Toan Ngo
Date:5/8/24

Final project

What is this project about:-> Objective

This project is about using the MRV and backtracking to solve the problem of cryptarithmetic puzzles where the number of each letter are equal to a value in [0-9]

Implementation-> Implementation Details

In this project I will be using a nested dictionary with a list and some helper functions.
These are the function name: session_reset, number_choice, resiction_add, check_valid_input, result_check, overwrite, check_sum_valid, new_resic, sum_test, sum_added, get_value_result, roll_back

Function explanation:

>Session_reset-this function is responsible for reset/remove all the value from a selected letter with the letter index as it tracking point

>Roll_back-this function is responsible for extracting the value that is in the prev letter and returning it, it's also responsible for calling session_rest to clear the prev letter output and initialize backtracking.

>sum_added -this responsible for applying the result to the letter.

>sum_test -this is an adding operation. Return the sum of value

>Number_choice- returning a random number that is picked from a list a available number but with a resection that if a number that has already applied to the letter it will not choose it. Return the numbers randomly chosen from the given list.

>Resiction_add- are responsible for checking if numbers are valid and adding the value to each letter. Return a string message.

>check _valid_input- are responsible for checking if the slots for the number the applying number are empty or not, returning a boolean for for each

>result _check- are responsible for checking if the results we get are matching with the conventional problem, in other words it checks for if the results are valid or not before applying the number to it. Return a boolean value

>Overwrite- are responsible for the overwrite value that already exists in the letter with a new value, if the result check passes but check_sum_valid fails.

Toan Ngo
Date:5/8/24

>check_sum_valid-are responsible for checking whether the sum that we get already existed or not, if the answer is true, it will initialize overwrite to fully generate new value and find a new sum that doesn't exist yet. This return a boolean value

>New_resic-this responsible for resetting the stage of restriction number that number generation can generate, it will take all the numbers that are already applying to a value in the dictionary to popularize it list.

>get_value_result-(I end up not using this method and use the sum-added instead) this returns the sum of the letter that has a value assigned to it.

Main code area explanation:

>for my variable i have this:
Letter list: a variable list that keep track the name and the letter that we looking at
list in a dictionary: keep track of the letter and its container.

Restriction list: keep track of what numbers are not allowed to pick.

Valid list:keep track of what number that is allowed to pick.

Boolean value for carry over value switches.

leading case(user input case) switches this to make sure that the carry over does not apply to the first value of the user input.

compare switches to put the function in a comparison mode for comparing the new result with the old result in the dictionary.

runback switches this responsible for initializing a backtrack mode in the code(not used in this code).
Run switches responsible for applying the carry over to value.

>code area:

The main dictionary is a container that is used to store the value and will constantly be checked and modified by the calling of other functions in the above explanation based on the step in the algorithm. The really important thing about the main function in this code is that the pointers that point to the letter are moving backward and mainly the function will work in the else statement after the first if statement run which is used to apply user data input perform check and initialize a number restriction (MRV) for number generate method, afterward every other iteration the if will be skipped.

In the else area: the first thing it gonna is generate a number and perform check on that number if the number are invalid it will go in a loop that contains checking to see what number is

Toan Ngo
Date:5/8/24

valid and the MRV constantly update as well and it will also take a screenshot of the current MRV list which we can use it later, however if the MRV has reach it end and number generation fail, the program will enter a backtrack mode that is “if(number_get=="no more number left")” trigger the old MRV will be discard and the function new_resic() are responsible to check what number are currently in the main container and make those number the new MRV, the reset function will also be call “Session_reset()” its function reset the current letter value back to empty container and before we can do that, the program will move the pointer current location back to the prev location and extract the value of the letter and put it in the new MRV list, generate new number that are not in the MRV list and perform a check of “Resiction_add()” this will loop until a new number is found, and in here we also handle carry over value from user input as well and set the appropriate state for the list of carry over value.

After the number has been picked and added to value, the program will perform a summation of the current letter and apply the carry over of the state to allow it.

However if the summation of both value reach a invalid number then the program will trigger overwrite mode which generate a compelled new number and with the prev number get push in the MRV list and perform all the other previous check operation again and applying the new summation with the inclusion of a newly added carry on number check.

After the main letter loop reaches its end, we take whatever carry-on value that is left in the list we created and add it to the final letter in the result including the default 1 carry-on in the cryptarithmetic puzzle. Finally the function should reach it end and return a dictionary that was modify with the value applying to each letters and from that on we can perform one final quality(edge case) check making sure that all the number are valid, if not, run it again until it again until it quality are acceptable, then afterward we format the output and print it out.

Running the program-> Running the program

Inorder for you to run this program you must type this in the command prompt:
python final_project [number here]
You also you must be in the folder that this files is in for this program to work
This also requires you to have python version 3 or python 3.10.8 either should be fine.