Part 1:

Solved with: “xXOZxrhuRhvxfZHwNKYDuxXFhsBcI”

This first part was pretty straightforward. It was very similar to our recitation where I just had to find the register that was holding the string for comparison and us the “x/s $” command to read the current string in that register. After entering “something” into the prompt, I was able to see that around comparison, specifically the repz command, that %esi contained by input and %edi contained what it was being compared to. Once I found this then I was able to check those two registers and see the password key.

Part 2:

Solved with: When the file was named “chs171_2” It was solved with chs171, but changes based on file name. File must have “_2” at the end or it will not work.

For the second part, I initially saw that the whole file path was being held and figured that it was standard for everyone. When looking at what returned from the “r” function, I saw that the file path was then I found in another register “_2” and then thought that it must be taking off the _2 of the original name if there was one and then comparing the two values. I thought I was a genius when I entered my username on the first try and got the correct answer. It seemed all too easy though, I quickly changed the file name to something that didn’t end in _2 and found the password no longer worked. After going step by step in the file I discovered that the file name was actually not getting changed at all for the string comparison. What was actually happening was that the input was getting “_2” added on to it. This would explain why “chs171” worked as a password initially. I renamed my file “testing” and was not able to create a good password because “_2” was being added on to every one of my inputs. While I was able to initially solve it, finding out more about the functions led me to believe that unless my file name ended in “_2” it would be impossible to crack.

Part 3:

Solved with: “%%%%%%%%%f” as well as any password of length ten with any combination of exactly 9 of “%/*-+” and any character.

Solved with: I had a significantly hard time with this one. First off, I realized I could not just use “b main” since there wasn’t one. I instead had to do any objdump and enter an “info files” command in the gdb in order to get the proper addresses of the .text file. I then used those two ranges to disas and go through the file step by step. This did not prove to work either most of the libraries were dynamically linked. Therefore when I went to go access something it would be out of bounds or there would be nothing to access. I decided to disassemble the program and just go through it the old fashioned way and see if I noticed any instructions. The first thing I noticed was a loop that looped through 9 times. This was a cmpl at 0x80484d5. This made me think that whatever my character(s) was, 9 of them would have to be present. I also then saw that it wanted a value of 10. I figured this meant that the length had to be exactly 10. Now I had to try and figure out what exactly these characters where and why there were 9 of them. I noticed a subtraction of 0x25 that was occurring and then a number of logical functions. This value was then compared to 0x18 and a jump was called from there depending on the value.

I figured this is where I would find my character solutions. I kept trying different iterations of the ASCII table that I would be able to subtract hexideicmal values from and then compared them to 0x18. I then decided to just try "%" symbol since that was 0x25. My thinking was that maybe if the value 0 was then it would have an effect on the operators and hopefully unlock my password. Fortunately I was correct! I then tried everything below that value, and only some characters worked. Finally I decided to try the rest of the symbols and found that any iterations of "%-+/*" and another character, as long as there were 9 of those symbols, worked.