**ME 466 Introduction to AI Fall**
**2021**
**Programming Assignment 1**
**Submitted on: 24 October 2021**

**Name: KORKUT EMRE ARSLANTÜRK**

**Student ID: 250206039**

---

**Grade:**

I hereby declare that the paper I am submitting under this cover is product of my own efforts only. Even if I worked on some of the problems together with my classmates, I prepared this paper on my own, without looking at any other classmate's paper. I am knowledgeable about everything that is written under this cover, and I am prepared to explain any scientific/technical content written here if a short oral examination about this paper is conducted by the instructor. I am aware of the serious consequences of cheating.

**Signature:**

**TABLE OF CONTENTS**

# 1. INTRODUCTION

In this assigment, my aim is that finding a logical algorithm to solve a modified Mastermind Game which is called the Bulls and Cows game. In original Mastermind game, a player try to guess colored pegs which is determined by other player. However, Bulls and Cows game is played with digits and  the all digits must be different. One of the first Bulls and Cows computer implementations was MOO which is published in 1970 by J.M. Grochow at MIT for Multics opearing system[1].

In our modified version, Player A write a secret four-digit number(first digit can not be zero) and Player B tries to guess secret number. After each guess, Player A gives information to Player B how many of the guess's digits correspond to the secret number; how many digits are in place(which is called bulls) and how many digits are out of place(cows). Player B finds the secret number after making numerous guesses. Computer plays as a player B and try to guess user’s(Player A’s) secret number as possible as the fewest guest.

My program includes 4 different functions and main part. One of the function check number has repeated digits or not, another function calculates number of cows and rows, third one provides that turn an integer into an array which includes all digits in the number and fourth function using for make a new guess. In my program, computer find the secret number in between 15 and 25 tries.

My prediction algorithm is based on creating a set of possible correct values and removing the wrong guess and other variables which have less bulls and cows than the number of prediction after each prediction. It is aimed to reduce the set and increase the probability of prediction in the new forecast.

## 2. BODY OF THE REPORT

First of all, I imported required libraries which were used in next steps in the code.


Figure 1: Importing libraries

I used random library while producing new guess in each try. It gave me a random value in the set of possible correct values. Also, I imported numpy library because I use np.array type while determining cows number with comparing two four-digit numbers.

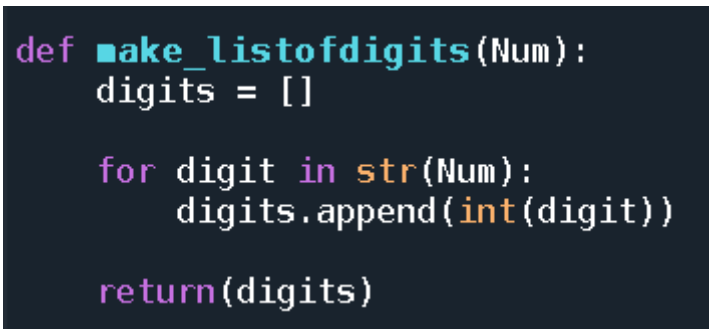


Figure2: Making list of digits with a function.

I prefer  doing processes with list of digits because it is needed while checking number has repeated digit or not. I took a number as a input and I created a empty list which name is digits. Then I put every digit which is belongs to number to ‘digits’ list using append command in for loop. Append command put a new variable to end of the list. For loop works 4 times because we processing with 4-digit numbers. Finally, I returned digits list to use. For example if input of function is 1923 we get (‘1’,’9’,’2’,’3’)

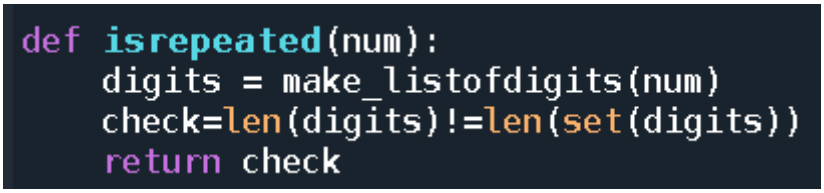


Figure 3: Check are there any repeated digits with repeated function

Secret number can not include repeated digits because of that our guess should not have repeated digits and we take into consideration that while generating set of possible correct values. I wrote a function to check that. Firstly, I got digits array from input number using make_listofdigits

function. ‘len’ function give us to length of array and ‘set’ function change list as a mathematical function. A function can not includes same value more than one times. In other words, ‘set’ function do that if array is equal to(‘1’,’2’,’3’,’2’) it transform to the (‘1’,’2’,’3’) after appyling set function. Consequently, we can recognize a number has repeated digits or not with comparing array of digits and array of digits with applying set function. If both are equal number does not have repeated values. I return check variable which is equal to 1 if length of these 2 array are not equal(number has repeated digits).

```
def Bulls_Cows_Number(Num1,Num2):
    digit_1 = make_listofdigits(Num1);  digit_2 = make_listofdigits(Num2)

    results = np.array(digit_1) == np.array(digit_2)

    bull = sum(results)
    cow = 0

    for item in digit_1:
        if (item in digit_2) == 1:
            cow = cow+1

    cow = cow-bull

    return(bull, cow)
```

Figure 4: Calculation of bulls and cows number with a function.

I wrote a function which calculate number of bulls and cows of 2 numbers. I use this function to check that does user enter us correct bulls and cows value or not. Since if user enter wrong bulls and cows values we may remove correct value the set and algorithm will be crashed so program print an error when user enter wrong values. I do not use this function for guess algorithm.

First of all, function needs two numbers, one of them is secret number, another is guess and I got lists of digits for each number. Then, code line which is :

results = np.array(digit_1) == np.array(digit_2)

I compared all digits of two list according to location of them and if they are equal it give us ‘1’(True) while it gives ‘0’(False) when they are different. For example if digit_1=(‘1’,’9’,’2’,’5’) and digit_2=(‘5’,’9’,’6’,’7’) results will return that (False,True,False,False) and sum(results) gives us bulls because it is equal to how many element is same location with same value in both of lists.

For number of cows, I check every digit_2 digits for digit_1 with using for loop. In other words, I'm going into the first digit list and I'm taking an element. I'm checking to see if that element is on the second digit list. I am increasing the number of cow's if there are any. but I get the same

ones as the location(bulls) .Therefore, I am subtracting the bulls value from the cows value that I found in the last operation.

```
def Guess(Num, set_of_val, bulls, cows):

    if (bulls == 0 and cows == 0):
        if Num in set_of_val:
            set_of_val.remove(Num)

        return(random.choice(set_of_val))



    else:

        for item in set_of_val:
            digits_set = make_listofdigits(item); digits = make_listofdigits(Num)
            if sum(np.array(digits_set)==np.array(digits)) < bulls:
                set_of_val.remove(item)

        for item in set_of_val:
            digits_set = make_listofdigits(item); digits = make_listofdigits(Num)


            val = 0
            for digit in digits:
                if (digit in digits_set):
                    val = val + 1

            if val<cows:
                set_of_val.remove(item)

        return(random.choice(set_of_val))
```

Figure 5: Guessing a secret number algorithm

I created a function to guess processing. I take a guess number, set of possible correct values, bulls and cows number as a input. I divided this process in two stages. In first stage, when bulls and cows value are 0 I just remove this number from set of possible correct values and I do next guess randomly because I have no information when both of bulls and cows are 0.

In second stage, if bulls and cows are not 0, firstly i perform according to bulls.

I'm chechking all the numbers which are in the set of possible correct values and I'm looking at the number of bulls according to the guess number and ‘item iteration’. If the bulls of this two number is less than the bulls number which entered by user, this ‘item’ cannot be the secret number. Therefore, I am removing it from the set of possible correct values. We remove the values that cannot be secret number from the set, so that we are more likely to guess.

Secondly I perform according to cows:

I am comparing all the elements in the set of possible correct values with the guess number. If the number of cows from this comparison is lower than the number of cows entered by the user, this ‘item iteration’ is subtracted from the set of correct values. That's how I'm qualifying inside the set. After that, I generate a new guess randomly between the numbers remaining in the set.

```
set_of_val = []

for k in range(1023, 9876):
    if isrepeated(k) == False:
        set_of_val.append(k)
```

Figure 6: Creating a set of possible correct numbers

In the first part of main code, I defined set of correct values as an empty list. Then, I added non-repeaters to the set of correct values list using the append command using my repeated digit function in for loop according to my minimum(1023) and maximum(9876) possible correct values and created a possible correct values list consisting of 4534 numbers.

```
secret = input("Enter Secret Number: ")

if int(secret)>10000 or int(secret)<1000:
    print("Secret number must be a four digits.")
    print("Game Over!")

elif isrepeated(secret) == True:
    print("Repeated digits are not allowed.")
    print("Game Over!")
    exit
```

Figure 7: Checking of secret number is allowed or not.

Then, I take secret number from the user. I do not use it in guessing algorithm. I just use it to be sure user entry correct secret number(which has 4-digits and nonrepeadted) using if conditionals. If user entry wrong number, game is over because algorithm written according the 4 unrepeated digits numbers.

```
else:

    flag_num = 0 ;try_count = 1

    while (True):
            if flag_num == 0:
                flag_num = flag_num + 1
                Guessed_nmbr = (random.choice(set_of_val))
                print("A: My guess is ", Guessed_nmbr)

            bul, cow = input("B: Number of bulls and cows respectively: ").split()
            bul=int(bul); cow=int(cow)
            if (bul, cow) != Bulls_Cows_Number(Guessed_nmbr,secret):
                print("Wrong inputs entered.")
                break

            if bul == 4:
                print("I WON")
                print("Your secret number is", Guessed_nmbr)
                break
            else:
                if Guessed_nmbr in set_of_val:
                    set_of_val.remove(Guessed_nmbr)
                    Guessed_nmbr = Guess(Guessed_nmbr, set_of_val, bul, cow)
                    print("A: My guess is", Guessed_nmbr)
                    try_count=try_count+1

    print("Number of tries:",try_count)
```

Figure 8: Checking of secret number is allowed or not.

I'm throwing a random guess value without any information at the first input, so I have assigned flag variable to be used only for the first guess. I also assigned the number of try_count to 1 and increased it by one with each attempt, finally I showed with how many guesses the user found the secret number.

After the first guess, I got the bulls and cows values from the user, and I used the split function when I did it, because this function are using to get more than one input. I am checking with the if conditional statement by calling the Bulls_Cows_Number function according to the secret number and guess number, whether the cows and bulls values entered by the user are correct or not.

Because if the user enters the wrong bulls and cows values, algorithm makes the wrong elimination from the set, and the algorithm will be crashed, so in this case I give an error and game is over. I am converting the cow and bull values to integer to make this comparison of the cow and bull values that we get from the user with input function as strings.

If the bulls value is 4, computer has guessed the secret number correctly and won the game. If the number that the user estimates is not correct, I am removing the estimate directly from the list. I'm making a new prediction using the Guess function I defined above. Meanwhile, the process continues until computer finds the correct forecast.

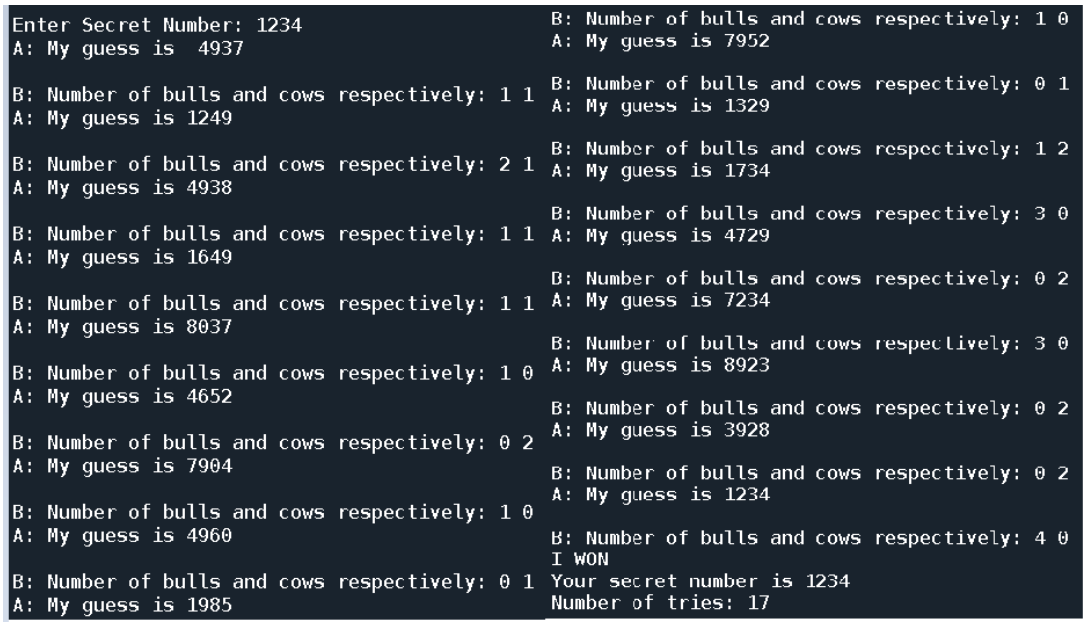


Figure 9: Example of one game.

Figure 9 shows example of one play. Computer found secret number in 17 tries.

```
f = open("readme250206039.txt", "r")
print(f.read())
```

Figure 10: Opening text file which includes rules of game

Text file which give information about game rules is given. To open it python file and text file must be in same file and also this file must be selected if code running in spyder on ‘‘Files’’.

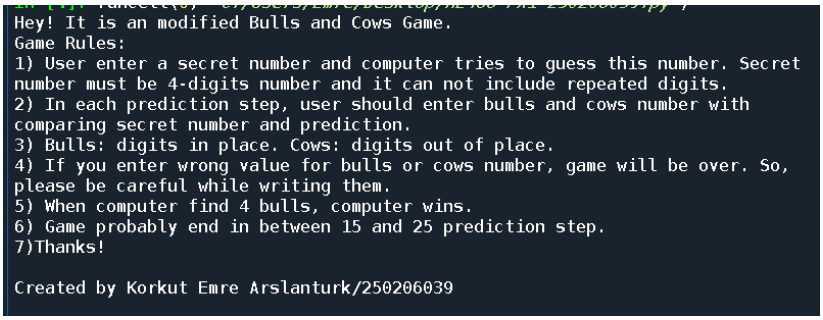


Figure 11: Rules of Game

## 3. CONCLUSION

Our goal in the game is to make the computer guess the 4-digit secret number in minimum attempt number. In doing so, I first determined a possible set of correct values(numbers which do not inclued repeated digits between 1023 and 9876). Then, after each incorrect guess, I removed the incorrect guess from this set. I also removed other set elements which have a value of cows and bulls(when set element compared to the prediction number) less than the number of cows and bulls entered by the user from the set, because it is impossible for them to be a secret number.. Thus, with each guess, we reduce the set of possible correct numbers, and I increase the probability of guessing correctly.

Since the secret number cannot have repeated digits, a function has been written that controls this. This function compares list of the digits and same list after applying ‘set’ function, using len function. When number has a repeated digit if statement give us ‘1’ and we return ‘1’. On the other hand, in order to perform this operation, a function has been written that keeps the digits of the number that is an integer in the digit list. Furthermore,  if the value of cows and bulls entered by the user is incorrect, the correct values may be removed from the set, and it will be impossible to reach the secret number. For this reason, a function that takes secret number and guess number as input has been written to check the cows and bulls values entered by the user.  For the prediction process, a function has been written that extracts the incorrect values from the set described above.

After the experiments, the secret number of the information was estimated between 15 and 25 trials dec It is mandatory to shorten this forecast by adding different expressions for situations such as 0 bulls and 4 cows to the written forecast function.

After some experiments, it can be said that computer can find secret number between 15-25 trials. However, it is possible to shorten this prediction process by adding different statements for different situations such as 0 bulls and 4 cows to the typed guess function. On the other hand, it is a verified fact that the secret number can be determined in at most seven guesses.

## REFERENCES

[1] N.Goel and, “A Mathematical Approach to the Simple Bulls and Cows Code Breaking Game” , Scientific Essay, 2015.