Section Exercises 8: Function Pointers, Hashing and the Symbol Table

Problem 1: Function Pointer Quiz

Here is the prototype for `IntFunction`, which takes no arguments, and returns an integer:

```
int IntFunction(void);
```

What is the difference between the following two lines of code? Assuming that the function calls are correct, what would the prototypes be for these two functions?

```
FunctionOne(IntFunction());
FunctionTwo(IntFunction);
```

Problem 2: `void *` Practice

You have an unordered binary tree. Assume there is an unspecified pointer type stored as the value in each tree node:

```
typedef struct _Node {
   void *value;
   struct _Node *left, *right;
} Node;
```

a) Write a function `Max` which takes such a tree and a comparator function and returns the maximum value in the tree (actually returns the pointer to that value). Note that the tree is not ordered like a binary search tree. The comparator function will return an integer which is less than zero, equal to zero, or greater than zero, precisely when the first value is less than, equal to, or greater than the second. (Just like `StringCompare`.) Here is the typedef for this class of functions:

```
typedef int (*cmpfn) (void *element1, void *element2);
```

b) You have a tree where the values are pointers to a student information struct such as this:

```
typedef struct {
   string firstName, lastName;
   char middleInitial;
   int studentId;
} Student;
```

You want to order students by alphabetically by last name, first name, middle initial. Write a comparator function `StudentCompare` which given two student pointers will return the correct value to indicate where the students are in relation to each other.

Show how to find the alphabetically last student in a tree using this comparator and a call to the `Max` function above.

Problem 3 : Hashing performance under collision

How are two different keys mapping to the same bucket handled in a hash table? In the worst case, what does this do to the Big-O of Lookup? How about the average case?

Problem 4 : Hash function analysis

Comment on the effectiveness and appropriateness of the followed suggested hash functions:

a) The table has 2048 buckets. The search keys are peoples' names.
hash(key) = Ascii value of the first letter of key mod 2048

b) The table has 1000 buckets. The search keys are integers in the range 0..999.
hash(key) = (Product of the digits of key) mod 1000

c) The table has 1000 buckets, the search keys are integers in the range -32768 to+32767.
*hash(key) = (key*RandomInteger(1,100)) mod 1000*

Problem 5 : Symbol Table Implementation Choices
Would you choose the hash table or the binary tree implementation of the symbol table if:

a) you wanted be able to print the contents of the table in **any** order?

b) you wanted to be able to print the contents the table in **sorted** order?

c) you wanted to be able to find the value with the alphabetically last key?

d) you expect to insert items into the table in sorted order?

Problem 6: Mapping functions, Iterators, and Symbol Tables

(a) Using the `MapSymbolTable` function defined in the text, write a function

```
string FindLongestKey(symtabADT table);
```

which returns the longest key in the symbol table. For example, suppose that you have just entered the following bindings into a symbol table called `elementTable`:

```
Enter(elementTable, "Hydrogen", "H");
Enter(elementTable, "Helium", "He");
Enter(elementTable, "Beryllium", "Be");
Enter(elementTable, "Boron", "B");
Enter(elementTable, "Carbon", "C");
Enter(elementTable, "Klepon", "Kl");
Enter(elementTable, "Plummerium", "Pl");
```

With these values in the symbol table, calling `FindLongestKey(elementTable)` should return `"Plummerium"`. If there are several keys that are the same length but longer than any other key, `FindLongestKey` may return any of them. For example, if you added the entry

```
Enter(elementTable, "Harrylaium", "Ha");
```

to the table, your function could return either `"Plummerium"` or `"Harrylaium"` because they have the same length.

(b) Solve the same problem using an iterator instead of a mapping function.

(c) Which method (a or b) seemed more natural? In general, when would you use a mapping function and when would you use an iterator?