Pre-requisite homework — due 4/10

This homework will count for 25% of the weight of a regular homework.
To submit:

1. Create a PDF file with all your answers. The name of the file should be:
   `FirstName_LastName_PreReq.pdf`
2. Go to coursework and look under Assignments. Upload your file by going to the assignment called `Pre-requisite homework`.
3. The time when you upload the file is recorded.
4. The deadline for the homework is 4/10 at 12:50PM.
5. There is a grace period of 24 hours for this homework, with no penalty. **However, your homework will not be accepted after that.**

Problem 1

25 points. Assume that we have two arrays of `double` given by pointer `double *a` and `double *b`. Memory for both arrays has been allocated using:

```
a = (double *)malloc(1000000*sizeof(double));
b = (double *)malloc(1000000*sizeof(double));
```

Write a `C++` function that swaps the values of the two pointers. Turn in your `C++` code.

Problem 2

We want to implement a `C++` class for square lower triangular matrices. This class should have at least the following capabilities:

- Use a template argument for the type of the matrix entries. Assume that the type supports all arithmetic operations: `+`, `-`, `*`, `/`, such as `float` or `double`.
- It should accept a constructor with input argument `n`, the size of the matrix.
- The storage should be $n(n+1)/2 + O(n)$ where n is the matrix size.
- You should define an operator `()` to access and modify entries in the matrix. The operator should take as input a row `i` and a column `j`.

1. 25 points. Write the `C++` class. Turn in your code.
2. 15 points. We want you to demonstrate that you know how to write correct code, which includes knowing how to test your code. The instructions are a bit vague because we want to see how much you can figure things out by yourself. This matrix class is very simple so the tests are somewhat unnecessary here. So the goal is to demonstrate that you understand the concept of testing a code and can apply it to this simple case.

   Describe the operations that are supported by your class. Explain which tests you want to write to check whether your class indeed supports the features you want correctly.
3. 25 points. Write and turn in code that implements these tests. Run your code. Did your class pass all your tests? Are there situations where your code would fail to behave in the expected fashion?
4. 10 points. List by order of importance some additional features you would like to add to your class. You do not have to implement anything.

Total number of points: 100