

Arrays and User-defined Functions:**Simple Inventory**

For this project, you will implement a program that loads a list of inventory information and then performs searches on it.

Your program will use a single array to store item descriptions read from an input file. That array will then be searched for matches to specific descriptions, and the results of those searches will be reported in an output file.

Input specification and sample input data:

This program will use two input files, one containing the information about what items are currently in the inventory, and the other specifying items that are to be looked up in the inventory list. The first input file for this program is named "ItemDB.txt". The input file begins with two lines that just label the remaining contents of the file. Each of the remaining lines contains a description, followed by a tab, followed by zero or more spaces, and concludes with a unit price and a newline character. There will never be more than 50 lines of item data in the file.

Item	Price
Cut Green Bean 14.5 oz	1.05
Chili Hot Beans 15 oz	0.65
Cut Wax Beans 14.5 oz	0.69
Lima Beans 15 oz	0.97
Cut Green Beans 14.5 oz	0.75
Lite Peaches 15 oz	1.09
Whole Green Beans 14.5 oz	0.75
Whole Kernel Corn 14.5 oz	0.75
Sweet Peas 15.25 oz	0.75
Red Kidney Bean	0.79

The second input file is named "SearchData.txt". Each line of this file contains the description of an item, followed by a newline character. There is no specified limit on the number of lines in this file.

```
Thin Slice Swiss Cheese 8 oz
Cut Green Bean 14.5 oz
American Singles 12 oz
Chili Hot Beans 15 oz
Cut Wax Beans 14.5 oz
Choice Chuck Steak 16 oz
Lima Beans 15 oz
Lite Peaches 15 oz
Whole Green Beans 14.5 oz
Whole Kernel Corn 14.5 oz
Sweet Peas 15.25 oz
Choice Flank Steak 16 oz
```

Output specification and sample output data:

The output file must be named "Results.txt". An output file, which corresponds to the input data given above, is shown below. The first section of the file is just a numbered listing of the item descriptions given in the first input file, with labels and separators as shown in the sample output file below.

The second section contains the results of the searches that were performed for the items described in the second input file. If the item description was found in the inventory list, the array index at which it was found is reported. If not, then the string "Item not found" is written, followed by the description. Labels and separators should match those shown in the sample output file below.

```
Database contents:
```

```
Index   Description
```

```
-----  
0:  Cut Green Bean 14.5 oz  
1:  Chili Hot Beans 15 oz  
2:  Cut Wax Beans 14.5 oz  
3:  Lima Beans 15 oz  
4:  Cut Green Beans 14.5 oz  
5:  Lite Peaches 15 oz  
6:  Whole Green Beans 14.5 oz  
7:  Whole Kernel Corn 14.5 oz  
8:  Sweet Peas 15.25 oz  
9:  Red Kidney Bean  
-----
```

```
Search Results
```

```
-----  
Item not found: Thin Slice Swiss Cheese 8 oz  
0  
Item not found: American Singles 12 oz  
1  
2  
Item not found: Choice Chuck Steak 16 oz  
3  
5  
6  
7  
8  
Item not found: Choice Flank Steak 16 oz
```

Specific requirements for arrays and functions:

Your program must use a single array of strings to hold the item descriptions supplied in the first input file. The dimension of the array must be at least large enough to handle the worst case specified above.

Since there's no specified limit on the number of descriptions that will be given in the second input file, your program must read them and perform the searches one-by-one.

Your program must make good use of user-defined functions. In particular, you should have at least one function satisfying each of the following requirements:

- a function to initialize each element of the array to some distinctive value, before the data is read
- a function to read the first input file and put the descriptions there into the array
- a function to determine whether a given description occurs in the array or not; this should return the array index if the description is found and some sort of error indicator if the description is not found
- a function to handle reading the descriptions out of the second input file, and use the function described above to perform the searches
- a function to write out the contents of the array

Some of your functions may satisfy one of these requirements and also do more than is described above; that's OK.

Your program should use the appropriate parameter-passing mechanisms. In particular, you should not pass a parameter by reference unless it is logically necessary to do so. Otherwise, pass by value or pass by constant reference.

Your solution must absolutely not use any global variables! It is perfectly OK to use global constants and to make your function declarations global.

Documentation requirements:

Since your solution must be checked to verify the requirements above, we will also check to see if you have included good comments. Read the Programming Standards page on the course website for guidelines. In particular, you must include a header comment for each of your user-defined functions that include the information shown in the sample function header comment.

Submitting your program:

You will submit this assignment to the Curator System (read the *Student Guide*), and it will be graded automatically. Instructions for submitting, and a description of how the grading is done, are contained in the *Student Guide*.

You will be allowed up to ten submissions for this assignment. Use them wisely. Test your program thoroughly before submitting it. Make sure that your program produces correct results for every sample input file posted on the course website. If you do not get a perfect score, analyze the problem carefully and test your fix with the input file returned as part of the Curator e-mail message, before submitting again. The highest score you achieve will be counted.

The *Student Guide* and other pertinent information, such as the link to the proper submit page, can be found at:

<http://www.cs.vt.edu/curator/>

Pledge:

Each of your program submissions must be pledged to conform to the Honor Code requirements for this course. Specifically, you **must** include the following pledge statement in the header comment for your program:

```
// On my honor:
//
// - I have not discussed the C++ language code in my program with
//   anyone other than my instructor or the teaching assistants
//   assigned to this course.
//
// - I have not used C++ language code obtained from another student,
//   or any other unauthorized source, either modified or unmodified.
//
// - If any C++ language code or documentation used in my program
//   was obtained from another source, such as a text book or course
//   notes, that has been clearly noted with a proper citation in
//   the comments of my program.
//
// - I have not designed this program in such a way as to defeat or
//   interfere with the normal operation of the Curator System.
//
// <Student Name>
```

Failure to include this pledge in a submission is a violation of the Honor Code.