Disclaimer: Solve these PRACTICE problems to reinforce your concept only
Do not treat them as a question bank

Do not expect them to appear in exam

Happy coding 😊

C++

1. **C++:** Consider the following main(). You need to complete the classes along with necessary fields and methods and run the program.

   Difficulty Level: 50%

```
class Address {
   // MUST have houseNo, roadNo, street, thana, district, zipCode, etc. as private fields
  Public: // add necessary methods including setters and getters,
          // AND/OR declare friends, so that the given main() works
};
class Employee{
  // MUST have employeeID, name, department and the following as private fields
  Address empAddress;
  Public: // add necessary methods including setters and getters,
          // AND/OR declare friends, so that the given main() works
};

int main()
{
     Employee e1, e2(4161, "S K Dey", "CSE");
     //parameterized constructor also ask for address details

     cout << "Give input for employee 1: " << endl;
     cin >> e1;
     //Should ask employeeID, name, department.
     //Then ask address related information.

     cout << "Complete information of the employees are: " << endl;
     cout<< e1 << e2 << endl;

     cout<<"The employee e1 ";
     if(e1 == 1229)   cout << "lives in Bashundhara R/A." << endl;
     else cout << "does NOT lives in Bashundhara R/A." << endl;
     //comparing zipCode field

     return 0;
}
```

P.T.O

2. **C++:** Consider the following main(). You need to complete the classes along with necessary fields and methods and run the program.

Difficulty Level: 50%

class Date {
// MUST have day (int), month (string), year (int) as private fields.
Public: // add necessary methods including setters and getters,
// AND/OR declare friends, so that the given main() works
};

class Employee{
// MUST have **employeeID**, **name**, **department** and the following as private fields.
Date dateOfJoining;
Public: // add necessary methods including setters and getters,
// AND/OR declare friends, so that the given main() works
};

```
int main()
{
   Employee e1, e2(4161, "S K Dey", "CSE");

   cout << "Give input for employee 1: " << endl;
   cin >> e1;
   //Should ask employeeID, name, department.
   //Then ask for date of joining related information.

   cout << "Complete information of the employees are: " << endl;
   cout<< e1 << e2 << endl;

   cout<<"Mr./Ms. "<< e1.getName();

   if(e1 == 2017 && e1 == "October")
     cout<<" joined the company in October 2017." << endl;
   else
     cout<<" did NOT joined the company in October 2017." << endl;

   return 0;
}
```

P.T.O

3. **C++:** Consider the following main(). You need to complete the classes along with necessary fields and methods and run the program.

Difficulty Level: 50%

class Author {
// MUST have **ID** (int), **name** (string) and **email** (string) as private fields.
Public: // add necessary methods including setters and getters,
// AND/OR declare friends, so that the given main() works
};

class Book {
// MUST have **title** (string), **ISBN_No** (string), **price** (float) and the following as private fields.
Int noOfAuthor;
Author *ptr;
Public: // add necessary methods including setters and getters,
// AND/OR declare friends, so that the given main() works
};

```
int main()
{
   Book b1, b2("Data Structures", "978-0-7334-2609-4", 550);
 //parameterized constructor also ask for no of authors and their details

   cout << "Give input for book 1: " << endl;
   cin >> b1;
   //Should ask title, ISBN_No, price.
   //Then ask #ofAuthors & author info (id, name and email) for all authors.

   cout << "Complete information of the books are: " << endl;
   cout<< b1 << b2 << endl;

   cout<<"For The book titled: "<< b1.getTitle() <<endl;

   if(b1 <= 1000 || b1 == "Ataul Karim")
     cout<<"Ataul Karim is an author, or the price is <= 1000"<<endl;
   else
     cout<<"Ataul Karim is NOT an author & price is > 1000"<<endl;

   return 0;
}
```

P.T.O

4. **C++:** Consider the following main(). You need to complete the classes along with necessary fields and methods and run the program.

Difficulty Level: 80%

class Author {
// fields: **authorName** (string), **`authorEmail`** (string),
Public: // add necessary methods including setters and getters,
// AND/OR declare friends, so that the given main() works
};

class Book {
// fields:
//`bookName` (string), **`authorList`** (Author*), **noOfAuthors** (int), **`price`** (float), **`isbn`** (string)
Public: // add necessary methods including setters and getters,
// AND/OR declare friends, so that the given main() works
};

class Course {
// fields: **`courseId`** (string), **`courseTitle`** (string), **`noOfCredits`** (int), **`textBook`** (Book)
Public: // add necessary methods including setters and getters,
// AND/OR declare friends, so that the given main() works
};

//Complete the following global function
returnType `allocateMemory` (//decide parameter list**) {**// complete the function**}**

```
int main(){
  Course *courseArr; int n, i;
  Cout<<"How many courses? "; cin>>n;
  allocateMemory(courseArr, n); //it is a global function
  for(i=0;i<n;i++)courseArr[i].populateCourse().displayCourse();

 int totalCredits = 0;
  for(i=0;i<n;i++) totalCredits += courseArr[i];
  cout<<"Total no of credits of these courses is: "
      <<totalCredits<<endl;
  return 0;
}
```

P.T.O

5. **C++:** Consider the following code. You need to complete the classes along with necessary fields and methods and run the program.

Difficulty Level: 80%

class HardDisk {
// fields: **rpm** (int), **capacityInGB** (int),
Public: // add necessary methods including setters and getters,
// AND/OR declare friends, so that the given main() works
};

class Computer {
// fields: **brand** (string), **speedInGhz** (float), **noOfHDD** (int), **hddArr** (HardDisk*), **price** (float)
Public: // add necessary methods including setters and getters,
// AND/OR declare friends, so that the given main() works
};

class ComputerLab {
// fields: **roomNo** (string), **noOfComputer** (int), **compArr** (Computer*)
Public: // add necessary methods including setters and getters,
// AND/OR declare friends, so that the given main() works
};

```
int main(){
  ComputerLab* labs; int n, i;
  Cout<<"How many computer labs? "; cin>>n;
  labs = new ComputerLab[n]; int totalStorageOfAllLabsInGB = 0;
  for(i=0;i<n;i++) {

    labs[i].setLabInfo().showLabInfo();

    totalStorageOfAllLabsInGB += labs[i];
    cout<<"Total Storage of ALL Labs combined is: "
       << totalStorageOfAllLabsInGB <<" GB"<<endl;

    if (labs[i] >= 30) cout<<i<<"-th Lab is a big computer Lab."<<endl;
    else cout<<i<<"-th Lab is a small computer Lab."<<endl;
   }

  return 0;
}
```

P.T.O

6. **C++:** Consider the following code. You need to complete the classes along with necessary fields and methods and run the program.

Difficulty Level: 70%

class Book {
  // private fields: `bookName` (string), `authorNames` (string*), noOfAuthors (int), `price` (float)
  Public: // add necessary methods including setters and getters,
        // AND/OR declare friends, so that the given main() works
};

class Course {
  // private fields: `courseId` (string), `courseTitle` (string), `noOfCredits` (int), `textBook` (Book)
  Public: // add necessary methods including setters and getters,
        // AND/OR declare friends, so that the given main() works
};

//add necessary global function so that the given main() works

```
int main(){
  Course *enrolledCourses; int n, i, ratePerCredit;

  cout<<"How many courses to enroll? "; cin>>n;
  cout<<"Rate per credit for this student? "; cin>> ratePerCredit;
  cout<<"Scholarship (%) for this student? "; cin>> scholarshipPercent;

  enrolledCourses = new Course[n];

  for(i=0;i<n;i++) enrolledCourses[i].setCourseInfo().showCourseInfo();

  cout<<"Total amount to be paid is:"
      <<getBillAmount( enrolledCourses, n,
                  ratePerCredit, scholarshipPercent ) <<endl;
  return 0;
 }
```

P.T.O

7. **C++:** Consider the following code. You need to complete the classes along with necessary fields and methods and run the program.

Difficulty Level: 90%

```
class ComplexNo {
    int real, img;
    Public: // add necessary methods including setters and getters,
            // AND/OR declare friends, so that the given main() works
}
//represents complex number in the form of "real +/- img i" e.g: 2+3i

class ComplexRow {
    int noOfCNperRow;
    ComplexNo *CnPtr[];
    Public: // add necessary methods including setters and getters,
            // AND/OR declare friends, so that the given main() works
}


int main(){
    int row, rowIndex1, rowIndex2, i;
    ComplexRow *cnRows;

    cout<<"How many rows of complex nos? "; cin>>row;
    cnRows = new ComplexRow[row];
    for(i=0;i<row;i++)
        cin>>cnRows[i];
    // ask no of complexNo in i-th row & populate them with ComplexNo objects;

    cout<<"The matrix/2-D shape of complex numbers is: " <<  endl;
    showComplexRows(cnRows, row); //global function

    ComplexRow mergedRow;
    cout<<"Enter indices of two rows (<row) to merge: ";
    cin>>rowIndex1>>rowIndex2;
    mergedRow = cnRows[rowIndex1].mergeWith(cnRows[rowIndex2]);
    cout<<"The MERGED rows of complex numbers is: "<<mergedRow<<endl;

    int lower, upper;
    //get values of lower & upper from user

    // Now display all the Complex numbers from mergedRow,
    // whose real<=lower & img>=upper

    Return 0;
}
```