COEN 79L - Object-Oriented Programming and Advanced Data Structures

Lab 4

- Sub-project 1: **Keyed Bag**

Another way to store a collection of items is in a **keyed bag**. In this type of bag, whenever an item is added, the programmer using the bag also provides a string called the `key`.

To this end, two arrays are used:

```
key_type keys[CAPACITY];    // The array to store the keys
value_type data[CAPACITY];  // The array to store items
```

Each item added to the keyed bag must have a unique key; two items cannot have the same key. So, the insertion function has the specification shown here:

```
void keyed_bag::insert (const value_type& entry, int key);
// Precondition: size( ) < CAPACITY, and the bag does not yet contain any
// item with the given key.
// Postcondition: A new copy of entry has been added to the bag, with the
// given key.
```

When the programmer wants to remove an item from a keyed bag, the key of the item must be specified, rather than the item itself. The keyed bag should also have a boolean member function that can be used to determine whether the bag has an item with a specified key.

For this project, do a complete specification, design, and implementation of a keyed bag.

- Sub-project 2: **Polynomials**

For the second project, specify, design, and implement a class for polynomials.
A polynomial is an arithmetic expression of the form:

$$a_0 + a_1x + a^2x_2 + \cdots + a^kx_k$$

The highest exponent, k , is called the degree of the polynomial, and the constants are the coefficients.

The class may contain a static member constant, `MAXDEGREE`, which indicates the maximum degree of any polynomial. (This allows you to store the coefficients in an array with a fixed size.) Spend some time thinking about operations that make sense on polynomials. For example, you can write an operation that adds two polynomials. Another operation should evaluate the polynomial for a given value of x.

You may add additional functions to the header files if you wish to write more to help you complete the assignment. You may NOT alter the names or structure of any of the

functions or attributes that are already within the header files. If you do add your own functions, make sure to update the header file's comments to discuss those functions' preconditions and postconditions. You will also be required to include an invariant in each of the implementation files that you write.

When doing your implementation, pay careful attention to the preconditions and postconditions given in the corresponding header file. Make sure to follow and enforce these conditions. In `poly.h`, there are very specific instructions about how a polynomial should be printed. Make sure to follow these and ask for clarification if you are confused, as the format of what the output operator prints will be part of the grade.

The files that you submit should be named exactly `keyed_bag.h`, keyed_bag.cpp, `poly.h`, and `poly.cpp`. The namespace used for the contents of these files should be exactly `"coen79_lab4"`. You should write test files for both of the above classes to test all of their features thoroughly before submitting your solutions.