Objective:

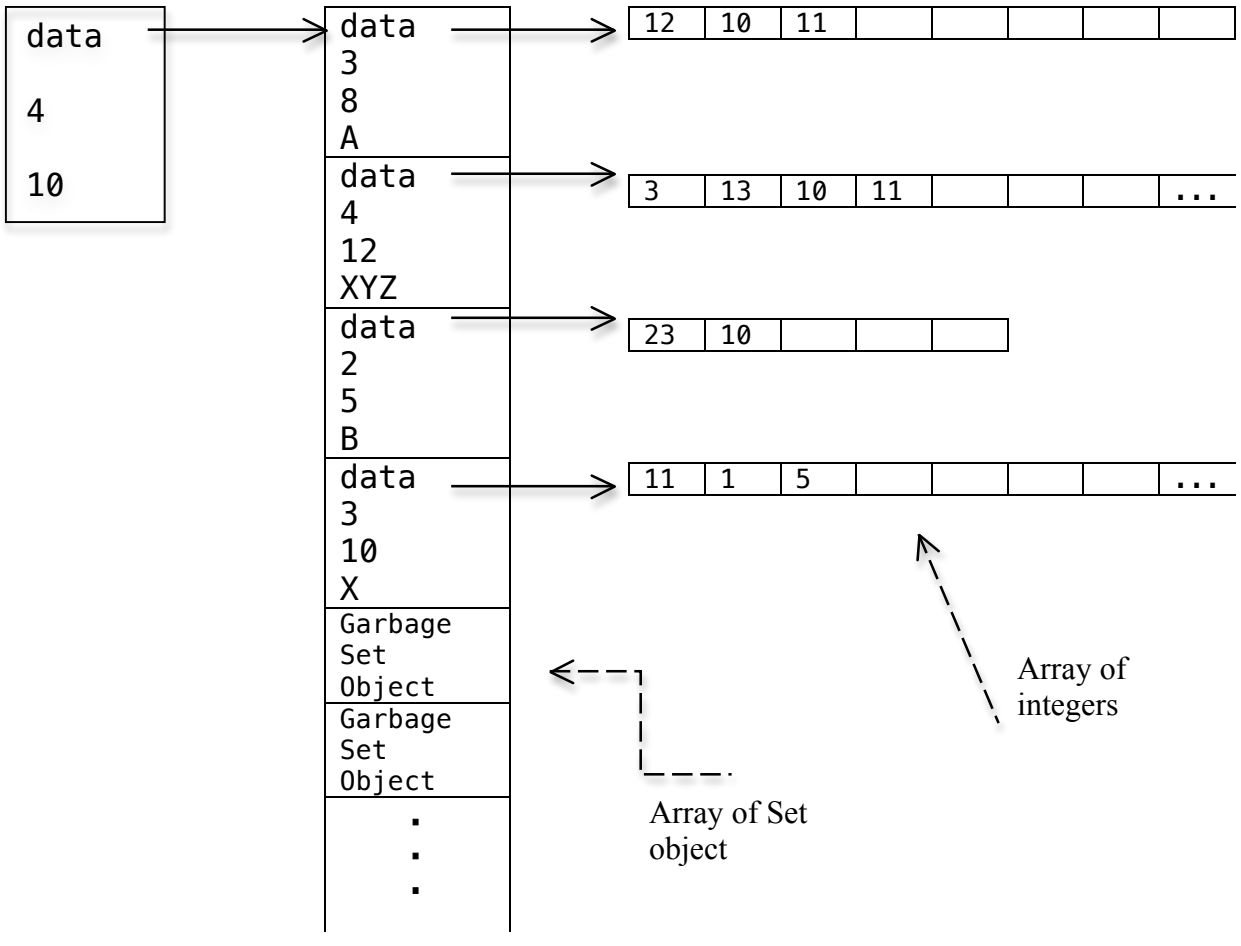
- The purpose of this quiz is to focus on the very basic fundamental concepts learned so far in previous lectures.

Following will help you to answer Question 1 and 2.
You guys should be quite familiar with struct 'Set' and struct 'SetArray'. The diagram below shows an object layout of a SetArray object named as 'sa', in which the number of sets stored by user are 4 and capacity to store sets in it is 10.

```
struct Set
{
   int * data;
   int noOfElements;
   int capacity;
   char name[30];
};
```

```
struct SetArray
{
   Set * data;
   int noOfSets;
   int capacity;
};
```

```
SetArray sa;
```



Question No. 1: (1.0)

What syntax should we write? If we have to access the second element from set of integers stored in 4th location of SetArray 'sa' object as shown in object layout above.

Question No. 2: (2.0)

Write the following function, which receives SetArray object, and deallocate all the data/array pointed by data pointer of SetArray object.
Remember: To deallocate the array of integers captured by each set object.

```
void freeSetArray ( SetArray & sa );
```

Question No. 3: (1.0)
Why C++ implicitly provides the mechanism of "this" pointer?

Because functions have single copy and the called function need to know, which object invoked him so that the called function may accordingly access the attributes of the calling object from memory.

Question No. 4: (3.0)
How is it possible to block the copying of objects of a particular class? For example for the class 'Set', user should get syntax error for doing following.

```
Set s1;
Set s2 = s1;  //must produce syntax error
```

Define copy constructor private ☺

Question No. 5: (1.0)
What enables a group of objects to belong to the same class

A. All of their attributes have identical values across all objects
B. A few selected attributes have identical values across all objects
C. All of them possess the same attributes
D. (B) and (C)

Question No. 6: (12.0)
Design an ADT 'Matrix' whose objects should be able to store a matrix of integers. Matrix ADT should also perform specified operations listed below related to matrices.

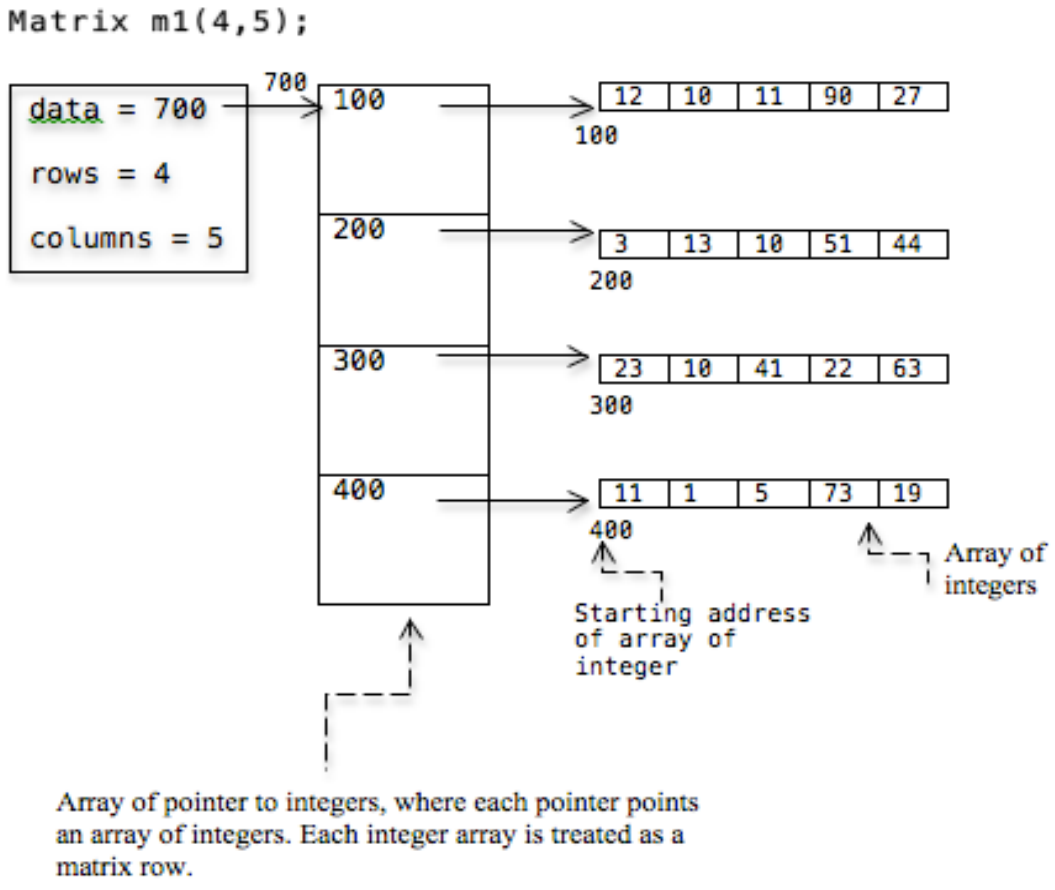
Data Members:

- `int * * data;` /* pointer to an array of pointers whose each location points to an array of integers */
- `int rows;` // number of rows in matrix
- `int cols;` // number of columns in matrix

Supported Operations:

1. `Matrix ();` (0.1)
Set row and col to 0 and obviously initializes data to zero as well. Represents a null matrix

2. `Matrix (int r, int c);` (0.5)
Set the r to row and c to col and creates matrix structure appropriately. If user sends invalid value in r or c or both then set them to 0.
As an example, if we pass (4,5) to constructor, then following object-layout/diagram will appear in memory.



3. Matrix (Matrix &) (1.0)
The copy ctor; you know what to do. ☺

4. ~Matrix (); (1.0)
Free the dynamically allocated memory.

5. int & at(int r, int c); (1.0)
For setting or getting some value at a particular location of matrix

6. int getRows(); (0.2)
returns the number of rows of the matrix.

7. int getColumns(); (0.2)
returns the number of columns of the matrix.

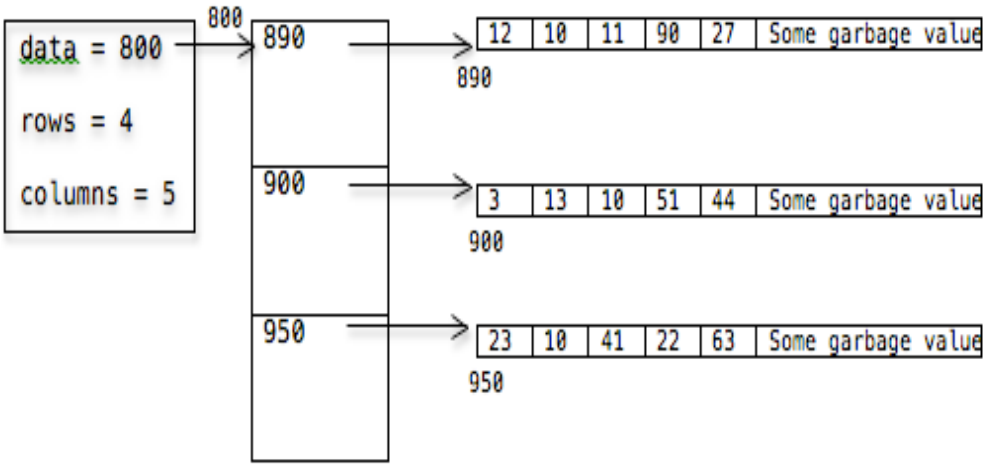
8. Matrix Transpose (); (2.0)
*Find Transpose of the *this object (calling object) and returns the transpose in a new Matrix object. No change in calling object.*

9. Matrix add (Matrix & m2); (2.0)
Add this(calling object) and m2 object and return the result in a new Matrix object. No change in calling object and received object.

10. void reSize (int newrow, int newcol); (4.0)
resize the matrix according to new row and column. Make sure that the old matrix elements should be preserved in the new resized matrix if possible.

Sample Run:

```
Matrix m1(4,5);// at this time all matrix values are garbage.
//initialize all value matrix cells to zero.

for ( int i=0; i<m1.getRows(); i++ )
{

      for ( int j=0; j<m1.getCols(); j++ )
      {
            m1.at(i,j) = 0; // either we can take input from keyboard
                        //by writing: cin >> m1[i][j];
                        //similarly we can display values of matrix
                        //by writing: cout >> m1[i][j];


      }
}

Matrix m2 = m1;

Matrix m3 = m1.add(m2);

m1.reSize(3,6); // assume the elements in m1 are what is shown in above diagram.
               // After resizing the m1 object diagram will be as follows:
```



```
class Matrix
{
private:
    int ** data;
    int rows;
    int cols;
    bool isValidBounds( int i, int j )
    {
        return i>=0 && i<rows && j>=0 && j<cols;
    }
public:
    Matrix()
    {
        data=0;
        rows=cols=0;
    }
    Matrix ( int r, int c )
    {
        if (r<=0 || c<=0 )
        {
            rows=cols=0;
            data=0;
            return;
        }
        rows=r;
        cols=c;
        data = new int* [rows];
        for ( int i=0; i<rows; i++)
        {
            data[i] = new int[cols];
        }
    }
    Matrix( Matrix & ref)
    {
        if (!ref.data)
        {
            data=0;
            rows=cols=0;
            return;
        }
        rows = ref.rows;
        cols = ref.cols;
        data = new int* [rows];
        for ( int i=0; i<rows; i++)
        {
            data[i] = new int[cols];
        }
        for ( int i=0; i<rows; i++ )
        {
            for ( int j=0; j<cols; j++ )
            {
                data[i][j] = ref.data[i][j];
            }
        }
    }
    ~Matrix()
    {
        if (!data)
            return;
        for (int i=0; i<rows; i++ )
            delete [] data[i];
        delete [] data;
```

```
    data=0;
    rows=cols=0;
}
int & at( int i, int j )
{
    if ( isValidBounds( i, j) )
        return data[i][j];
    exit(0);
}
int getRows()
{
    return rows;
}
int getCols()
{
    return cols;
}
Matrix transpose()
{
    Matrix t(cols,rows);
    for ( int i=0; i<rows; i++)
    {
        for ( int j=0; j<cols; j++)
        {
            t.data[j][i] = data[i][j];
        }
    }
    return t;
}
Matrix add (Matrix & ref )
{
    if (!(rows==ref.rows && cols==ref.cols))
        exit(0);
    Matrix res(rows,cols);
    for ( int i=0; i<rows; i++)
    {
        for ( int j=0; j<cols; j++)
        {
            res.data[i][j] = data[i][j] + ref.data[i][j];
        }
    }
    return res;
}
void reSize(int nr, int nc)
{
    if (nr<=0 || nc<=0)
    {
        this->~Matrix();
        return;
    }
    int ** mat = new int*[nr];
    for ( int i=0; i<nr; i++)
    {
        mat[i] = new int[nc];
    }
    int i=0,j=0;
    while(i<nr && i<rows)
    {
        while(j<nc && j<cols)
        {
            mat[i][j] = data[i][j];
            j++;
```

```
            }
            i++;
        }
        this->~Matrix();
        rows = nr;
        cols = nc;
        data = mat;
    }
};
```