};
```

6 Objects of chapter class and NOT POINTERS to ensure composition

1 mark

class Scholar
{
string name;
int eno;
date dor;
Thes

Implementation

1 mark — composition

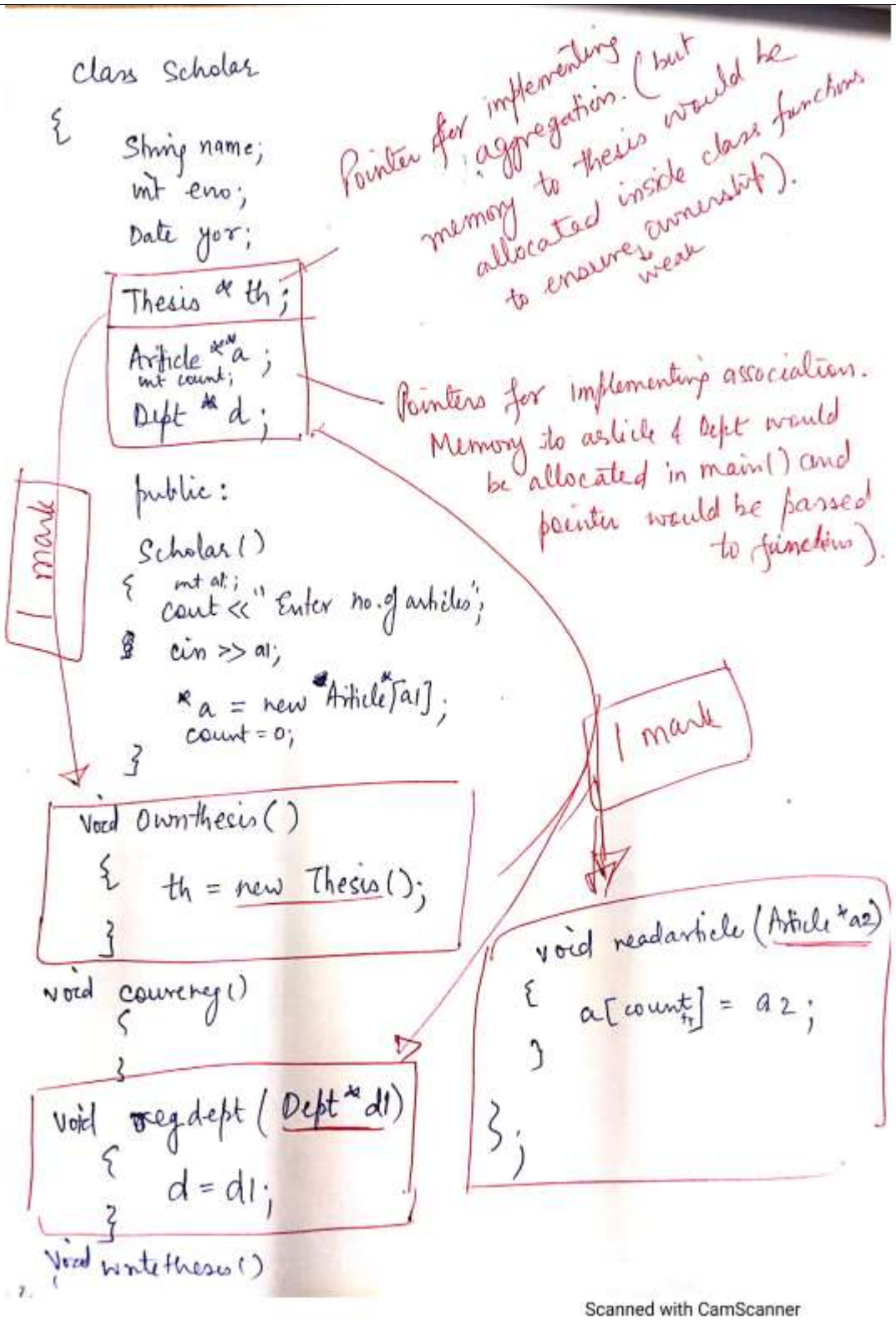
1 mark — Aggregation (b/w Scholar & thesis)

1 mark — 0.5 + 0.5

Association

Scholar & Dept

Scholar & Article



3. Write a program that can throw integer and double exceptions in the same try block. For both exceptions, implement the exception handling blocks with three different catch blocks of type (int, double, and default). Default catch block which can accept any exception. [**Marks 4**] **[CO-4]**

SOL

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

int main()
{
   try
   {
      cout << "Throwing an int exception..." << '\n';
      throw 12;
      cout << "Throwing a double exception..." << '\n';
      throw 56.728;
   }
   catch (int ex)
   {
cout << "Integer exception: " << ex << " caught and handled."<< '\n';
   }
   catch (double ex)
   {
       cout << "Double exception: " << ex << " caught and handled."<< '\n';
   }
   catch (...)
   {
      cout<<"For unmatched exception";
   }
return 0;
}
```

MARKING SCHEME [flexible Marking]
-For one Catch—1-mark
-for two Catch—1+1=2 Marks
-for all three catch blocks ---1+1+1=3 Marks.
---for proper main structure and input – ---1 Marks
(Marks allotted only when the whole program is syntactical and logical correct)

4. Find the error/output of the following programs. [**2.5 Marks Each**] **[CO-3]**

1.

```
#include<iostream>
using namespace std;
class A
{
   public: A()
   {cout<<"\n 1";}
   A(int x)
   {cout<<"\n 2";}
   virtual void T2()
  { cout<<"\n A T2()"; }
   ~A()
  {cout<<"\n A dead";}
};
class E
{
   public: E()
   {cout<<"\n 3";}
   E(int x)
   {cout<<"\n 4";}
};
```

```
class B: virtual public A
{
   public: B()
{cout<<"\n 5";};
   B(int x) {cout<<"\n 6";}
   void T2()
  {cout<<"\n B T2()";}
   ~B()
{cout<<"\n B dead";}
};

class C: virtual public E
{
   public: C()
{cout<<"\n 7";}
   C(int x)
{cout<<"\n 8";}
};
```

```
class D: public C, public B
{
   public: D(): E(5), B(1)
            {cout<<"\n 9";}
   D(int x): A(3), C(2)
           {cout<<"\n 10";}
     void T2()
     {cout<<"\n D T2()";}
   ~D()
      {cout<<"\n D dead";}
};

int main()
{
   B* obj=new D(2);
   obj->T2();
   delete obj;
}
```

OUTPUT [Partial Marking]
[MARKS 2.5]

Statement in Void main	Output of each Statement	Complete output
B* obj=new D(2); **[1 Marks]**	3 2 8 5 10	3 2 8 5 10
obj->T2(); **[.5 Marks]**	D T2()	D T2()
delete obj; **[1 Marks]**	B dead A dead	B dead A dead

2.
```
#include <iostream>
using namespace std;
class University
{
public:
   University() {}
   virtual void print() { cout << "University"; }
};
class Institute: public University
{
public:
   Institute() {}
   void print() { cout << "Institute"; }
};
int main()
{
   try
   {
      try
      {
         throw Institute{};
      }
      catch (University &b)
      {
         cout << "Exception in University, which is actually ";
         b.print();
         cout << "\n";
         throw b;
      }
   }
   catch (University &b)
   {
      cout << "Exception in University, which is actually ";
      b.print();
      cout << "\n";
   }
   return 0; }
```

OUTPUT

Exception in University, which is actually Institute

Exception in University, which is actually University

[Partial Marking]

[1 mark] will be given if get the first output only

[2.5 marks] will be given if get the both the output