Program #1
CS 163 Data Structures

Submit your program to the D2L Dropbox
Email a backup copy to karlaf@pdx.edu

LATE work will be accepted – but ONLY within a week of the original due date.
We do not accept late work beyond that. There are no exceptions.

Scope: When beginning with this project, the first thing to keep in mind is that we only have approximately 2 weeks to complete each assignment. Therefore, it is critical that you focus on a limited scope, with an emphasis on the class(es) and data structures. Your program **DOES** need to compile and run, but you won't be building complete applications. *You will be primarily graded on your use of classes, member functions, arguments, data structures, pointers and the efficiency of your code.* You will need a main, but it should focus on thoroughly testing out your class member functions rather than being a complete application program. Imagine that there is another software engineer who will be building the actual application software. Your job is to build the Abstract Data Type (ADT) and test it!

Therefore, focus on how to design classes that are well structured and efficient and on the required data structures. Limit the development of the application that uses the data structures. Of course, your user interface must be clear enough for us to test your program and we must be able to thoroughly test all features. **And, it is not appropriate to hard code in the test cases - all tests should be interactive with the user.** This first program of the term is an exercise in building, traversing, and destroying linear linked lists.

Background Information: Over spring break, my green house was delivered. I was really excited to start using it, but first we had to build it. Hard to believe but there are thousands of parts. So, while my husband was deciphering the instructions (which have no words!) I had to organize the parts so that at each step we would have the correct materials. For example, Step 1 needed two 7448's, two 7449, and four 447 screws. Step 2 needed two 7448's, two 7449's, two 7451, eight 411 bolts and eight 412 nuts *(why aren't these numbers the same?).* There are 91 steps! So far we have made it to step 13 and I've only found the parts up through step 16. Sounds fun? Not really. Only some of the parts are labeled and the hardware is in big bags all together with no organization at all. I've experienced the same type of issue building furniture from IKEA but at a much smaller scale. Your assignment is to build a LLL to help organize the parts.

Programming Assignment: With program #1, you will be creating a program that will assist organizing do-it-yourself building projects (such as furniture purchased from IKEA). *IKEA should give this program away with every purchase!*

Your ADT will read in from an external data file all of the parts that came with the product. Name this file **parts.txt**. Separate the fields by colons and end the sequence of fields with a newline. Do not assume that the parts in the file are sorted. Have at least the following information for each part:

1. Part Number (e.g., 7449)
2. Description and/or size (e.g., 1 inch bolt)
3. Quantity provided (e.g., 411)
4. The number of steps that use this part
5. A list of the steps where the part is used (steps 1, 4, and 91)
6. For example, the format of the file should be: 7449**:**1 inch bolt**:**411:3 steps: 1:4:91**n**

Create a class or struct for an individual part. To start I would recommend a struct call Part.

Your ADT will then take this information and insert it into a LLL, sorted by part number. Now, when the user needs four 447 screws, they can easily be found. It will also make a second LLL, sorted by step number. Now the user can ask for what parts are needed for step 1, and the ADT will display that information.

The two most important parts of this assignment are implementing the data structures and creating the ADT to manage the data structure. The following class is a suggested class interface for this first assignment:

```
class DIY   //For Do It Yourself!
{
    public:
        DIY ();    //constructor should load the parts and build the two LLLs
         ~DIY ();
        int Display_Parts();  //display all parts, in order of part number
        int Display_Steps(); //display all steps and the parts needed for each step
```

```
        //Give the ADT a step number, and it will provide the client with the parts
        int Next_Step(int step_number, Part parts_list[], int & num_parts);
    private:
        node * part_head;
        node * step_head;
};
```

Things you should know...as part of your program:

1) Do not use statically allocated arrays in your classes or structures. All memory must be dynamically allocated and kept to a minimum!
2) All data members in a class must be private
3) Never prompt and read from the user when inside a class member function
4) Never output error messages from a class member function
5) Global variables are not allowed in CS163
5) **Do not use the String class! (use arrays of characters instead!);** however you may use the cstring library of strlen, strcpy, strcmp
6) Use modular design, separating the .h files from the .cpp files. Remember, .h files should contain the class header and any necessary prototypes. The .cpp files should contain function definitions. You must have at least 1 .h file and 2 .cpp files. **Never implement functions in your .h file! And, never "#include" .cpp files!**
7) Use the iostream library for all I/O; do not use stdio.h.
8) Make sure to define a constructor and destructor for your class. Your destructor must deallocate all dynamically allocated memory.
9) Remember that 20% of each program's grade is based on a written discussion of the design. *Take a look at the style sheet which gives instruction on the topics that your write-up needs to cover.*