CS 60, Lab 6
Template functions and classes

Start a new email. Include the names of both members of your team and any additional information or code as instructed below. At the end of the lab, please email your lab report to cs60labtues@gmail.com or cs60labthurs@gmail.com **as appropriate with the subject line "CS60 Lab6 lastname lastname", with the last names of both members of your group.**

Please do not write on this sheet

Alternate typist and reader each week.

Part A: Array template toolkit.
Create a new project.

Let's use the array `arr` below as our example:

6	2	3	2	6	2

1. Define the template function
```
template <typename T1, typename T2>
int count_exact(T1 a[], T2 size, T1 find)
```
that returns the number of times `find` occurs in `a`. For instance, `count_exact(arr, 6, 2)` would return 3. Test your function in your main() *using an array of ints, and an array of strings*. Write a comment explaining which operations the types T1 and T2 need to support for your code to work. **Copy your code into your lab report**. **Be sure to get your lab checklist initialed before moving on.**

2. Define the template function
```
template <typename T1, typename T2>
int count_range(T1 a[], T2 size, T1 low, T1 high)
```
that returns the number of times an item between low and high (inclusive) occurs in a. For instance, `count_range(arr, 6, 2, 4)` would return 4. Test your function in your main(). Write a comment explaining which operations the types T1 and T2 need to support for your code to work. Test your function in your main *using an array of ints, and an array of strings*. **Copy your code into your lab report**. **Be sure to get your lab checklist initialed before moving on.**

Part B: n-point template class. Create a new project.

1. In this problem, you'll make a **template class** for an n-dimensional point; that is, points with 2, 3, 4, etc. coordinates, up to 10. For example, you could have any of the following:

   - A two-dimensional point of ints, like (2, 6)
   - A five-dimensional point of strings, like ("Hello", "how", "are", "you", "today")

Your class should have private members as follows: an array of size 10, and a variable that holds the dimension of the point. Also include:

- Three constructors: one constructor that takes 0 arguments, one that takes in just the dimension of the point, and one that takes in the dimension of the point and the initializing array.
- A single getter that returns the ith item in the tuple
- A getter for the size

Test your functions in your main(). **Copy your code into your lab report**. **Be sure to get your lab checklist initialed before moving on.**

2. Overload the = operator for your class as a member. Write a comment explaining which operations the type variable needs to support for your code to work. Your = function should return the object you're inside of (not a pointer to it). Test your function in your main(). **Copy your code into your lab report**. **Be sure to get your lab checklist initialed before moving on.**
3. A += member function for the class. If the two n-points have different size, the resulting thing should have the larger dimension; e.g. ("hello", "there") += ("I", "am", "Sam") would change the first object to ("helloI", "theream", "Sam") (Note that this comes from ("hello"+"I", "there"+"am", "Sam"))

---MINIMUM STOPPING POINT---

4. Overload the following operators for your class: +, ==. Note that (2, 6) + (1, 5) = (3, 11). These should be implemented as non-members. Write a comment explaining which operations the type variable needs to support for your code to work. Test your function in your main(). **Copy your code into your lab report**. **Be sure to get your lab checklist initialed before moving on.**

5. Implement a non-member `distance` function that returns the distance between two points. For two dimensions, the distance formula is as below; what would it be for three dimensions?
   $dist((x, y), (a, b)) = \sqrt{((x - a)^2 + (y - b)^2)}$
   Distance should only require *, +, -, and sqrt to exist for the type variable. You will need to overload – for strings: It should take in two strings and return an int (the distance between the encodings of the first characters of the strings). Test your function in your main() using Rational points. **Copy your code into your lab report**. **Be sure to get your lab checklist initialed before moving on.**

Part C. Continuing the Array template toolkit

1. Define the template function

```
template <class T1, class T2>
T1 frequent(T1 a[], T2 size)
```

that returns the item in `a` that occurs the largest number of times. For example, frequent(arr, 6) would return 2. Your function should work for any array of size up to 50 items; use an assert to enforce this. Test your function in your main(). **Copy your code into your lab report**. **Be sure to get your lab checklist initialed before moving on**