Lab # 2

Object Oriented Programming

BS SE F17 Afternoon

Instructions:

• **Attempt the following tasks exactly in the given order.**

• Indent your code properly.

• Use meaningful variable and function names. Follow the naming conventions.

• Use meaningful prompt lines and labels for all input/output.

• Make sure that there are **NO dangling pointers** or **memory leaks** in your program.

Task # 1

Given the following two structures:

```
struct Course
{
        char course_name[100];          // Name of the course
        double  obtained_marks;
        double total_marks;
};
struct Result
{
        int student_id;                 // Unique id for student
        int num_of_courses;             // No. of courses
        Course *courseList;             // Array containing the courses which are
                                        // part of the result
        double percentage;           // Total percentage  (which is
                                    //calculated by marks of all the courses)

};
```

You are required to implement two functions as described below:

- **computeAllResult**: it receives two parameters: an array (***allResult***) of type Result and its size The function **computeAllResult** calls another function **computeOneResult** for each element of Result type, using a loop.

- **computeOneResult**: it receives one parameter: a pointer to a Result (***ptrResult***) and computes percentage by using the courseList and stores it in its member variable ***percentage***.

  Here are the prototypes of functions

  void computeAllResult(Result *allResult , int size)

  void computeOneResult(Result * ptrResult)

  Also write down the main function to demonstrate your implementation.

Task # 2

Matrix Application

In this problem, our goal is to design a library, which will support basic operations of Matrices. You have following structure for matrix

```
struct Matrix{

        int rows;
        int cols;
        int **data
 };
```

You are required to design the following operations.

1. void createMatrix (Matirx m, const int row=1, const int Col=1);

2. int& at(Matrix m, const int r , const int c);

//For setting or getting some value at a particular location of matrix

3. void printMatrix(const Matrix m)

4. int isIdentity (const Matrix m)
if a_{ij} = 0 for i != j and a_{ij} = 1 for all i = j .

5. bool isRectangular (const Matrix m)
In which number of rows are not equal to number of columns.

6. bool isDiagonal (const Matrix m)
I f a_{ij} = 0 for all i != j and at least one a_{ij} != 0 for i = j;

7. bool isNullMatrix (const Matrix m)
A matrix whose each element is zero.

8. bool isLowerTriangular (const Matrix m)

9. bool isUpperTriangular (const Matrix m)

10. bool isTriangular (const Matrix m)

11. Matrix getMatrixCopy (const Matrix m)

12. bool isEqual(const Matrix m1 , const Matrix m2)

13. void freeMatrix (Matrix m); *//Free the dynamically allocated memory.*

14. Matrix Transpose (const Matrix m);

16. void reSize (Matrix m, const int newrow, const int newcol);

17. bool isSymmetric (const Matrix m)
If $A^t = A$

18. bool isSkewSymmetric (const Matrix m)
If $A^t = -A$

19. Matrix add (const Matrix m1, const Matrix m2);

20. Matrix multiply(const Matrix m1, const Matrix m2);
Also write down the main function to demonstrate your implementation.