**ENGS 65 Homework 1: ADTs**

In designing a 3-D Vector data type, I made the following overarching design choices:

1. Data Structure
   * Private instance variables for x,y,z components
     1. Alternative: public instance variables
     2. This design choice was made in order to uphold the principles of information hiding. The user should be able to see vector dimensions, but not able to manipulate them once the vector is created. This allows any further programs that might depend on a vector to always be the same vector to remain stable.
   * All objects returned should never be references to the object itself, but should be dummy copies
     1. Alternative: passing a pointer to the object itself
     2. This is for a similar reason as above. Information hiding and employing private types ensures that the user can never manipulate the actual data that represents the vector itself, once created.
2. Object Operations
   * Create member functions for vector operations
     1. Alternative: let the user create their own
     2. I chose to implement a series of member functions in order to reinforce the encapsulation of the 3D vector class. The alternative would require compromising on the principles of encapsulation and information hiding, as it would require user access to private data, and would limit the reusability of the code for complex vector operations. Instead, I included a number of common vector operations such as the dot product, cross product, magnitude calculation, and the other operations included in the code, so that the class would ensure code portability and stability.
   * Override +/- operators
     1. Alternative: let user do it manually
     2. This design choice also serves to hide information, and encapsulate all relevant manipulations for the data.
   * Return vector dimensions in array form
     1. Alternative: Have user access each dimension individually
     2. This was chosen because returning the vector components in an array facilitates access to every element at once, and more importantly, allows for the user to represent a vector object as a column vector in a matrix. This is a common convention in linear algebra, and allows for easy manipulation of data, as well as integration with MATLAB and other matrix-oriented programs.
3. Implementation in C++
   * See attached C++ pseudocode for a rough conceptual implementation of the above ideas. **NOTE:** **Assume all arguments to functions are passed by reference, and that all function and get/set return values point to copies, and not actual data.**

SEE ATTACHED CODE