ECE 2420 Programming Exercise #7

(Associative Arrays: Simple Implementation)

# Overview

In this programming exercise the coder will create an associative array data structure. An associative array accepts a key and returns an associated value. It is an extremely useful data structure which may also be referred to as a map, dictionary, or symbol table. The first phase of this exercise deals with the interface and creating the correct behavior of the data structure. Follow on phases will make the data structure more efficient.

As a practical use of the data structure, the coder will construct a database of electronic components. Each of the components will have a number of features/specifications, and each feature/specification will have a value. For example, here are a couple of resistors that may be found in our parts database:

Name: Resistor#1

 Temp coefficient: 50ppm/C

 Frequency response: 10MHz

 Power dissipation: 1W

 Max temperature: 100C

 Max voltage: 3.3V

Name: Resistor #2

 Temp coefficient: 40ppm/C

 Frequency response: 20MHz

 Power dissipation: 5W

 Max temperature: 120C

 Max voltage: 5V

# Parts Catalog Architecture

The parts catalog is an associative array of associative arrays. The outermost array is keyed by part name, (a string in this case). The content of the outer associative array is the inner associative array. The inner associative array is keyed by a parameter name and the content is the parameter value, (both strings).

When iterating through the structure it should be possible to print all part names in order and then print each parameter and associated value for that part in order of parameter name. It should also be possible to extract any given parameter value by using the get methods.

# Programming Concepts

This exercise covers many programming concepts including inheritance, deep copy, constructor types including default and delete, operator overloading, pass by ref and by val, return by ref, stack vs heap, linked lists, templates, reference counting pointers, lambdas, closures, functors, and abstract base classes.

# System Requirements

The design must use the provided interface header verbatim. This will allow automated testing of the design you produce. See grading rubric for specific system requirements and associated grade values.

# Turn-in Procedures

Turn in all source code via a git push to your 2420 repository by 11:59p.m. on <DUE DATE>

Grading Rubric

(ECE 2420 PEX7)

|  |  |  |
| --- | --- | --- |
| Requirement / Criteria | Available Points | Student’s Score |
| Uses base class interface verbatim | 10 |  |
| Can be created on the stack | 10 |  |
| Is copy constructible and assignable | 10 |  |
| Correctly deep copies in insert | 10 |  |
| Modification after “get” modifies content in structure | 10 |  |
| Modification in “forEach” modifies content in structure | 10 |  |
| Method “forEach” iterates in order | 10 |  |
| Inserting/deleting at head of list functions correctly | 10 |  |
| “getting” and “deleting” non-existent key functions correctly | 10 |  |
| System is free of memory leaks | 10 |  |
| Total | **100** |  |