Independent University, Bangladesh *10 July 2021*
CSE213/CSC305: Object Oriented Programming – I
Mid Term Exam, Summer 2021, Section ALL

**Total Marks: 100; Duration: 2 hours**

1. **PROJECT:** Answer the following: **[To answer Q1, write your project topic first]** **[30 marks]**

   a. Similar to milestone-1 of your project, write **adequately elaborated logically acceptable goals** for your given course project topic. Write the MOST important goal for each of the user types (**one goal per user**). **[5]**

   b. Write **workflow** (similar to milestone-1 of your project) for **ONE** of the above goals described in 1.a considering all user-initiated and background events including any pre-processing/calculation/verification, where at least three classes will be needed to support the workflow. **[10]**

   c. Define **At least three (there may be more) classes needed to support your workflow**, and define them using following structure: **[5+5+5 = 15]**

```
//write .java file name here
//add package statement
public class ClassName { //if you choose Java-format
    //declare private fields with proper name, type and scope
    private fieldType field1;  private fieldType field2;
    private fieldType field3;  // and so on…

    //define public methods with:
          // proper name, returnType & parameterList
          // and pseudocode inside the body
    public returnType method1(type parameter1, type parameter2, ...){
          //pseudocode of the method
          //return statement, if necessary
    }
    public returnType method2(type parameter1, type parameter2, ...){
          //pseudocode of the method
          //return statement, if necessary
    }//and so on...
}                          //*****OR ******//
class ClassName { //if you choose C++ format
    private:
    fieldType field1; fieldType field2; fieldType field3; //and so on…
    public:
    //define public methods with:
          // proper name, returnType & parameterList
          // and pseudocode or work-explanation inside the body
    returnType method1(type parameter1, type parameter2, ...){
          //pseudocode of the method & return statement, if necessary
    }
    returnType method2(type parameter1, type parameter2, ...){
          //pseudocode of the method & return statement, if necessary
    }
    //and so on...
};
```

**Field and Method names must be aligned to satisfy the goal & workflow described in answer of 3.a & 3.b**
Instead of **System.out.println(...)**, you can use **sout(...)** to print relevant messages.
For example, the Student class necessary for course registration of Student user may look like: **[P.T.O]**

## // EXAMPLE of a java-format class for question 1.c:

```
package packageName;
import java.util.Random;
import anotherPackageName.ImportedClassNameFromAnotherPackage;

public class Student{
      private int id; private String name, dept, major;
      public float calcCgpa(){
            float cgpa = …;
            // pseudocode or explanation of cgpa calculation
            return cgpa;
      }
      public void printTranscript(){
            // pseudocode or explanation of generating and printing
            // transcript
      }
}
```

**P.T.O**

2. **JAVA:** Complete the following incomplete code to a Full java program (MainClass.java): **[30 marks]**

```
class Array{
   protected int[] intData;
   private int sum;
   // also define necessary getter and/or setter method(s)
[5]
}

class ArrayOfArrays{
   private Array[] arrOfarr;
   // define method: "setArrayOfArrays"                                    [10]
   // set random integers and sum to the fields for each of arr [see RUN]

   // define method: "showArrayOfArrays"                                   [5]
   // show elements and sum as per the RUN

   // define method: "getAvgOfSums"                                        [5]
   // display the average of sums as per the RUN


}

public class MainClass {  //rewrite the MainClass into a FULL class       [5]
  public static void main(String[] args){
     ArrayOfArrays obj = new ArrayOfArrays();
     int n, max = 100;
     // n = no of Array instances in obj, i.e. size of arrOfarr
     // (can be set with user input OR random number, your choice)
     //  max  is  the  upper  limit  of  random  values  for  Array  elements  in
intData

     obj.setArrayOfArrays(n, max);
     // set-random size of each Array instances inside obj,
     // i.e. size of obj.arrOfarr[i], then instantiate and populate them
     // with random integers up to max. [see sample RUN]
     // Beware of required multi-level memory allocation

     obj.showArrayOfArrays();
     // display the Array instances inside obj as shown in sample RUN

     sout("Average of sums of ALL Arrays in obj is: " + obj.getAvgOfSums());
     // display the average of sums of m as shown in sample RUN

  }//end main()
}//end class
```

**RUN:**

```
Size of obj.arrOfarr is randomly set to: 3
Upper limit of random values is 100

Size of the Array of obj.arrOfarr[0] is randomly set to: 2
Size of the Array of obj.arrOfarr[1] is randomly set to: 4
Size of the Array of obj.arrOfarr[2] is randomly set to: 3

The elements of obj having 3 Array instances are:
{45,73}sum:118
{54,76,87,98}sum:315
{33,44,55}sum:132

Average of sums of ALL Arrays in obj is: 188.33
```

P.T.O

3. **C++:** Consider the following code. You need to write the COMPLETE program. **[40 marks]**

```
class Engine {
      private:
      float horsepower; string engType; int noOfCylinder;
      public:
      // define necessary methods, AND/OR declare friends and define them globally,   [5+5=10]
      // for input and output so that the given main() works.
};
```

```
class Truck {
      private:
      Engine eng; string model, truckType; float price;
      // define necessary methods, AND/OR declare friends and define them globally,   [5+5=10]
      // for input and output so that the given main() works.
};
```

//add necessary global functions (which are friends) so that the given main() works

```
int main(){
  Truck *truckPtr; int garageCapacity, trucksInGarage = 0, choice, i;
  cout<<"Initial garage capacity "; cin>> capacityOfgarage;
  do{
     cout<<"Enter [1] to add a new truck to the garage."<<endl;
       cout<<"Enter [2] to show truck info for ALL trucks of the
    garage."<<endl;
     cout<<"Enter [3] to exit."<<endl;
     cout<<"Enter your choice..."<<endl; cin>>choice;

     switch(choice){
        case 1:
         allocateOrExtendTruckPtr(truckPtr,garageCapacity,trucksInGarage);[15]
          break;
            // this global friend function of Truck class will allocate memory for the garage (for 1st time)
            // or extend the garage capacity (if garage capacity is full) using truckPtr,
            //  and then add a new Truck instance (after taking truck details including
            // engine info from user) as the last element to truckArr

       Case 2: for(i=0;i<trucksInGarage;i++) cout<<truckArr[i]<<endl;     [5]
             break;
            // the loop show information of ALL trucks
            //define necessary operator method/function for above "cout<<truckArr[i]" statement
     }// end of switch
  }while(choice!=3);
  return 0;
}
```