By: Andres Cabrera, Mike Dong

CSCI 60

Lab Report 7

1.)

```
//1, Define the template function
//returns the number of times find occurs in a. For instance, count_exact(arr, 6, 2) would return
3.
template <typename T>
int count_exact(T a[], int size, T find) {
	int count = 0;
	for (int i = 0; i < size; i++) {
		if(a[i] == find) {
			count++;
		}
	}
	return count;
}
//Write a comment explaining which operations the type T needs to support for your code to
work:
//The type T needs to support the equality operator (==) for comparison in order for the code to
work.
```

2.)

```
//2.  Define the template function
//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.
template <typename T>
T frequent(T a[], size_t size) {
	T most_frequent = a[0];
	int max_count = 1;
	size_t capacity = 50;
	if (size <= capacity) {
		for (size_t i = 0; i < size; i++) {
			int count = 1;
			for (size_t j = i + 1; j < size; j++) {
				if (a[i] == a[j]) {
					count++;
				}
```

```
                        }
                        if (count > max_count) {
                                max_count = count;
                                most_frequent = a[i];
                        }
                }
        }
        return most_frequent;
}

3.)
template <typename T>
class NPoint {
private:
        static const size_t MAX_DIM = 10;
        T coords_[MAX_DIM];//hold the coordinates of the point
        size_t dim_;
public:
        //0-argument constructor
        NPoint() : dim_(0) {}
        //Constructor that takes in just the dimension of the point
        NPoint(size_t dim) : dim_(dim) {}
        //Constructor that takes in the dimension of the point and an initializing array
        NPoint(size_t dim, T arr[]) : dim_(dim) {
                for (size_t i = 0; i < dim_; i++) {
                        coords_[i] = arr[i];//coords_ is a static array that holds the coordinates of
the point, while arr is the array that is passed to the constructor to initialize the coordinates.
                }
        }
        T get_coord(size_t index) const;
        size_t get_size() const;
        void operator =(const NPoint<T>& rhs); // assignment operator
};
//A single getter that returns the ith item in the tuple
template <typename T>
T NPoint<T>::get_coord(size_t index) const {
        return coords_[index];
}
//A getter for the size
template <typename T>
size_t NPoint<T>::get_size() const {
        return dim_;//PS: dim_ is the member variable that holds the dimension of the point, so it
is the correct value to return for the size of the point.(dim_ is the size)
}
```

4.)

```
template <typename T>
void NPoint<T>::operator=(const NPoint<T>& rhs) {
	if (this == &rhs) return;//not returning *this because the function is declared as void
	dim_ = rhs.dim_;
	for (int i = 0; i < dim_; i++) {
		coords_[i] = rhs.coords_[i];
	}
}
```