**Note:** Must implement all three type of constructors and one destructor in each of the following classes. Define the member functions out of the class using scope resolution operator.

## Task # 01 (10 points)

Define a class that will hold the set of integers from 0 to 31. An element can be set with the set member function and cleared with the clear member function. It is not an error to set an element that's already set or clear an element that's already clear. The function test is used to tell whether an element is set.

**Member functions:**

```
void small_set::set(int item); // Set an element in the set
void small_set::clear(int item); // Clear an element in the set
int small_set::test(int num); // See whether an element is in set
```

Sample usage: Main program

```
small_set   a_set;
a_set.set(3);     // Set contains [3]
a_set.set(5);      // Set contains [3,5]
a_set.set(5);     // Legal (set contains [3,5])
cout << a_set.test(3) << '\n';      // Prints "1"
cout << a_set.test(0) << '\n';     // Prints "0"
a_set.clear(5);  // Set contains [3]
```

## Task # 02 (10 points)

Write a **parity** class. This class allows the program to put any number of items into it and returns TRUE if an even number of items is put in and FALSE if an odd number is used.

**Member functions:**

```
void parity::put(int num); // Put another element
void parity::print(void); // Prints all elements that have been Put till now
void parity::delete(int num); // Delete only last elements that had been added using Put function
int parity::test(void); // Return TRUE(1) if an even number of puts have been done. Return FALSE(O) for an odd number.
```

## Task # 03 (10 points)

Write a class to implement a simple queue of real life. A queue is very similar to a stack except the data is removed in first-in-first-out (FIFO) order. Means elements will be added to the backside (last or end) of queue and removed from front side (beginning) of the queue only.

**Member functions:**

```
void queue::put(int item); // Insert an item in the queue
int queue::get(void); // Get the next item from the queue
```

Sample usage:

```
queue a_queue;
a_queue.put(1); // Queue contains: 1
a_queue.put(2); // Queue contains: 1 2
a_queue.put(3); // Queue contains: 1 2 3
cout << a_queue.get() << '\n'; // Prints 1, queue contains 2 3
cout << a_queue.get() << '\n'; // Prints 2, queue contains 3
```

## Task # 04 (20 points)

Create a class **2DArray** that contains these members: 2D array of type integers created statically and its size only i.e. int SIZE=5; int matrix[size][size].

Include the following member functions for **2DArray** class:

- All three type of constructors and one destructor
- Initialize(); // initialize 2D array with random values
- Print(); // print values of 2D array
- Transpose(); // take transpose of 2D array
- isSymmetry(); // check if matrix is symmetric or not.
- insertRowsCols (int x, int y) // insert x-rows and y-cols into existing 2D array
- deleteRowsCols (int x, int y) // delete x-rows and x-cols from existing 2D array
- 2DArray multiply (2DArray obj1, 2DArray obj2) // matrix multiplication of two object and return the result

Test your program by calling all these functions in main program.

## Task # 05 (10 points)

Create a class **student** which contains roll number, name and marks of 5 courses as data members. Write the data of ten students in **students.txt** file in following order RollNo, Name, Marks1, Marks2, and Marks3….etc. Your task is to read the data from file with following implementations;

**Member Functions:**
**readData()** // read data from file
**printData()** // Display all students data on console
**secondMaxMarks()** // find and display students having 2nd maximum grand total of all students
**calculateGrade()** // calculate grade of all students and output them in another file

**Note: must** create an array of **student** class objects for storing file data