You must write **Java** or **C++** on the first page of your answers!! You may answer the questions using either Java or C++, but you must use the same language for each question on the exam.

1. **(2 pts)** Write a function named `hasDups` that takes an array of strings and an int indicating the number of strings stored in the array. The function should return true if the array contains **any** string that occurs more than once in the array (otherwise it should return false). For example, if the function is called on the following array: `{"AA", "BB", "CC", "BB"}` the function should return true.

2. **(2 pts)** Write a **recursive** function named `countV` that takes three arguments: an array of integers and its size and an integer `v`, and returns a count of the number of times `v` appears in the list.  
   **Do not use loops, extra parameters, or global or static variables.**

The next question uses the following class:

```c++
// C++
class List {
private:
    struct Node {
        double value;
        Node *next;
    };
    // points to the first node
    Node *head;

public:
    List() {head=NULL;}
    void lastToFirst();
};
```

```java
// Java
class List {
    private class Node {
        double value;
        Node next;
    }
    // refers to the first node
    private Node head;

    public List() {head = null;}
    public void lastToFirst() {
        ...
    }
};
```

3. **(2 pts)** Define the public member function: `void lastToFirst()` that **removes** the last value from the list and inserts it at the beginning of the list. If the list is empty or has one element, the function should not alter the list. Do not create a new node or deallocate or change the value of an existing node. If the list contains 1.1, 2.2, 3.3 then `lastToFirst()` will change the list to: 3.3, 1.1, 2.2.
4. (4 pts) Declare and implement a class to represent a queue of integers, called IntQueue. Include the following functions in your class:

- a no-argument constructor that sets up an empty queue.
- a void function enqueue(x) that inserts a new value, x, at the rear of the queue (you should assume the queue is not full).
- an int function dequeue() that removes the value from the front of the queue, and returns it (you should assume the queue is not empty).

Note that a queue adds and removes elements in “First In, First Out” order. The element that has been in the queue the longest is the one that is removed next.

Use an array of integers of size 100 to store the values in the queue. Use integer variables front and rear to store the subscripts of the front and rear elements. You should not use any loops in your function definitions. When the front or rear subscripts reach the end of the array, they should “wrap around” back to the front of the array. In other words, if rear is 99, enqueue should store the next value at subscript 0.