Home Work 1

Object Oriented Programming

Instructions:

- You are required to create a **multi-file project** for each task.
- 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

Matrix Application

In this problem, your goal is to design an ADT, which will support basic operations of Matrices. You have following class for matrix

```
class Matrix
{
private:
	int row;
	int col;
	int **data;
public:
	Matrix();
	Matrix(const Matrix &);
	Matrix(int, int);
	void setRow(int);
	void setCol(int);
	int getRow() const;
	int getCol() const;
	int& at(const int r, const int c); //For setting or getting some value at a particular location of matrix
	void printMatrix() const;
	int isIdentity() const;
	bool isRectangular()const ;
	bool isDiagonal()const ;
	bool isNullMatrix() const ;
	bool isLowerTriangular() const;
	bool isUpperTriangular() const;
	bool isTriangular()const ;
	Matrix getMatrixCopy()const;
	bool isEqual(const Matrix m2) const;
	void reSize(const int newrow, const int newcol);
	bool isSymmetric() const;
	bool isSkewSymmetric()const ;
	Matrix Transpose() const;
	Matrix add(const Matrix ) const;
	Matrix minus(const Matrix ) const;
	Matrix multiply(const Matrix ) const;
	void freeMemory();

};
```

Task 2

Design an ADT 'Set' whose objects should be able to store an integer set.

Data Members:

- int *data; /* pointer to an array of integers which will be treated as set of integers */
- int noOfElements; // number of elements in the Set
- int capcity; /* maximum possible number of elements that can be stored in the Set */

Supported Operations:

The class 'Set' should support the following operations

1. Set (int cap = 5);
   Sets cap to capacity and initializes rest of the data members accordingly. If user sends any invalid value then sets the cap to default value.

2. Set(Set & ref)
   The sizzling copy constructor

3. void freeMemory()
   Free the dynamically al located memory.

4. void insert (int element);
   stores the element in the Set .

5. void remove (int element);
   removes the element from the Set .

6. int getCardinality()
   returns the number of element s in the set .

7. Set calcUnion (Set & s2)
   returns a new Set object which contains the union of ' s2 ' set and calling object set .

8. Set calcIntersection (Set & s2)
   returns a new Set object which contains the inter section of ' s2 ' set and calling object set .

9. Set calcSymmetricDifference (Set & s2)
   returns a new Set object which contains the symmetric Difference of 's2' set and calling object set .

Where symmetric difference is: $A \Delta B = A \cup B - A \cap B$

10. Set calcDifference (Set & s2)
    returns a new Set object which contains the difference of ' s2 ' set and calling object set .

11. int isMember (int val)
    returns 1 if ' val ' is member of the set otherwise return 0 .

12. int isSubSet (Set & s2)
    returns 1 if s2 is proper subset of calling object set , return 2 if improper subset otherwise return 0 .

13. void reSize (int newcapacity)
    resize the set to new capacity. Make sure that elements in old set should be preserved in the new set if possible.