For the first problem I utilized the STL map and unordered map, together with a Student class which carries a student ID (int) and Homework ( <int , Item>). Homework is an unordered map which holds the homework item’s key and the Item itself (every Item has an isChecked property).

My first thought was that every Student is unique so I figured I could use a Set for the students container. I overrode the compare function so my set could be sorted, and inserted 3 students. But as I tried to look one up I realized that the find function takes a key of type Student, and if I wanted to search for a student with a given name, that would not be easily possible (Maybe something could be done, like overriding the find() but I couldn’t figure it out and it also doesn’t seem like the right way). If we wanted to go through all of them or search by integer ID (with id’s mapped like array indexation), it could work by getting the iterator and iterating to the Nth element, but that is inefficient as it goes through all the previous elements.

So I scrapped that idea and went with an ordered map for the Students. I figure that when working with students, we would want them to be somehow sorted, because we are sure to have other operations on them. This also provides guaranteed lookup times of O(log N) and is a bit lighter on memory, since we don’t store a hash table.

For the homework container on the other hand I chose an unordered map. Unlike the ordered map, the worst-case look up time is O(N) but is generally much faster – usually O(1). But that requires a good hasher, and is also implementation defined, so no guarantees. I chose it because when testing on a win7 64bit machine with visual studio it was quite faster than the ordered map. I also figured that the homework would just be looked up and checked and not printed or accessed in order, but it’s equally plausible that we would need the homework sorted.

So in conclusion, every container has its perks and disadvantages. Set are good but not for our task, because they do not provide easy search of a class element with a given property. Ordered and unordered Map could both be used for a given task, but it all depends on what we are after.

If we want speed and don’t mind the memory overhead, and also have a good hasher function that does not produce many collisions, unordered map (hash table) is the way to go.

If we need sorted data for printing and accessing, predecessor/successor to element or guaranteed lookup, insert and delete speed (for insert and delete we have to factor in the rebalance), we should go with the ordered map (balanced BST).

P.S. There are 3 students, every student has 2000 homework items, every third item is checked.