**PASSWORD 1**

For the first passcode, I basically just followed what we did in the lab. Given that there was no strcmp function called, I saw that there was something that looked like a comparison called cmpsb and there were arguments passed to it. I first set a break point when this function was called. Then I looked at the register %ecx but this returned address out of bounds. Then I looked at the register %esi which returned “gdbYovgAUzEzAfeRukPuLZP”. I entered this as the password and it worked.

**PASSWORD 2**

For the second part, I disassembled main first. Nothing really stood out to me, so I started to look at the values that were passed to the functions c, p, and s. Upon doing this I got nowhere, so I dug deeper. I figured out that the c function just removes the new line character, as it returns the string with the new line character removed. I found out by slowly working through the S function that S just seems to count the length of the string to make sure you have at least a length of 15. The P function goes through the string character by character seeing if there is only one character in the string. If you do something such as “aaaaaaaaaaaaaacaaaaa” the password will fail as there is a ‘c’ that is not the same as the ‘a’. The password can be any single character that is repeated at least 15 times and less than 100. Once you pass 99 of the same character, you can put in any character as the program is only reading the first 99 characters to check. The character you use can be any letter, number, or punctuation, including ‘!’, and the space character. There can only be one character in the first 99 characters.

**PASSWORD 3**

The third part was much harder than the first two. I didn’t even know where to start to disassemble at first. I eventually used the info functions command to list out all of the functions, which at least gave me some ideas on where to start but I still had no real clue where to go as the functions had such strange names. After looking through a majority of the functions and seeing that there was nothing that really stood out, I decided to use control c once the program was running to see where the program goes after input was entered because I assumed this is where the password would be checked. I got lost again as I was not sure where the string was even stored when the kernel\_\_syscall function took in the input. After stepping through the functions one step at a time I decided to look at the getchar function as I figured this method at the very least got a single character and sent it somewhere else to compare it to something. As I went through the getchar function, I checked the registers to see what they held to see if anything stood out. I saw at one point, the line 0x00ca6037 pulled a ‘s’ from the edx register. I then tried a lot of different passwords such as s, mass, etc. I eventually put in mas682\_3ssss and it worked. As I worked through more, I got confused about exactly what was needed for the password. At one point I thought I needed a a certain number of letters to characters because something such as aaaaa9ssss worked but aaaa1ssss did not work. Eventually I figured out I needed 5 s’s as I tried one s, then two, then three, and so on and noticed that only 5 s’s worked. But looking back at aaaaa9ssss, I saw this worked but only had 4 s’s. I knew there must have been some other requirements to this. As I looked at passwords that worked and those that didn’t I noticed that the ones that worked all had the same characters. Some of them, such as ‘a’ did not work, but when I tried 5 9’s this worked. Eventually I went through all the letters, numbers, and symbols testing 5 of each until I figured out the password needs exactly 5 characters that are c, s, 4, 9, and 0. You could have 5 of one, or a mixture of any of them, but you cannot have more than or less than 5 of those characters. I still was not satisfied as I wanted to know how this was working, and where was it keeping track of the number of times one of those characters was used. I could not figure out exactly how it was counting them, but I noticed getchar pulled out the characters s, c, and 4 at points in the assembly code even when I did not have one of them in the password. I also noticed getchar returned to a function that was called “??”. I was not sure what this function did as I could not disassemble it, but I assume it had something to do with counting the number of times one of those characters appeared. To conclude, the password must consist of a mixture of 5 of the characters c, s, 4, 9, and 0, with c and s being lowercase. Any other characters can be included. If you have more than 5 of those specific characters or less, it will not work. I also noticed the program only looks at the first 16 characters, thus the length of the password must be at least 5 and less than 16. If the length of the password is less than 16, you must hit enter until you reach 16 characters as the new line character is counted as a character.