README
Noriaki Nakano
nnakano@ucsc.edu
Assignment 2

Problem 1:

My approach to this problem was to think of it has 3 object type:
institute, classroom, and student. To put these classes together I made a linked-list called PointerList, which utitilizes a template to adapt to many kinds of objects.

This pointer list takes in pointer of objects and stores them in a list.
Institute holds 2 PointerLists, one that holds the students, and one that holds the classrooms.

By making this PointerList, it made managing the classroom's and student's much easier than trying to
implement it with an array.

The one problem that I did encounter was the fact that I needed to be careful of memory leaks because I was
handling so many pointers. But, I eventually did manage to get rid of all memory leaks, and this can be
proven through the valgrind, or using my make file: make check1.

The one thing to note about PointerList<T>, is the fact that the implementation is inside the header file.
This is nessarary PointerList itself, and all the functions inside it, are template functions. This means
that on compilation, if I left the template functions inside the .cpp file, I get a undefined reference link

error. This is because template functions are not really defined functions, but are only just "templates", and they are not acutally a function definition. Therefore I need to put it into my header file in order to avoid that link error.

When you run the first program(problem 1), it will automatically create a insitution with 256 students, and 8 classrooms, in which the students are assigned to random classrooms. The program will print out all the student information with which class he/she is in, which is followed up with one print showing how many seats are left in the institution.

Then the user will be asked to input a student number from 0 - 255 and a classroom number > 0, which then the program takes in and creates a new institute with those parameter. The program will then proceed to do the exact same thing it did for the default institution run.

Problem 2:

My approach to this problem was incredibly simple. FIrst, create 3 classes: scanner, product_type, and product. The implementation for this problem was so simple because I reused the PointerList<T> from the previous problem. Scanner holds the inventory which is a PointerList<product_type> of 256 product types and product_type each has a PointerList<product> of 8 different colors. Scanner also holds the main function.

When the program runs very similarly to problem # 1, prints out all the information, and ask for an user input as well.

Problem 3:

I had issues with this part, because as much as I see, I though that a linked-list is better for imp lementing a
data structure like this. To be more specific, I t hought that a ordered list is much more superior i n this
case then using a binary tree-like structure. An o rdered list can just stack whatever is the smalles t on the top
and largest in the bottom, making the less workloa d performances much easier to access. I tried to i mplement this
but inheritance and other features got in my way an d I couldn't achcieve it after all.