# Project 1: Bulls and Cows

**Background:**

*Bulls and Cows* is an old code-breaking paper and pencil game for two players, predating the similar commercially-marketed board game Master Mind.

It is a game with numbers that may date back a century or more. It is played by two opponents. One player becomes the *code maker*, the other the *code breaker*. The code maker has a 4-digit secret number (the digits must be all different). Then, the code breaker tries to guess the secret number. For every guess, the code maker gives a hint: if the matching digits are on their right positions, they are "bulls", if on different positions, they are "cows". For example:

- Secret number: 4271
- code breaker's try: 1234
- Hint: 1 bull and 2 cows. (The bull is "2", the cows are "4" and "1".)

The code breaker wins the game by providing the correct guess (possibly through many iterations according to the hints).

**Programming assignment:**

Implement Bulls and Cows in MIPS assembly language on MARS

- A secret code is set at the first round by the user (assuming to be the code maker to initialize the secret code).
- Then the program enters the game mode by prompting the user (now assuming to be the code breaker) to enter the guesses and giving out hints until the correct guess is made.

**Example I/O: (user's input in bold font)**

```
Input the secret code:
                                3298

Beginning Bulls and Cows…

Input your guess (round 1):
                                1235
hint: 1 bull and 1 cow
Input your guess (round 2):
                                1278
hint: 2 bulls and 0 cow
Input your guess (round 3):
                                3298

You win in 3 rounds!
```

**Extra Credit:**

- Error tolerance: check for illegal input.
- Surrender allowance: "q" to exit and prompt "you lose by giving up in x rounds! The secret code is xxxx".

**What to turn in:**

The following two files zipped.

- **Lab report (pdf), consisting of the following parts:**

  a) Program features:
     i. What have you implemented – basic function, extra credit, anything else?
     ii. Is your program running correctly? If not, describe the malfunction in details.
     iii. Have you run into any significant problems / bugs in this project? Describe how they are solved.

  b) Provide 5 screenshots on the running I/O of your program to show how your program behave and illustrate your features. Be creative and thorough in these tests so as to be convincing and to cover various corner cases.

  c) Technical answers:
     i. How does your program take the user's input and translate it into the 4 numbers? Do you detect illegal input? If so, what kind of illegal input and how do you detect?
     ii. How does your program calculate the hint (i.e., the number of bulls and cows) for each guess trial of the code breaker?
     iii. If you are a code breaker in this game, what's your best strategy of breaking the code in as few rounds of trials as possible?

- **Assembly code (.asm) well commented.**

You will be graded on the following criteria: 1) program functionality, 2) thoroughness and insights in answering the questions in the lab report, and 3) code quality.

# Project 1: Bulls and Cows

**Background:**

*Bulls and Cows* is an old code-breaking paper and pencil game for two players, predating the similar commercially-marketed board game Master Mind.

It is a game with numbers that may date back a century or more. It is played by two opponents. One player becomes the *code maker*, the other the *code breaker*. The code maker has a 4-digit secret number (the digits must be all different). Then, the code breaker tries to guess the secret number. For every guess, the code maker gives a hint: if the matching digits are on their right positions, they are "bulls", if on different positions, they are "cows". For example:

- Secret number: 4271
- code breaker's try: 1234
- Hint: 1 bull and 2 cows. (The bull is "2", the cows are "4" and "1".)

The code breaker wins the game by providing the correct guess (possibly through many iterations according to the hints).

**Programming assignment:**

Implement Bulls and Cows in MIPS assembly language on MARS

- A secret code is set at the first round by the user (assuming to be the code maker to initialize the secret code).
- Then the program enters the game mode by prompting the user (now assuming to be the code breaker) to enter the guesses and giving out hints until the correct guess is made.

**Example I/O: (user's input in bold font)**

```
Input the secret code:
                                3298

Beginning Bulls and Cows…

Input your guess (round 1):
                                1235
hint: 1 bull and 1 cow
Input your guess (round 2):
                                1278
hint: 2 bulls and 0 cow
Input your guess (round 3):
                                3298

You win in 3 rounds!
```

**Extra Credit:**

- Error tolerance: check for illegal input.
- Surrender allowance: "q" to exit and prompt "you lose by giving up in x rounds! The secret code is xxxx".

**What to turn in:**

The following two files zipped.

- **Lab report (pdf), consisting of the following parts:**

  a) Program features:
     i. What have you implemented – basic function, extra credit, anything else?
     ii. Is your program running correctly? If not, describe the malfunction in details.
     iii. Have you run into any significant problems / bugs in this project? Describe how they are solved.

  b) Provide 5 screenshots on the running I/O of your program to show how your program behave and illustrate your features. Be creative and thorough in these tests so as to be convincing and to cover various corner cases.

  c) Technical answers:
     i. How does your program take the user's input and translate it into the 4 numbers? Do you detect illegal input? If so, what kind of illegal input and how do you detect?
     ii. How does your program calculate the hint (i.e., the number of bulls and cows) for each guess trial of the code breaker?
     iii. If you are a code breaker in this game, what's your best strategy of breaking the code in as few rounds of trials as possible?

- **Assembly code (.asm) well commented.**

You will be graded on the following criteria: 1) program functionality, 2) thoroughness and insights in answering the questions in the lab report, and 3) code quality.

Kai Zhao 670720413
ECE 267 Spring 2012
2012 February 25

**Project 1: Bulls and Cows**

a) Program features:

i) What have you implemented – basic function, extra credit, anything else?

Answer: The features implemented in this program are

- Guess input were mirrored for the guesser to check what was inputted
- Singular and plural forms of cow(s), bull(s), and round(s) were accounted
- The guesser may enter 'q' to quit
- Error tolerance for repeated digits for secret code input
- Error tolerance for non-number secret code
- Error tolerance for non 4 digit entry
- Error tolerance for non-number guesses except 'q'
- The program will show the correct secret code at the end
- Round number is not increased if the guess is invalid

ii) Is your program running correctly? If not, describe the malfunction in details.

Answer: Yes, my program is running correctly.

iii) Have you run into any significant problems / bugs in this project? Describe how they are solved.

Answer: Yes, I ran into a lot of significant problems in this project.

- I had trouble trying to understand how to read in string. I was not aware that it read x-1 amount of characters specified at $a0 in the memory address specified at $a0. Running the program one step at a time allowed me to figure out how to read in a string.

TL; DR: How to use string input.

- I was unable to find a short cut in doing the extra credit of this project because when the project reads in an integer, a lot of manipulation had to be done to extract the 4 digits. However, when reading in the string, those manipulations were quite useless. Attempting the extra credit basically means having to rewrite the entire program. This was solved by spending more time on the program to rewrite it.

  TL; DR: Switching from word to ascii.

- I had trouble trying to display the correct secret code when the user quitted. Whenever I wanted to load the address of the secret code, it seems to have loaded in 2 words instead of 1. At the end, I just emptied out the word guesses so that the 2$^{nd}$ word is empty so that the secret code can be displayed. This was solved by debugging and running the program one step at a time while keeping track of the memory.

  TL; DR: Confused about memory location of strings.

- I had trouble trying counting the number of cows and bulls. Initially, I iterated through all possible digits and add a cow if the digits appeared in both the secret code and the guess, and then iterated through all the possible digits again and if the digit location matches, then I would subtract 1 cow and add 1 bull. This was very bad code quality as it took 200 lines of code to count the number of cows and bulls alone. I fixed this by thinking about a different approach.

  TL; DR: Bad code quality.

- Sometimes when I ran the program with the same code, I ended up with different results, which was insanity. I was not sure what caused that problem, but I was suspecting that my computer's lag cause the program to assemble incorrectly or I did not properly reset

MIPS memory and registers before each trail. I attempted to fix this program by repeatedly hitting the reset button before each trail and running each trail twice.

TL; DR: Inconsistent program output with the same program codes.

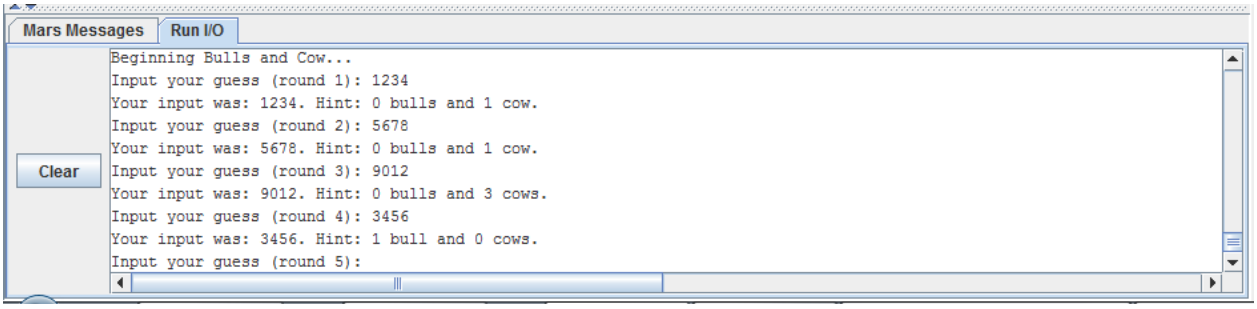
- I had trouble trying to figure out a formula to solve for the secret code in as few tries as possible. After a lot of attempts, I think that the secret code can be guessed in 11 tries, using the first 10 tries to guess every digit in every position with no more than any 2 of the same digit appear in the same guess more than twice. However, I am not able to confirm that this method will work for all 5040 possible combinations of the secret code. I had trouble figuring out a formula to solve for the secret code. I attempted to solve this problem by providing an analogy and a few examples of how to go about solving for the secret code on the 11$^{th}$ try.

  TL; DR: Putting my strategy in words.

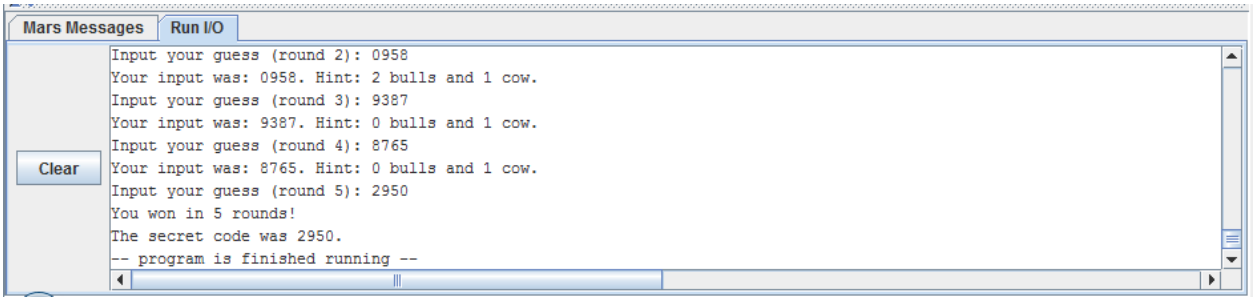
b) Provide 5 screenshots on the running I/O of your program to show how your program behave and illustrate your features. Be creative and thorough in these tests so as to be convincing and to cover various corner cases.

Answer:

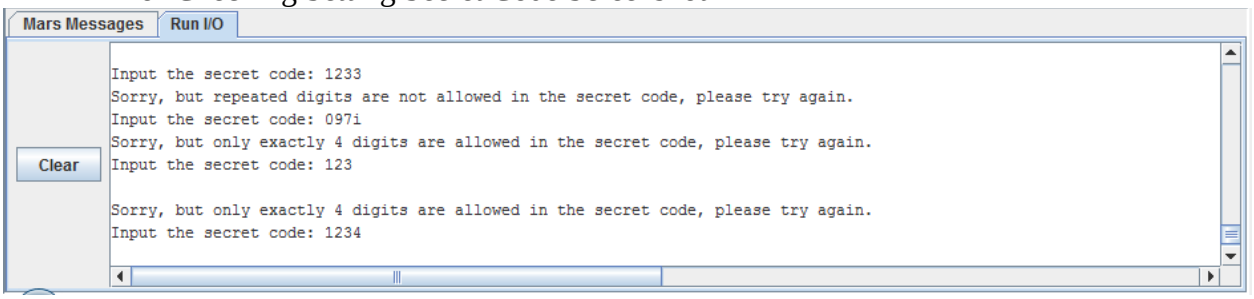
- Plain Screenshot



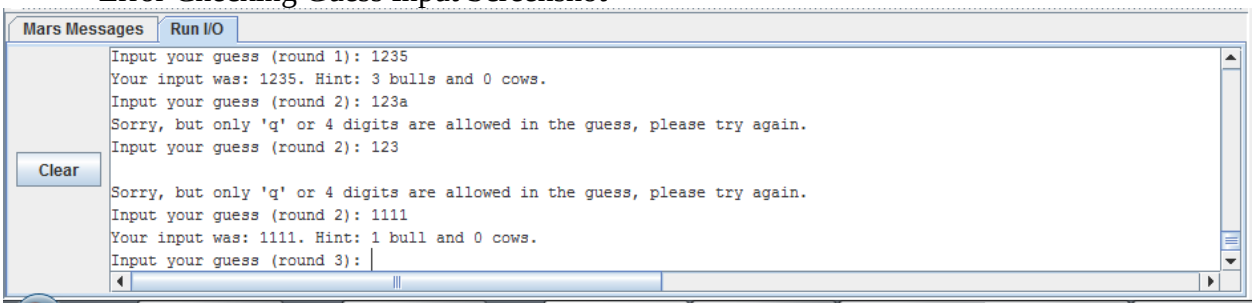
- Win Game Screenshot



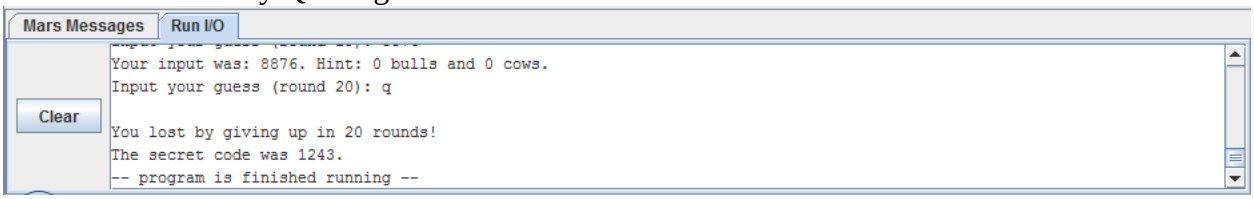

- Error Checking Setting Secret Code Screenshot




- Error Checking Guess Input Screenshot




- End Game by Quitting Screenshot




Question: Besides getting a new monitor, is there any way to increase the Run I/O past 10 lines so that the 5 screenshots are more meaningful?

c) Technical answers:

i) How does your program take the user's input and translate it into the 4 numbers? Do you detect illegal input? If so, what kind of illegal input and how do you detect?

Answer: My program read in a string by placing a memory address in $a0, number of characters minus one in $a1, and 8 in $v0. My program translates it into 4 numbers by loading the 4 bytes individual into registers from the memory address I loaded into $a0. Yes, my program detects for illegal input. My program detects that the secret code input was of length 4, there was no repeats, and there was no non-numbers by comparing ascii values. Each of the 4 digits should be between '0' and '9' in terms of the ascii value, and none of them should be each to another. My program also detects that the guess input was of length 4 and there was no non-numbers except for 'q' if the user wants to quit also by comparing ascii values in a similar fashion.

TL; DR: Loaded bytes, program detects for length, repeat, and char invalid inputs.

ii) How does your program calculate the hint (i.e., the number of bulls and cows) for each guess trial of the code breaker?

Answer: My program first checks the first digit for a bull. If there is a bull in the first position, then my program increases the bull by 1 and move onto the next digit. If none exist, then it checks for a cow that matches with the first digit of the secret code. If at least one exists, the code will branch so that it will only increase the cow counter by 1. The same is repeated for the other 3 digits. Therefore, the sum of cows and bulls are always less than or equal or 4.

TL; DR: Program compared the bytes of code to the bytes of guess.

iii) If you are a code breaker in this game, what's your best strategy of breaking the code in as few rounds of trials as possible?

Answer: A good strategy is to use the first at most 10 tries to place every digit in every position with any 2 of the same digit occurring more than twice together. This will allow you to check your guess with the results from the first 10 guesses. Also, the numbers should be evenly divided to increase entropy so that a probability mass function is more meaningful in helping to find the

4 digits. Also, if any 2 of the same digit does not occurring more than twice together, it helps differentiate between using one over the other. After one have the results of the hint from the first 10 guesses, one can guess until his or her heart's content until the guess matches with the hints. It's more or less like solving a Sudoku puzzle, but I cannot guarantee one solution due to the 5040 different combinations of the secret code. The list of numbers to guess that I have devised is 1234, 5678, 9012, 3456, 7890, 6183, 4961, 2749, 0527, and 8305. This list of number which satisfies my first 10 tries specifications is not unique because the numbers can be shifted by 1, 2, or 3, or the numbers can be interchanged. I guess the best way to describe this method is with a few examples.

TL; DR: Exhaust every digit in every position.

**Example 1**

Secret Code: 4850

| Guess | Bull | Cow | P(x)=c*sum(seq(1/(bulls + cows)) |
|---|---|---|---|
| 1234 | 0 | 1 | P(1)=c*(0.25+0.25+0.25+0.25)=1.00*c |
| 5678 | 0 | 2 | P(2)=c*(0.25+0.25+0.25+0.50)=1.25*c |
| 9012 | 0 | 1 | P(3)=c*(0.25+0.50+0.25+0.75)=1.75*c |
| 3456 | 1 | 1 | P(4)=c*(0.25+0.50+0.25+0.25)=1.25*c |
| 7890 | 2 | 0 | P(5)=c*(0.50+0.50+0.50+0.75)=2.25*c |
| 6183 | 0 | 1 | P(6)=c*(0.50+0.50+0.25+0.25)=1.50*c |
| 4961 | 1 | 0 | P(7)=c*(0.50+0.50+0.25+0.50)=1.75*c |
| 2749 | 0 | 1 | P(8)=c*(0.50+0.50+0.25+0.75)=2.00*c |
| 0527 | 0 | 2 | P(9)=c*(0.25+0.50+0.25+0.25)=1.25*c |
| 8305 | 0 | 3 | P(0)=c*(0.25+0.50+0.50+0.75)=2.00*c |

As you can see, to find the P(x), I just summed their probability of being a bull or a cow every time they occurred. For example, in the guess 1234, each digit 1, 2, 3, and 4 has a 25% chance of being a bull or a cow because there is only 1 cow in those 4 digits.

Collecting all the terms with the highest probability of being in the secret code, I have 35780, with 5 being the highest probability. I would probability try to eliminate 1 to check with 4 digits.

If I eliminate 3 (the digit with the lowest probability of the 5) to guess 5780, then the hint from guess 5678 will not be satisfied. If I eliminate 7(another digit with the lowest probability of occurring of the 5) to guess 3580, then the hint from guess 6183 will not be satisfied. To attempt to satisfy the hint from guess 6183, I suspect that I need to eliminate 3 or 8 (since I need to eliminate 1 cow because the hint from 6183 tells me that there is only 1 cow while I current have 2 cows when comparing 3580 to 6183). If I eliminate 3, then I would need to include 4 so that guess 1234 remains satisfied. If I eliminate 8, then I would need to include 7 to keep guess 5678 and guess 6183 satisfied. Since 7 have a higher probability of occurring than 4, I exclude 8 and include 7 to guess 3570. However, since 3570 does not satisfy the hint from guess 4961, I'm going to go backtrack to guess 3580 and replace 3 with a 4 to guess 4580. Guess 4580 seems to satisfy all 10 hints. Therefore, I now need to satisfy the bull position. Guess 4961 tells me that 4 is the leftmost digit. Guess 3456 tells me that 5 is the middle-right. Guess 5678 tells that 8 is NOT the 4$^{th}$ digit. Therefore, 8 must be the middle-left by process of elimination. Finally, 0 must be the rightmost digit by the process of elimination. Therefore, I would guess 4850 on the 11$^{th}$ guess, which is correct.

TL; DR: Demonstration of the process of guessing.

To summarize, you can first use probability and number of occurrences to attempt to find the 4 digits. After you have a guess, you check to see if it fits. If it fits, then try to figure out the order and guess it. Solving for the secret code on the 11$^{th}$ try is much like solving a Sudoku puzzle because in a Sudoku puzzle, you just guess digits and see if it fits in the Sudoku.

TL; DR: Keep changing the guess until all 10 hints are satisfied.

**Example 2**

Secret Code: Unknown (Entered without looking at keyboard)

| Guess | Bull | Cow | P(x)=c*sum(seq(1/(bulls + cows)) |
|---|---|---|---|
| 1234 | 0 | 1 | P(1)=c*(0.25+0.50+0.25+0.50)=1.50*c |
| 5678 | 0 | 2 | P(2)=c*(0.25+0.50+0.50+0.50)=1.75*c |
| 9012 | 0 | 2 | P(3)=c*(0.25+0.50+0.25+0.25)=1.25*c |
| 3456 | 2 | 0 | P(4)=c*(0.25+0.50+0.50+0.50)=1.75*c |
| 7890 | 0 | 1 | P(5)=c*(0.50+0.50+0.50+0.25)=1.75*c |
| 6183 | 0 | 1 | P(6)=c*(0.50+0.50+0.25+0.50)=1.75*c |
| 4961 | 1 | 1 | P(7)=c*(0.50+0.25+0.50+0.50)=1.75*c |
| 2749 | 1 | 1 | P(8)=c*(0.50+0.25+0.25+0.25)=1.25*c |
| 0527 | 0 | 2 | P(9)=c*(0.50+0.25+0.50+0.50)=1.75*c |
| 8305 | 0 | 1 | P(0)=c*(0.50+0.25+0.50+0.25)=1.50*c |

The collected digits are 245679, but that does not satisfy hint from guess 1234 nor is it 4 digits.

The hint from guess 8305 tells me that I am pretty confident that 5 is included.

The hint from 5678 and 6183 tells me that 6 is also included, but without a high confidence level. However, if I were to compute the conditional probability of 6 given that 5 is included, I suspect that a high confidence level that 6 will be included. For now, I'll just go with 6 being included and backtrack if I hit a dead end.

If I included both 5 and 6, then the hint from 5678 tells me that 7 and 8 is not included.

The hint from 7890 tells me that 9 is included.

To choose between 2 and 4, I used the hint from guess 4961 to see that 4 is not included, so 2 is included.

The remaining included digits are 2569, which does satisfy all the hints.

The hint from 3456 tells me that 5 is the middle-right digit and 6 is the rightmost.

The hints tell me that 2 is not the middle-left digit, 6 is not the middle-left digit, and 5 is not the middle-left digit. Therefore, 9 is the middle-left digit.

The hint from 2739 tells me that 2 is the leftmost, since 9 must be a cow if 9 is not middle-left digit.

My 11$^{th}$ guess on the program will be 2956, which is correct.

TL; DR: Another example of the process of guessing.

Note that solving for the secret code will be much easier if one of the first 10 tries comes up with 0 cows and 0 bulls because the process of elimination would play a huge role to simplify and solve for the secret code without even having to bother with the probability mass function. I will try to demonstrate this in the next example.

TL; DR: Some hints are more helpful than others.

**Example 3**

Secret Code: 0315 (Set on purpose to make hint from 2749 show 0 bulls and 0 cows)

| Guess | Bull | Cow |
|---|---|---|
| 1234 | 0 | 2 |
| 5678 | 0 | 1 |
| 9012 | 1 | 1 |
| 3456 | 0 | 2 |
| 7890 | 0 | 1 |
| 6183 | 0 | 2 |
| 4961 | 0 | 1 |
| 2749 | 0 | 0 |
| 0527 | 1 | 1 |
| 8305 | 2 | 1 |

Crossing out the digits that will not be include means to cancel out 2, 7, 4, and 9, since the hint says 0 cows and 0 bulls, we have:

| Guess | Bull | Cow |
|---|---|---|
| 1~~2~~3~~4~~ | 0 | 2 |
| 56~~7~~8 | 0 | 1 |
| ~~9~~01~~2~~ | 1 | 1 |
| 3456 | 0 | 2 |
| ~~7~~890 | 0 | 1 |
| 6183 | 0 | 2 |
| ~~49~~61 | 0 | 1 |
| ~~2749~~ | 0 | 0 |
| 05~~27~~ | 1 | 1 |
| 8305 | 2 | 1 |

The hint from ~~9~~012 tells us that <u>0 and 1 is included</u>.

The hint from 05~~27~~ tells us that <u>5 is included</u> as well.

The hint from 1~~2~~34 and 8305 tells us that <u>3 is included</u> as well.

The hints tell us that 1 is not the leftmost digit, 5 is not the leftmost digit, and 3 is not the leftmost digits. Therefore, 0 is the leftmost digit.

Then the hint from 9012 tells us that 1 is the middle-right digit.

Then the 8305 tell us that 3 is the middle-left digit and 5 is the rightmost digit.

Guess: 0315, which is correct.

TL; DR: An example of using a hint with 0 cows and 0 bulls to quickly solve for secret code.

In conclusion, the digits analyses take too much time to the point where it might not even be feasible to do the digit analysis if one is hoping to save time. However, this strategy should almost always beat having to guess each digit individually and then guessing that digit in each slot to find where it becomes a bull. This strategy can be combined with another computer program that iterates through all (10 choose 4) or 210 possible 4-digit combinations to first find the 4 digit. Then the computer program should be able to iterate through the (4!) or 24 possible combinations of the 4 digit combination to match the bulls and cows. Furthermore, not all hints are needed. Rarely do I need to check the hint from the tenth guess for the contradiction to revise my guess. If a hint shows 0 cows and 0 bulls, then it is better to use guesses that involves half cows or bulls and half neither cows nor bulls to quickly identify the cows. On the other hand, if the hint shows if the cows and bull sum up to 3 or 4, then it is better to use guesses that involve the cows to quickly fund their bull positions. Therefore, figuring out which guess to use first within the first 10 guesses may help to solve for the secret code before the eleventh guess. I would recommend starting with guess 1234 due to Benford's law, also known as the first-digit law, because although Benford's law does not model secret codes, the secret code may model things that do model Benford's law. Then guess up to the other 9 guesses based on conditional probability. Similar to Inception (which has a dream inside of a dream) and Sudoku puzzles

(which has 9 boxes inside of 9 boxes), attempt to solve for the guesses to solve for guesses so that not all 10 guesses are needed. One way to go about doing so is to guess the guesses with more digits that require more information about whether to include them. Also remember that not having a cow or a bull tells us just about as much information as do having a cow or bull due to the process or elimination. In conclusion, my best strategy of breaking the code in as few rounds as possible is to guess for the guesses to use to eliminate 5039 out of the 5040 possible combinations, which should take no more than 11 guesses.

TL; DR: Not all 10 hints may be needed to solve for secret code to further minimize the number of rounds use to break the secret code.