EECE 230 INTRODUCTION TO PROGRAMMING
Lab Assignment 10

1. Implement the following:

a. Write a function `bool contains(string *A, int sizeA, string str)` that returns true if the array pointed to by `A` contains `str` and false otherwise.

b. Write a function `void intersection(string *A, int sizeA, string *B, int sizeB, string* &C, int& sizeC)` that determines the intersection of `A` and `B` and stores the result in `C`.

c. Using the functions described above, write a main that prints the names of students who registered both of EECE200 and EECE230. You can assume that the names of EECE200 students are found in a file named *EECE200.txt* and those of EECE230 students are found in *EECE230.txt*. In each of the two files, the first line contains the number of students who registered the course and the remaining lines contain the names. Note that the names read from the files should be stored in dynamic arrays.

2. Re-implement exercise 3 of assignment 6 by defining a class `Binary` as follows:

a. Two private variables: `int *sequence` and `int length` to store the binary representation as well as its length.

b. A default constructor that initializes `length` to 1 and sets the sequence to 0 (i.e. you have to allocate one memory location, store the address in `sequence`, and initialize the allocated memory to 0)

c. A parameterized constructor `Binary(string str);` that takes a string representation of the binary sequence and uses it to define the private variables. That is, it initializes `length` to the size of `str`, allocates the necessary memory for `sequence`, and sets the values accordingly. For example, if `str` is `"11001"` then this constructor would result in an object with the following values:

`length`: 5

`sequence`:

0	1	2	3	4
1	1	0	0	1

d. A member function `int computeValue()` that returns the integer value represented by the binary sequence.

e. A copy constructor that performs a deep copy of the variable `sequence`.

f. A destructor that deallocates the memory space pointed to by `sequence`.

Also, you need to write a main that reads a string from the user representing a binary sequence and outputs the corresponding value (note that the input string should be used to instantiate an object of type `Binary` and the member method `computeValue` should be used to compute the result).

3. **(Optional)** Arrays of fixed size have their limitations. The programmer has no control over the size of the data sets the user is interested in. Declaring very large arrays can be extremely wasteful of memory and declaring small arrays is problematic as the user might exceed the allocated size without being able to expand the array. These problems can be avoided by dynamically allocating an array of the right size, or reallocating an array when you need to expand it. Both of these are done by declaring an array as a pointer and using the `new` operator to allocate memory, and `delete` to free memory that is no longer needed.

   You are required to apply this methodology on exercise 2 of assignment 9. To do so, you have to declare the array of reservations as a pointer (`Reservation *res`) and keep track of its size (`int currentSize`) as well as the actual number of reservations made. Whenever the array becomes full (i.e. when the number of reservations made becomes equal to the array size), you have to call a function `resize` that doubles the size of the array while maintaining the reservations already made as follows:

   - Create a dynamic array whose size is double that of the original one
   - Copy the reservations stored in the original array to the first half of the new one
   - Delete the memory allocated to the original one
   - Store the base address of the new array in `res`
   - Update the value of `currentSize`

   Prototype of `resize`:

   ```
   void resize(Reservation* &, int&);
   ```

   Calling `resize`:

   ```
   resize(res, currentSize);
   ```