Problem 1

I started off by putting all three of the programs as input into mystrings. If the password were typed in directly to the c file, then it would show up as a string in the executable. I ran all three and looked through the strings that were output. Obviously, all of them had the "Congratualtions!" and "Sorry! Not correct" strings. The first one was the only program with an extra string which was "GVayFIHWfcoFTcgyxQiyhJTKH." This turned out to be the password.

Problem 2

Since the password could not be found from using the mystrings program, I decided to use gdb to look at the assembly code. I created a breakpoint at the main and then used disas to look at the assembly code. In the code, I found that strcmp was used. This meant that the input and the password were compared at this point so I created a breakpoint there and looked in the registers. I found "cwj11_2", "_2", and my input with "_2" added to the end of it. I guessed that the program added "_2" to the input and compared it to "cwj11_2". So, I tested cwj11 as the password and it worked.

Problem 3

I first tried to use the gdb debugger again but the main isn't saved in the assembly code. I was unable to access the assembly code using this because I couldn't make a breakpoint at the main. Instead, I used objdump -d to create the assembly code. Looking through the code, there was no sscanf or strcmp. Instead, I found getchar(). I knew this was how the program took my input so I used this as a starting point. From there, I was having trouble accessing all the data stored in memory so I decided to trace through step by step and check all the registers. I found the point where my input was stored in $eax. I found there was a comparison and a counter being used. Looking at how the code worked, I found that a comparison was used on each char entered, and a counter was incremented based on this. If all five chars came out true on the comparison, the password was accepted. For it to be true, the value of $eax had to be equal to 0, 1, or 2. Before this happened 0x58 or 88 was subtracted from $eax as seen in sub $0x58, %eax. This meant $eax had to be equal to 88, 89, or 90. This is X, Y or Z as a char. Thus, the password is any combination of X, Y, and Z.