**Task 4**

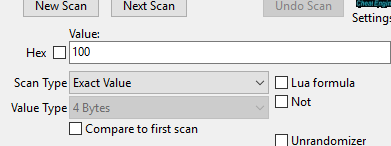
**Group: Isaias Leos, David Amparan, and Alex Vasquez**

Task 4 – Add Invincible Health Functionality

Your task is to add the ability to toggle the invincible health cheat functionality. With this cheat enabled, the player will not take any damage. Please note: Make sure you test the functionality extensively; even if you change the numeric value in the doom display, it is not necessarily the case that the player has invincible health.

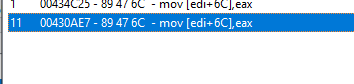
This task is more difficult than the previous and you will have to trace through some assembly (it is recommended that you use Cheat Engine’s memory view window to do this, but you may use any tool of your choice)

1. Our first task again is to find the memory that holds the value of health and the instruction that writes into that memory to modify it
   1. Like the previous tasks we attack Cheat Engine to chocolate doom and begin taking damage to find out what memory is changed. We look for the value 100 since we visually see that is its starting point.



* 1. We then see multiple addresses that are changed whenever we take damage as well as increase whenever we get a health up. However, for this cheat we are focusing on the reduction of health. Let us choose 029AFE14. Now we click on what writes into the memory and take damage once more to prompt the instruction/s.

* 1. Instruction 00430AE7 is the instruction that occurs every time we take damage so let’s begin there. Opening the disassembler, we see a subtraction instruction, which leads us to believe that the health must be reduced there. Part of our cheat will NOP this address so we do not loose health. (Let us remember this address for now)





* 1. Going back to the instruction we found 00430AE7, looking down we find test exa, eax. This instruction is taking what ever we have in our health and computing a bitwise and to itself with no modification to eax. This instruction then must also be NOP. (Again, save the address)



* 1. Through testing, we found that these modifications would not affect the visual health and we’d still die when health hit 0, visually.

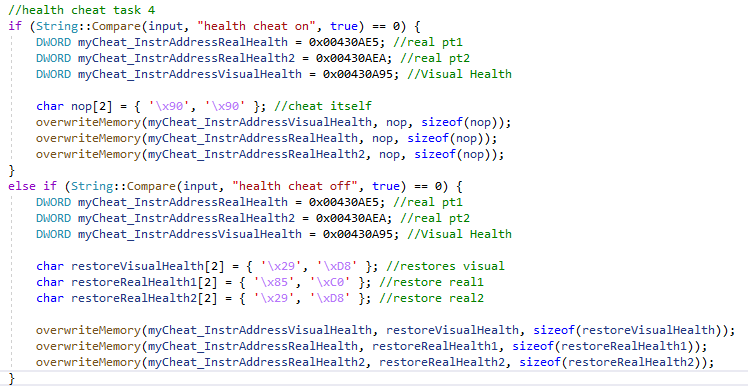
1. Find the visual health memory and modify it
   1. We attach the debugger onto address 0050D1D0 and open the disassembler for the instruction 00430A97. Effectively, this instruction modifies our visual health memory, but we must find what makes that change. the operation.
   2. Thanks to the disassembler we see a subtraction being down on eax, which then is passed and modifies our health.
2. Write cheat into the DLL
   1. We have three addresses/instructions that we must NOP to not loose any health. Visually we use 0x000430A95, for real health it is split into two addresses, 0x00430AE5, 00x00430AEA. Let us create the instruction address variables, those will be overwritten.



* 1. We also need the string comparison to verify we would like the cheat on or off. This can be copied like the one from task 2. Now, lets write instructions for the cheat which is simply 2 bytes of NOP instructions. We need 2 bytes since the original code is 2 bytes long.





* 1. We can then create some if statements, cheat on and cheat off. You can then use overwriteMemory() to turn on the cheat, add the NOP’s, or to turn off the cheat, adds the original code back in.
     1. This is the source code attained: