<u>Homework #1: 1D Closest Pair (100 pts)</u>

For this assignment, you must submit **C++** source code that solves the **1D Closest Pair** problem using two algorithmic approaches, i.e., 1) **brute force** and 2) **recursive divide and conquer**.

The **1D Closest Pair** problem is defined as follows:

INPUT: A set of **N** random integers representing points on a line.

OUTPUT: Two points (**p_1, p_2**) representing the *closest pair* within the input set.

For example, input set { 0, 5, 10, 11, 15, 20 } would have output **(10, 11)**.

Your C++ source code **must** meet the following *requirements*:

- *Compile* and *execute* on the provided Linux virtual machine or OSU's EECS server.
- Get input set from user (**stdin**)
  - Input consists of N random integer values, one per line.
  - Does **NOT** prompt user in any way
- Solve 1D Closest Pair problem using the **brute force algorithm**.
- Solve 1D Closest Pair problem using the **recursive divide and conquer algorithm**.
- Write output to **stdout**
  - Output is closest pair of points (**p_1, p_2**) within input set.
  - Output is **labeled.**

EXAMPLES:

```
//Example 1: get input from terminal user
UNIX> ./hw1
0
5
10
11
15
20
<CTRL-D>
brute force closest pair:                    (10,11)
recursive divide and conquer closest pair: (10,11)
```

```
//Example 2: get input using file redirection
//      NOTE: both commands assume input.txt already exists
UNIX> cat input.txt
100
30
0
20
50
10
40
70
90
60
80
31

UNIX> ./hw1 < input.txt
brute force closest pair:                    (30,31)
recursive divide and conquer closest pair: (30,31)
```

HINTS:

- start early
- solve 1D closest pair on paper before writing any code
- work incrementally
- consider even and odd **N**
- consider boundary conditions

- while(cin >> n)

- `vector <int>`
    - `push_back( )`
    - `size( )`

- `sort( )`

- pass by reference
    - `void fx( int &x )`

- `UNIX> g++ homework_01.cpp`