Sabancı University

Faculty of Engineering and Natural Sciences
CS204 Advanced Programming
Spring 2022

Homework 2 – Guessing Secret Number with Linked Lists

Due: 29/3/2023, Wednesday, 21:00

PLEASE NOTE:

Your program should be a robust one such that you have to consider all relevant programmer mistakes and extreme cases; you are expected to take actions accordingly!

You can NOT collaborate with your friends and discuss solutions. You have to write down the code on your own. Plagiarism and homework trading will not be tolerated!

Introduction

In this homework, you are going to implement a number guessing game with user interaction. The objective of the game is to reach a secret number by trying a sequence of guesses. The guesses are not completely uneducated ones. An oracle function (provided to you) will give the program some feedback for each guess, and your program will process this feedback in several linked lists by keeping possible and impossible alternatives. This way, the player is helped to consider that feedback while he/she is coming up with his/her next guess.

In order to understand the program flow and the data structures that you have to use, let us explain the game and its playing rules shortly. The program will keep a 3-digit secret number of which all digits are distinct (i.e. between and including 012 and 987 and all digits are different than each other). The secret number is associated with a `game id` that you (the player) enters. In this way, the player will not know the secret number. The player will try to guess it. Once the player enters a guess, your program will call `get_the_feedback` function, which is provided to you as source code. This function returns a feedback for each slot of the guess as **R** (the digit does not appear in the secret number), **Y** (the digit appears in the secret number but in a different slot; not in the tried slot), or **G** (the slot has been guessed correctly). Since the digits of the slots are all distinct, there is no possibility of having both Y and G feedback for a particular slot. Your program will not show the feedback to the player, but it will process six linked lists to keep *possible* and *impossible* digits for each slot using the feedback. At each round, these lists will be updated and displayed so that the player will make use of them for the next guess. The program will end when

there is single solution for each slot in the linked lists (not necessarily after entering correct guess). The structure of the linked lists, manipulation operations and other details for the game playing are explained later in the document.

Data Structure

The main data structure of the program is one-way integer linked lists. You should use the following node structure for the linked lists.

```
struct node
{
    int digit;
    node * next;
};
```

In the program, you will need a total of six such linked lists; two for each slot. For each slot, one linked list will be used for the *possible* digits, and the other is used for the *impossible* ones. You may use two built-in arrays with three `node` pointers each. These node pointers are used as the heads of the possible and impossible linked lists of the slots as shown in Figure 1. The `Impossibles` array (to the right of Figure 1) is for holding the pointers to three heads of soon-to-be linked lists that contain the impossible numbers of the corresponding slots (0^{th} element for slot 1, 1^{st} element for slot 2 and 3^{rd} element is for slot 3). Simlarly, `Possibles` array is to keep the possible digits of each of the slots.

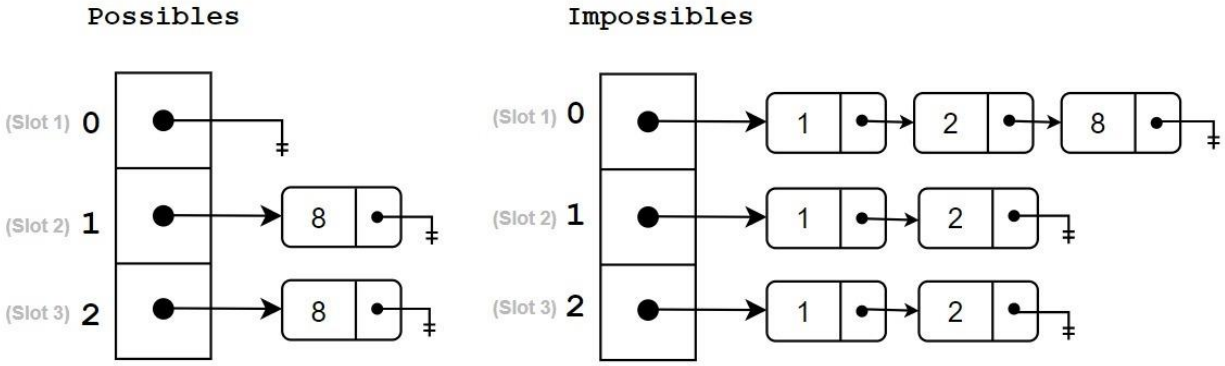

Figure 1 Visualization of the linked lists structure

At the beginning of the program, all six linked lists will be empty (their heads pointing `NULL`). You will insert and delete nodes, via dynamic memory allocation and de-allocation, while processing the feedbacks to the guesses. You have to keep the lists always sorted in ascending order.

You do not have to implement this structure as class(es). However, if you prefer to write one or more classes in the homework, you can do so provided that you perform a proper object oriented design.

Note: the sample structure given in Figure 1 shows a certain point in time while the program is running; i.e., it is not a static structure, but rather it is a snapshot. However, the backbone of the

structure is fixed. By backbone, we refer to the two arrays of pointers that hold the pointers to the heads of six linked lists.

Here, the term slot 1 is used for 100s digit, slot 2 is for 10s digit and slot 3 is for 1s digit. For example, If the number is 345, then slot 1 is 3, slot 2 is 4 and slot 3 is 5.

Note (regarding the allowed data structure in this homework): As shown in the given structure, you are only allowed to use built-in arrays in the backbone structure. Other than that, you cannot use any other types of data structures other than linked lists. You must use linked list to store the possible and impossible digits and manipulate them throughout the program as will be mentioned below. If you do not follow these rules (i.e. if you use another data structure and another guessing approach), depending on how you ease the homework, your grade will be lowered.

Input, Secret Number and Guesses

The program first reads an integer for the *game ID*. The `get_the_feedback` function automatically determines the secret number according to this game ID. You can assume that the game ID is always entered as a valid integer. You can also assume that the secret number is a 3-digit number and all of its digits are different than each other. The 100s slot, which is the most significant one (slot 1), can be 0. In other words, the range is between and including 012 and 987, but all digits are different than each other.

Then, the program will ask player guesses to be entered through the keyboard. Your program should perform an input check for the guesses entered. Each guess must have 3 distinct integer digits. In other words, it should have the same structure as the secret number, but you have to check it in your program. If the guess fails the input check, then your program should keep asking a new one until input check passes. Please see the sample run for prompts and error messages. Hint: although the discussion here implies that the guess is an integer, it is better to read and process it as `string`.

Program Flow and Output

Each time the user enters a guess, the program should firstly make the input checks (rules are given in the section above). Then, your program should get the feedback using the oracle (`get_the_feedback` function), update the linked list structure according to certain rules that are specified below. After that, the program will print the status of the linked lists on the screen in ascending sorted manner. This process will repeat until the contents of the linked lists dictate a single solution (details are explained later).

Feedback and Rules for Updating the Linked Lists

The oracle `get_the_feedback` function's prototype is below.

```
string get_the_feedback(string guess, int game_id);
```

It gets the `guess` and `game_id` as parameters and returns a 3-characters length string. In the returned string, the character with index 0 is the feedback for slot 1, the character with index 1 is

the feedback for slot 2, and the character with index 2 is the feedback for slot 3. Each character can either be 'R' for red, 'Y' for yellow, or 'G' for green.

- **R**: the corresponding digit of the guess does not exist anywhere in the secret number.
- **Y**: the corresponding digit of the guess exists in the secret number, but not in that slot.
- **G**: the corresponding digit of the guess is the correct digit in the corresponding slot of the secret number.

Rules for Updating the Linked Lists Based on the Feedback

For every feedback character value, the linked lists should be updated according to the following rules:

Suppose the digit value at slot t, where $t = 0, 1$ or 2, of guess is n, where $n = 0, 1, 2, \ldots, 9$. Depending on the feedback received for that slot, the linked lists will be updated accordingly:

- **R**: n should be inserted to all of the impossible lists of all of the slots as new nodes. The possible lists will not change.
- **Y**: n should be:
    - removed from the possible linked list of slot t, if exists.
    - inserted to the impossible linked list of slot t.
    - inserted to the possible linked lists of the other two slots if it wasn't in the corresponding slot's impossible linked list.
- **G**: n should be:
    - inserted to the possible linked list of slot t.
    - inserted to the impossible linked lists of the other two slots (other than t).
    - removed from the possible linked lists of the other two slots (other than t), if any.
    - all digits except n (i.e. from zero to nine except n) should be inserted to the impossible list of slot t. Also, all other digits (from zero to nine except n) should be removed, if any, from the possible list of slot t.

While you are inserting a digit to a linked list, you have to create a new node for it. Moreover, the insertions must not yield duplications in the corresponding list (i.e. if the digit to be inserted already exists in the list, you should not insert it at all). Similarly, while you are removing a digit, you have to `delete` the node containing that digit, if it exists. If the digit to be deleted does not exist, the list remains as is.

After insertions and deletions, the linked lists must be kept in ascending sorted order (all the time).

Stopping criteria

The game will end when the contents of the linked lists dictate a unique solution after processing a guess. This condition is **having nine distinct digits in the impossible lists of all three slots**. Once you get to this point, you have to display a message containing the secret number found. Please see the sample runs for message structure. Hint: you can reconstruct the secret number via the missing digits of the impossible lists of all slots.

Here we have three remarks:

- Depending on the sequence of guesses, the correct digits may or may not end up in the possible lists at the end of the game.
- It is not a must to get `GGG` feedback to end the game; sometimes, feedbacks to other guesses yield the unique correct solution (see sample runs). Thus please do not use the `GGG` feedback to test the end of the game.
- Before ending the program, do not forget to deallocate all the dynamically allocated memory.

The `get_the_feedback` function and its files

In the homework pack, we give out `feedback.h` and `feedback.cpp` files that contain the prototype and implementation of the oracle function named `get_the_feedback`. Please consider the following facts about them:

- Please do not change anything in these files.
- In C-Lion, you can use these files by adding to your project (by editing `CMakeLists.txt` file) and including the header file (`feedback.h` without changing its name) in your main cpp file.
- You will **not** copy the content of `feedback.cpp` in your main cpp.
- Moreover, you will not upload `feedback.h` and `feedback.cpp` files to CodeRunner; we have already put them there.

Some Important Rules

In order to get a full credit, your programs must be efficient and well presented, presence of any redundant computation or bad indentation, or missing, irrelevant comments are going to decrease your grades. You also have to use understandable identifier names, informative introduction and prompts. Modularity is also important; you have to use functions wherever needed and appropriate.

Since you will use dynamic memory allocation in this homework, it is very crucial to properly manage the allocated area and return the deleted parts to the heap whenever appropriate. Inefficient use of memory may reduce your grade.

When we grade your homework, we pay attention to these issues. Moreover, in order to observe the real performance of your codes, we may run your programs in *Release* mode and **we may test your programs with very large test cases**. Of course, your program should work in *Debug* mode as well.

Please do not use any non-ASCII characters (Turkish or other) in your code (not even as comments). And also do not use non-ASCII characters in your file and folder names. We really mean it; otherwise, you may encounter some errors.

You are not allowed to use codes found somewhere online or other than course material. You can use only the material that are covered in lectures (CS201 and CS204 so far). However, if you use a non-standard C++ library/function/class (such as strutils) covered in the courses, either (i) you have to submit your main code together with these extra source and header files so that CodeRunner runs your code properly, or (ii) copy necessary functions from them into your main

cpp by specifying where you copied them. In some assignments, we allow option (i) above in CodeRunner settings, but in some others we may not allow, so you can follow option (ii).

You are allowed to use sample codes shared with the class by the instructor and TAs. However, you cannot start with an existing .cpp or .h file directly and update it; you have start with an empty file. Only the necessary parts of the shared code files can be used and these parts must be clearly marked in your homework by putting comments like the following. Even if you take a piece of code and update it slightly, you have to put a similar marking (by adding "`and updated`" to the comments below.

```
/* Begin: code taken from ptrfunc.cpp */
…
/* End: code taken from ptrfunc.cpp */
```

Submission Rules (PLEASE READ, IMPORTANT)

It'd be a good idea to write your name and last name in the program (as a comment line of course). Do not use any Turkish characters anywhere in your code (not even in comment parts). For example, If your full name is "Satılmış Özbugsızkodyazaroğlu", then you must type it as follows:

// Satilmis Ozbugsizkodyazaroglu

We use CodeRunner in SUCourse for submission. No other way of submission is possible. Since the functionality part of the grading process will be automatic, you have to strictly follow these guidelines; otherwise, we cannot grade your homework.

The advantage CodeRunner it is that you will be able to test your code against sample test cases. However, the output should be exact, but the textual differences between the correct output and yours can be highlighted (by pressing "show differences" button) on the submission interface.

You should copy the full content of the main .cpp file and paste it into the specified “Answer” area in the relevant assignment submission page on SUCourse. Then you can test your code via CodeRunner against the sample runs (by pressing the "Check" button). You can check as much as you can; there is no penalty for multiple checks. Since you will not upload a file, your local cpp file name is not important.

You will **not** upload `feedback.h` and `feedback.cpp` files to CodeRunner; we have already put them there.

Even any tiny change in the output format will result in your grade being zero (0) for that particular test case, so please test your programs yourself, and against the sample runs that are available at the relevant assignment submission page on SUCourse (CodeRunner).

In the CodeRunner, there are some visible test cases. However, none of them will be used in grading. The grading test cases will be different and it is always a possibility that your program

works with these given test cases, but does not work with one or more grading test cases. Thus you have to test your program thoroughly.

You have to manually "Submit" after you test and finish with your code (there is no automatic submission). There is no re-submission. That means, after you submit, you cannot take it back. On the other hand, this does not mean that you have one shot to test CodeRunner. You don't have to complete your task in one time, you can continue from where you left last time, but you should not press submit before finalizing it. Therefore, you should make sure that it's your final solution version before you submit it. Also, we still do not suggest that you develop your solution on CodeRunner but rather on your IDE on your computer.

Last, even if you cannot completely finish your homework, you can still submit.

Sample Runs

Sample runs are given below, but these are not comprehensive, therefore you have to consider **all possible cases** to get full mark. Inputs are shown in **bold**.

We will configure CodeRunner to test these sample runs for you. However, grading test cases will be totally different.

Sample Run 1

```
Please enter a game ID.
15
Enter your guess.
942
Linked lists:
Slot: 1
Impossibles: 0 1 2 3 4 5 6 7 8
Possibles: 9

Slot: 2
Impossibles: 4 9
Possibles: 2

Slot: 3
Impossibles: 2 4 9
Possibles:

Enter your guess.
920
Linked lists:
Slot: 1
Impossibles: 0 1 2 3 4 5 6 7 8
Possibles: 9
```

```
Slot: 2
Impossibles: 0 1 3 4 5 6 7 8 9
Possibles: 2

Slot: 3
Impossibles: 0 2 4 9
Possibles:

Enter your guess.
921
Linked lists:
Slot: 1
Impossibles: 0 1 2 3 4 5 6 7 8
Possibles: 9

Slot: 2
Impossibles: 0 1 3 4 5 6 7 8 9
Possibles: 2

Slot: 3
Impossibles: 0 1 2 4 9
Possibles:

Enter your guess.
928
Linked lists:
Slot: 1
Impossibles: 0 1 2 3 4 5 6 7 8
Possibles: 9

Slot: 2
Impossibles: 0 1 3 4 5 6 7 8 9
Possibles: 2

Slot: 3
Impossibles: 0 1 2 4 8 9
Possibles:

Enter your guess.
925
Linked lists:
Slot: 1
Impossibles: 0 1 2 3 4 5 6 7 8
Possibles: 9

Slot: 2
Impossibles: 0 1 3 4 5 6 7 8 9
```

```
Possibles: 2

Slot: 3
Impossibles: 0 1 2 4 5 8 9
Possibles:

Enter your guess.
923
Linked lists:
Slot: 1
Impossibles: 0 1 2 3 4 5 6 7 8
Possibles: 9

Slot: 2
Impossibles: 0 1 3 4 5 6 7 8 9
Possibles: 2

Slot: 3
Impossibles: 0 1 2 3 4 5 8 9
Possibles:

Enter your guess.
927
Linked lists:
Slot: 1
Impossibles: 0 1 2 3 4 5 6 7 8
Possibles: 9

Slot: 2
Impossibles: 0 1 3 4 5 6 7 8 9
Possibles: 2

Slot: 3
Impossibles: 0 1 2 3 4 5 7 8 9
Possibles:

The secret number is: 926
Congrats! Now, deleting the lists...
```

Sample Run 2

```
Please enter a game ID.
184
Enter your guess.
djh
Invalid guess. Enter your guess.
```

87
Invalid guess. Enter your guess.
12
Invalid guess. Enter your guess.
012
Linked lists:
Slot: 1
Impossibles: 1 2 3 4 5 6 7 8 9
Possibles: 0

Slot: 2
Impossibles: 0 1 2
Possibles:

Slot: 3
Impossibles: 0 1 2
Possibles:

Enter your guess.
012
Linked lists:
Slot: 1
Impossibles: 1 2 3 4 5 6 7 8 9
Possibles: 0

Slot: 2
Impossibles: 0 1 2
Possibles:

Slot: 3
Impossibles: 0 1 2
Possibles:

Enter your guess.
13
Invalid guess. Enter your guess.
013
Linked lists:
Slot: 1
Impossibles: 1 2 3 4 5 6 7 8 9
Possibles: 0

Slot: 2
Impossibles: 0 1 2 3
Possibles:

Slot: 3

```
Impossibles: 0 1 2 3
Possibles:

Enter your guess.
04k
Invalid guess. Enter your guess.
y$3x2
Invalid guess. Enter your guess.
045
Linked lists:
Slot: 1
Impossibles: 1 2 3 4 5 6 7 8 9
Possibles: 0

Slot: 2
Impossibles: 0 1 2 3 4 5
Possibles:

Slot: 3
Impossibles: 0 1 2 3 4 5
Possibles:

Enter your guess.
098
Linked lists:
Slot: 1
Impossibles: 1 2 3 4 5 6 7 8 9
Possibles: 0

Slot: 2
Impossibles: 0 1 2 3 4 5 8 9
Possibles:

Slot: 3
Impossibles: 0 1 2 3 4 5 8
Possibles: 9

Enter your guess.
079
Linked lists:
Slot: 1
Impossibles: 1 2 3 4 5 6 7 8 9
Possibles: 0

Slot: 2
Impossibles: 0 1 2 3 4 5 6 8 9
Possibles: 7
```

```
Slot: 3
Impossibles: 0 1 2 3 4 5 6 7 8
Possibles: 9

The secret number is: 079
Congrats! Now, deleting the lists...
```

Sample Run 3

```
Please enter a game ID.
1
Enter your guess.
256
Linked lists:
Slot: 1
Impossibles: 2 5 6
Possibles:

Slot: 2
Impossibles: 2 5 6
Possibles:

Slot: 3
Impossibles: 2 5 6
Possibles:

Enter your guess.
938
Linked lists:
Slot: 1
Impossibles: 2 5 6 8 9
Possibles: 3

Slot: 2
Impossibles: 2 3 5 6 8
Possibles: 9

Slot: 3
Impossibles: 2 5 6 8
Possibles: 3 9

Enter your guess.
000
Invalid guess. Enter your guess.
002
```

```
Invalid guess. Enter your guess.
093
Linked lists:
Slot: 1
Impossibles: 0 2 5 6 8 9
Possibles: 3

Slot: 2
Impossibles: 0 2 3 5 6 8 9
Possibles:

Slot: 3
Impossibles: 0 2 3 5 6 8
Possibles: 9

Enter your guess.
317
Linked lists:
Slot: 1
Impossibles: 0 1 2 4 5 6 7 8 9
Possibles: 3

Slot: 2
Impossibles: 0 1 2 3 5 6 7 8 9
Possibles:

Slot: 3
Impossibles: 0 1 2 3 5 6 7 8
Possibles: 9

Enter your guess.
349
Linked lists:
Slot: 1
Impossibles: 0 1 2 4 5 6 7 8 9
Possibles: 3

Slot: 2
Impossibles: 0 1 2 3 5 6 7 8 9
Possibles: 4

Slot: 3
Impossibles: 0 1 2 3 4 5 6 7 8
Possibles: 9

The secret number is: 349
Congrats! Now, deleting the lists...
```

Sample Run 4

```
Please enter a game ID.
-5
Enter your guess.
258
Linked lists:
Slot: 1
Impossibles: 2 5 8
Possibles:

Slot: 2
Impossibles: 2 5 8
Possibles:

Slot: 3
Impossibles: 2 5 8
Possibles:

Enter your guess.
147
Linked lists:
Slot: 1
Impossibles: 1 2 4 5 8
Possibles: 7

Slot: 2
Impossibles: 2 4 5 8
Possibles: 1 7

Slot: 3
Impossibles: 2 4 5 7 8
Possibles: 1

Enter your guess.
713
Linked lists:
Slot: 1
Impossibles: 1 2 3 4 5 7 8
Possibles:

Slot: 2
Impossibles: 1 2 3 4 5 8
Possibles: 7
```

```
Slot: 3
Impossibles: 2 3 4 5 7 8
Possibles: 1

Enter your guess.
071
Linked lists:
Slot: 1
Impossibles: 0 1 2 3 4 5 7 8
Possibles:

Slot: 2
Impossibles: 0 1 2 3 4 5 6 8 9
Possibles: 7

Slot: 3
Impossibles: 0 2 3 4 5 6 7 8 9
Possibles: 1

Enter your guess.
6
Invalid guess. Enter your guess.
671
Linked lists:
Slot: 1
Impossibles: 0 1 2 3 4 5 7 8 9
Possibles: 6

Slot: 2
Impossibles: 0 1 2 3 4 5 6 8 9
Possibles: 7

Slot: 3
Impossibles: 0 2 3 4 5 6 7 8 9
Possibles: 1

The secret number is: 671
Congrats! Now, deleting the lists...
```

Sample Run 5

```
Please enter a game ID.
100
Enter your guess.
168
Linked lists:
```

```
Slot: 1
Impossibles: 1 6 8
Possibles:

Slot: 2
Impossibles: 1 6 8
Possibles:

Slot: 3
Impossibles: 1 6 8
Possibles:

Enter your guess.
976
Linked lists:
Slot: 1
Impossibles: 0 1 2 3 4 5 6 7 8
Possibles: 9

Slot: 2
Impossibles: 1 6 7 8 9
Possibles:

Slot: 3
Impossibles: 1 6 7 8 9
Possibles:

Enter your guess.
9
Invalid guess. Enter your guess.
924
Linked lists:
Slot: 1
Impossibles: 0 1 2 3 4 5 6 7 8
Possibles: 9

Slot: 2
Impossibles: 1 2 6 7 8 9
Possibles: 4

Slot: 3
Impossibles: 1 4 6 7 8 9
Possibles: 2

Enter your guess.
94
Invalid guess. Enter your guess.
```

943
Linked lists:
Slot: 1
Impossibles: 0 1 2 3 4 5 6 7 8
Possibles: 9

Slot: 2
Impossibles: 0 1 2 3 5 6 7 8 9
Possibles: 4

Slot: 3
Impossibles: 1 3 4 6 7 8 9
Possibles: 2

Enter your guess.
945
Linked lists:
Slot: 1
Impossibles: 0 1 2 3 4 5 6 7 8
Possibles: 9

Slot: 2
Impossibles: 0 1 2 3 5 6 7 8 9
Possibles: 4

Slot: 3
Impossibles: 1 3 4 5 6 7 8 9
Possibles: 2

Enter your guess.
940
Linked lists:
Slot: 1
Impossibles: 0 1 2 3 4 5 6 7 8
Possibles: 9

Slot: 2
Impossibles: 0 1 2 3 5 6 7 8 9
Possibles: 4

Slot: 3
Impossibles: 0 1 3 4 5 6 7 8 9
Possibles: 2

The secret number is: 942
Congrats! Now, deleting the lists...

Sample Run 6

```
Please enter a game ID.
20
Enter your guess.
186
Linked lists:
Slot: 1
Impossibles: 1 6 8
Possibles:

Slot: 2
Impossibles: 1 6 8
Possibles:

Slot: 3
Impossibles: 1 6 8
Possibles:

Enter your guess.
953
Linked lists:
Slot: 1
Impossibles: 1 3 6 8 9
Possibles: 5

Slot: 2
Impossibles: 1 3 5 6 8 9
Possibles:

Slot: 3
Impossibles: 1 3 6 8 9
Possibles: 5

Enter your guess.
852
Linked lists:
Slot: 1
Impossibles: 1 3 6 8 9
Possibles: 2 5

Slot: 2
Impossibles: 1 3 5 6 8 9
Possibles: 2
```

```
Slot: 3
Impossibles: 1 2 3 6 8 9
Possibles: 5

Enter your guess.
254
Linked lists:
Slot: 1
Impossibles: 1 2 3 4 6 8 9
Possibles: 5

Slot: 2
Impossibles: 1 3 4 5 6 8 9
Possibles: 2

Slot: 3
Impossibles: 1 2 3 4 6 8 9
Possibles: 5

Enter your guess.
507
Linked lists:
Slot: 1
Impossibles: 0 1 2 3 4 5 6 8 9
Possibles: 7

Slot: 2
Impossibles: 0 1 3 4 5 6 8 9
Possibles: 2 7

Slot: 3
Impossibles: 0 1 2 3 4 6 7 8 9
Possibles: 5

Enter your guess.
725
Linked lists:
Slot: 1
Impossibles: 0 1 2 3 4 5 6 8 9
Possibles: 7

Slot: 2
Impossibles: 0 1 3 4 5 6 7 8 9
Possibles: 2

Slot: 3
Impossibles: 0 1 2 3 4 6 7 8 9
```

```
Possibles: 5

The secret number is: 725
Congrats! Now, deleting the lists...
```

Good Luck
Albert Levi and Ahmed Salem