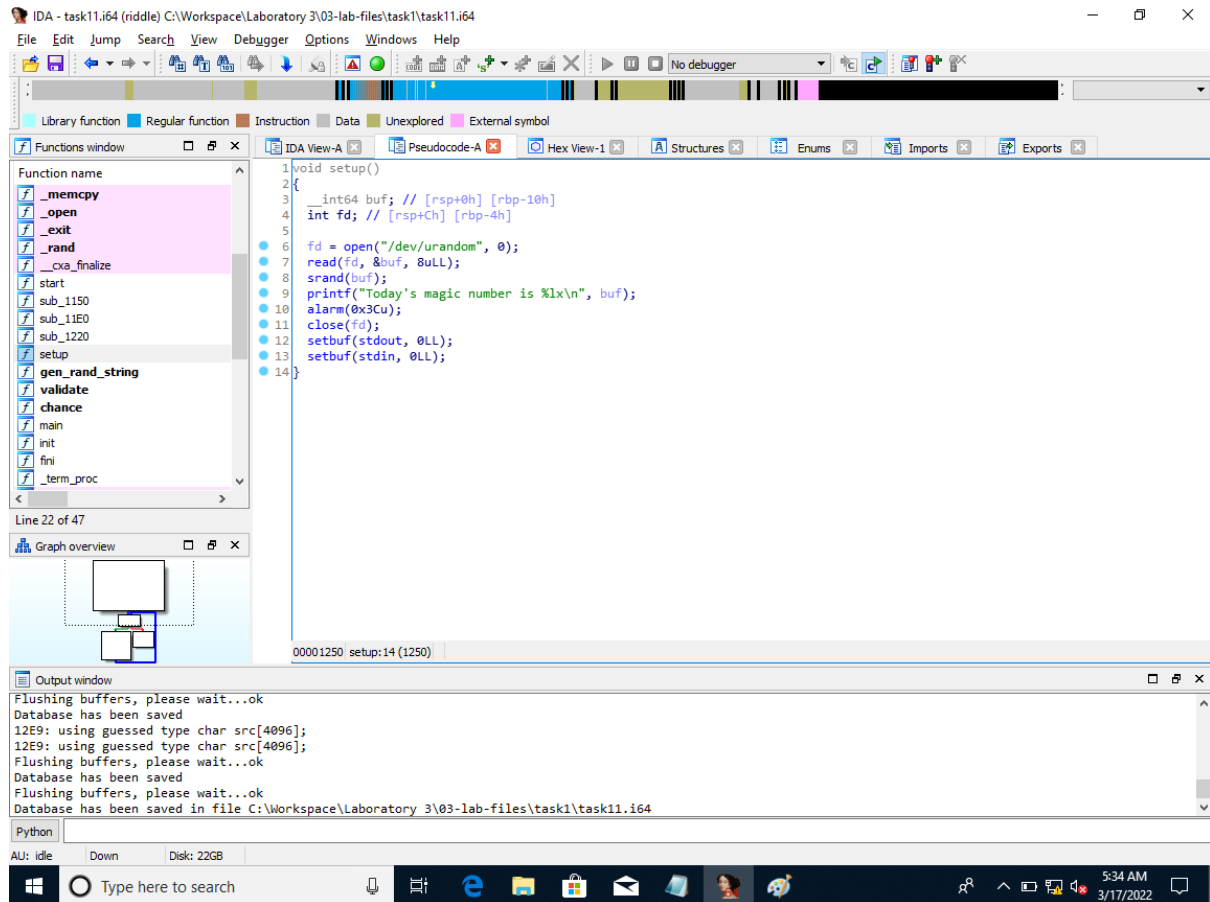
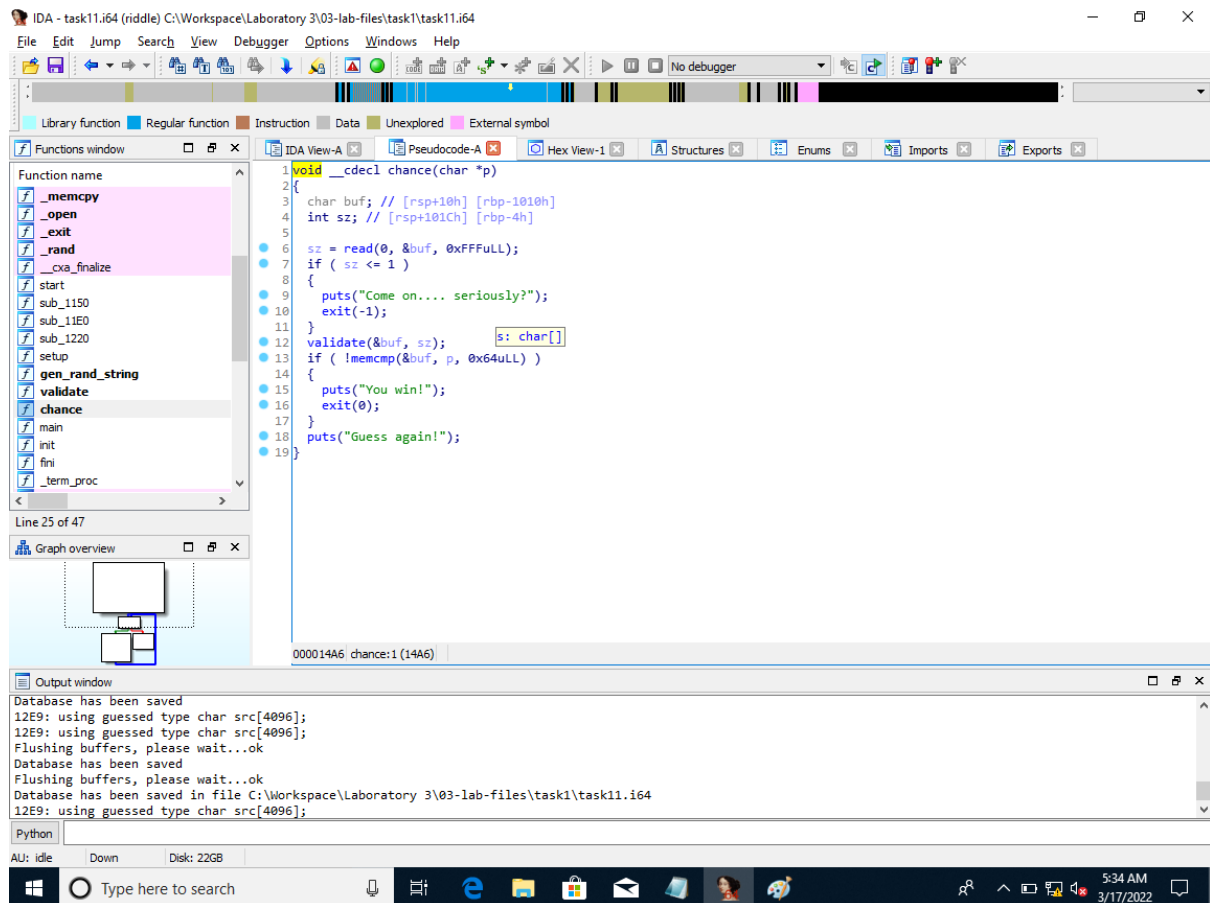
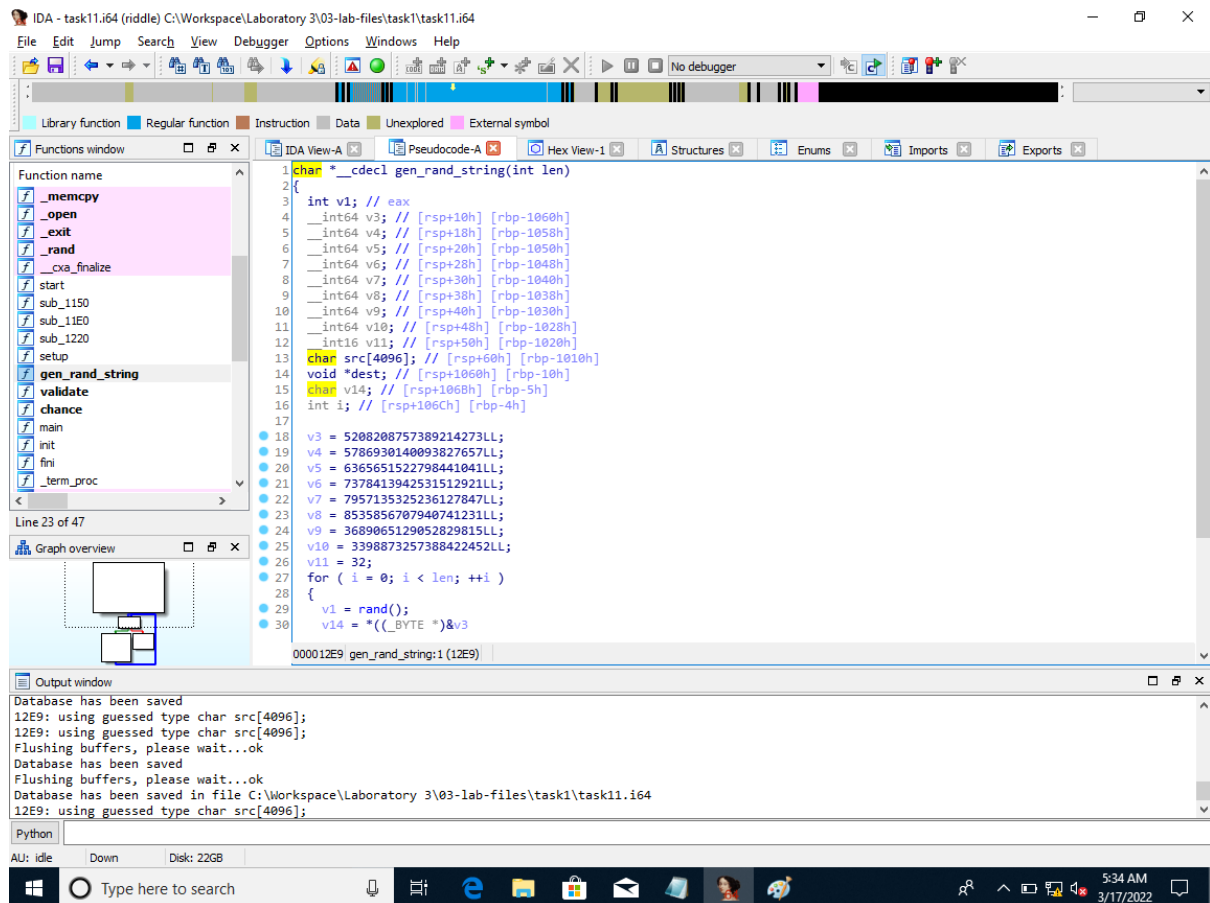


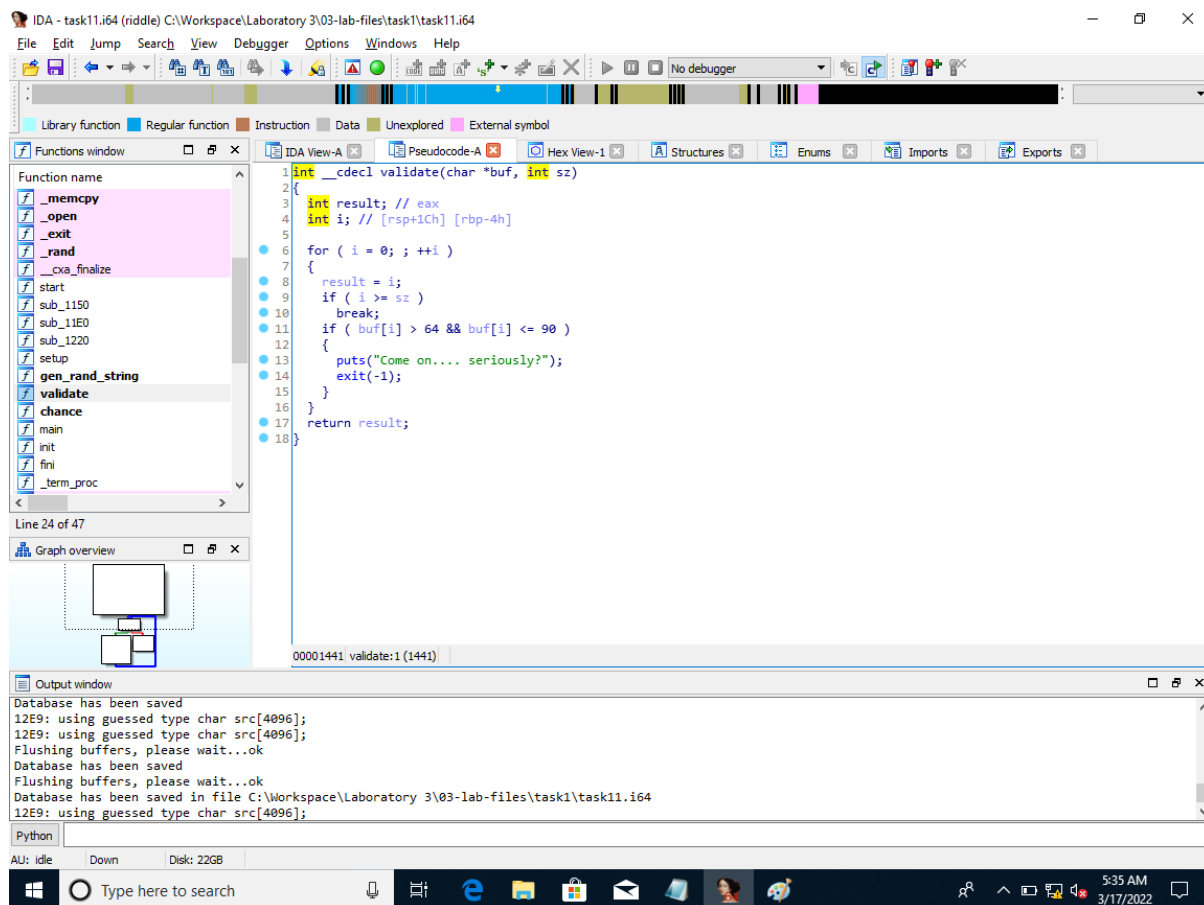
Task 1: Reverse engineering with spoilers

- Rename and retype the 4 functions in the source code (aside from main()) (3p)

The IDA file containing the renaming and retyping of the functions is task11.i64. I've identified all 4 functions: setup, gen_rand_string, chance and validate. Below, there are screenshots from IDA for each function.

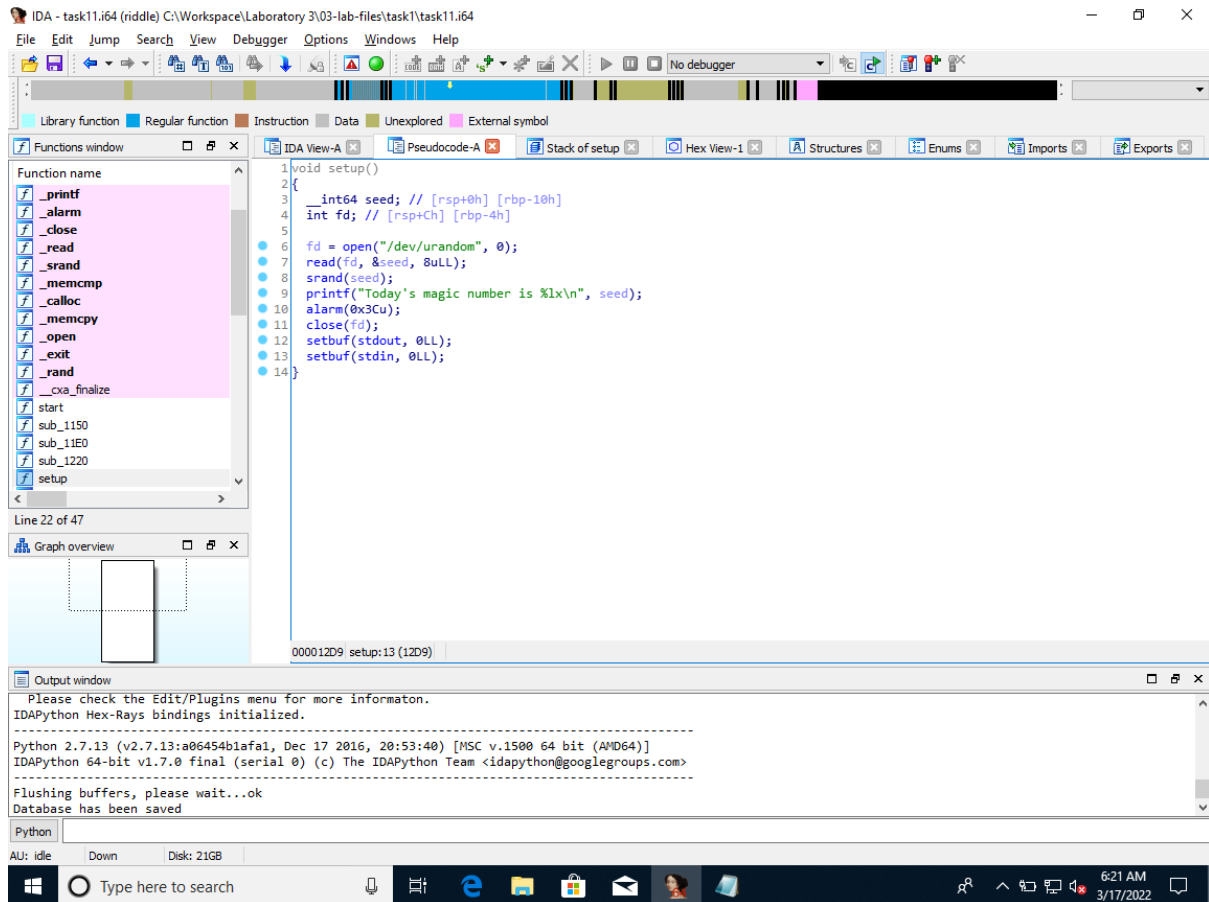


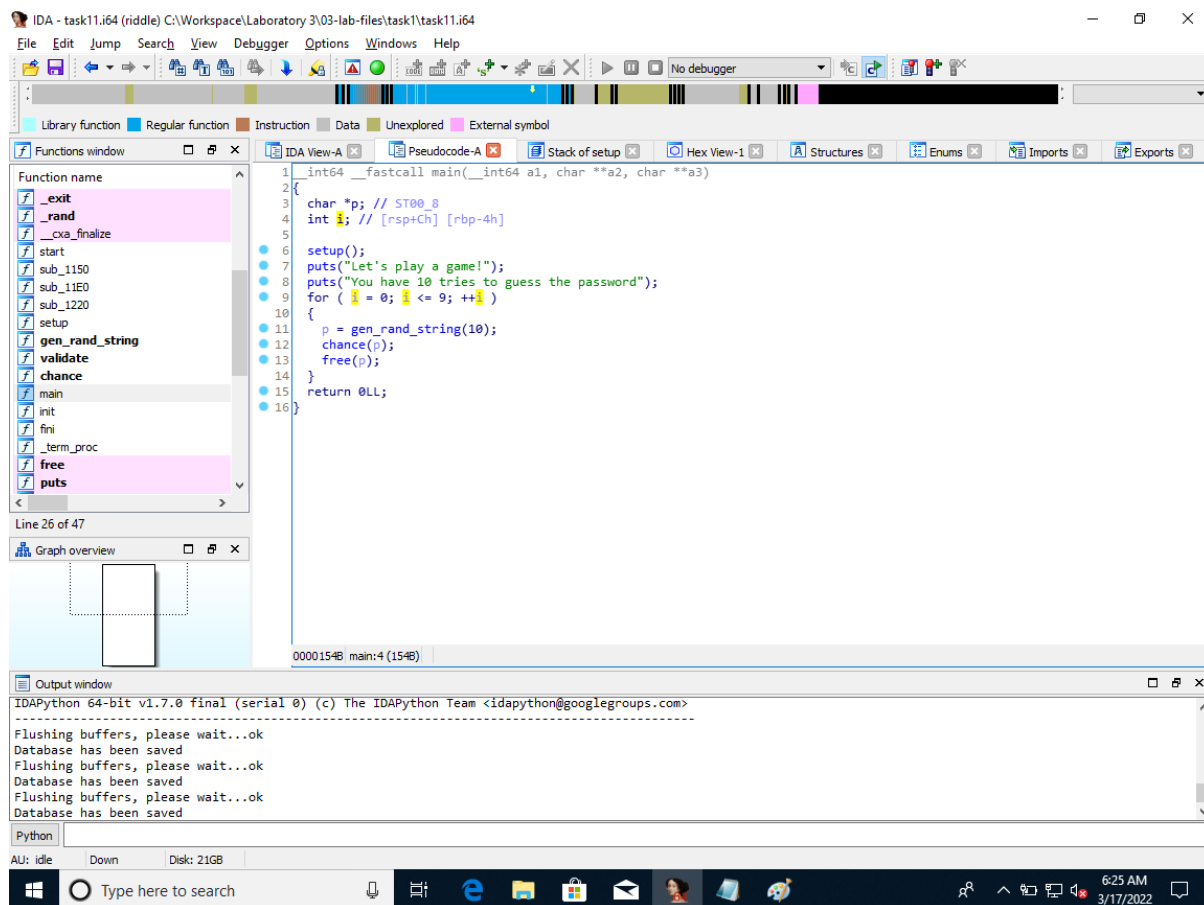




- Rename and retype the stack variables in **setup()** and **main()** (1p)

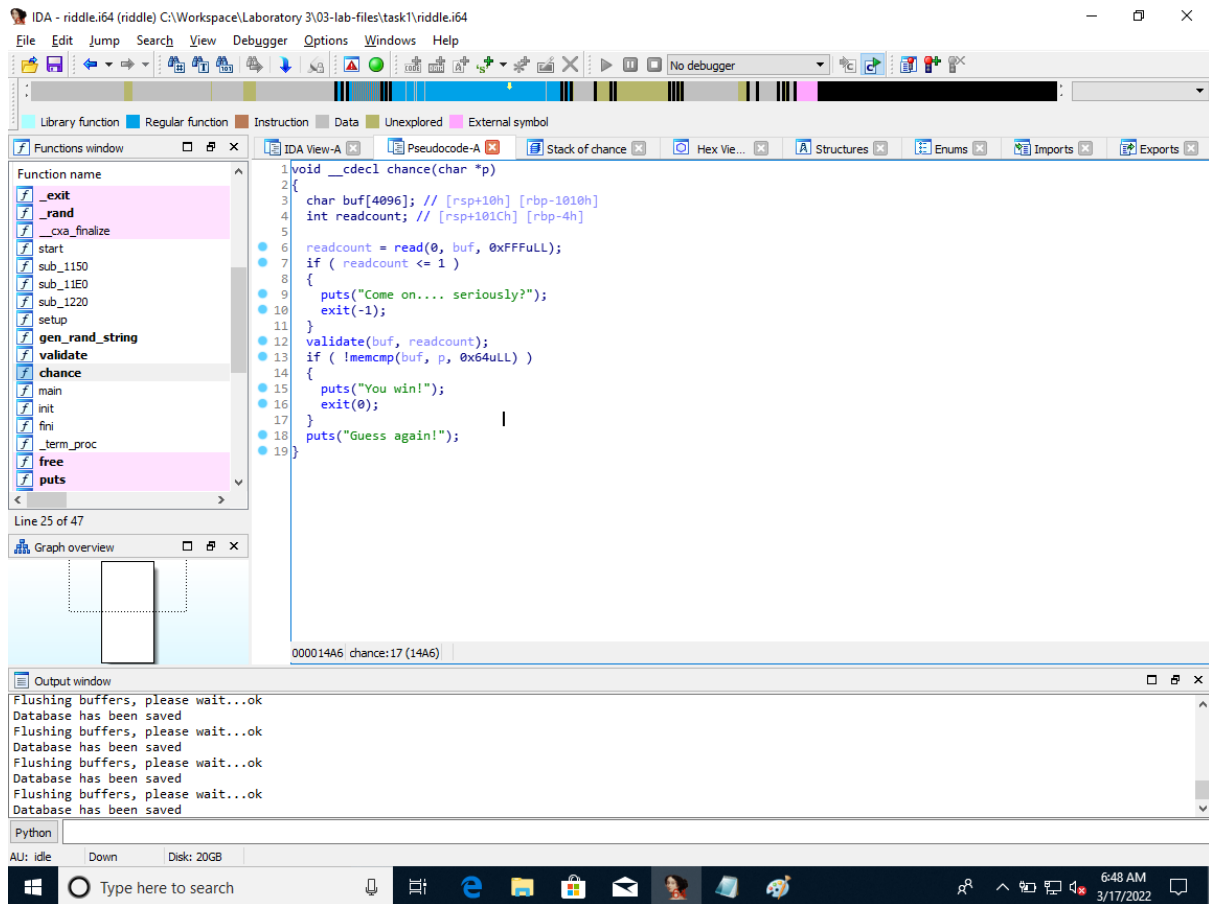
The IDA file containing the renaming and retyping of the variables from setup and main is task12.i64. Below, there are screenshots from IDA for each function.

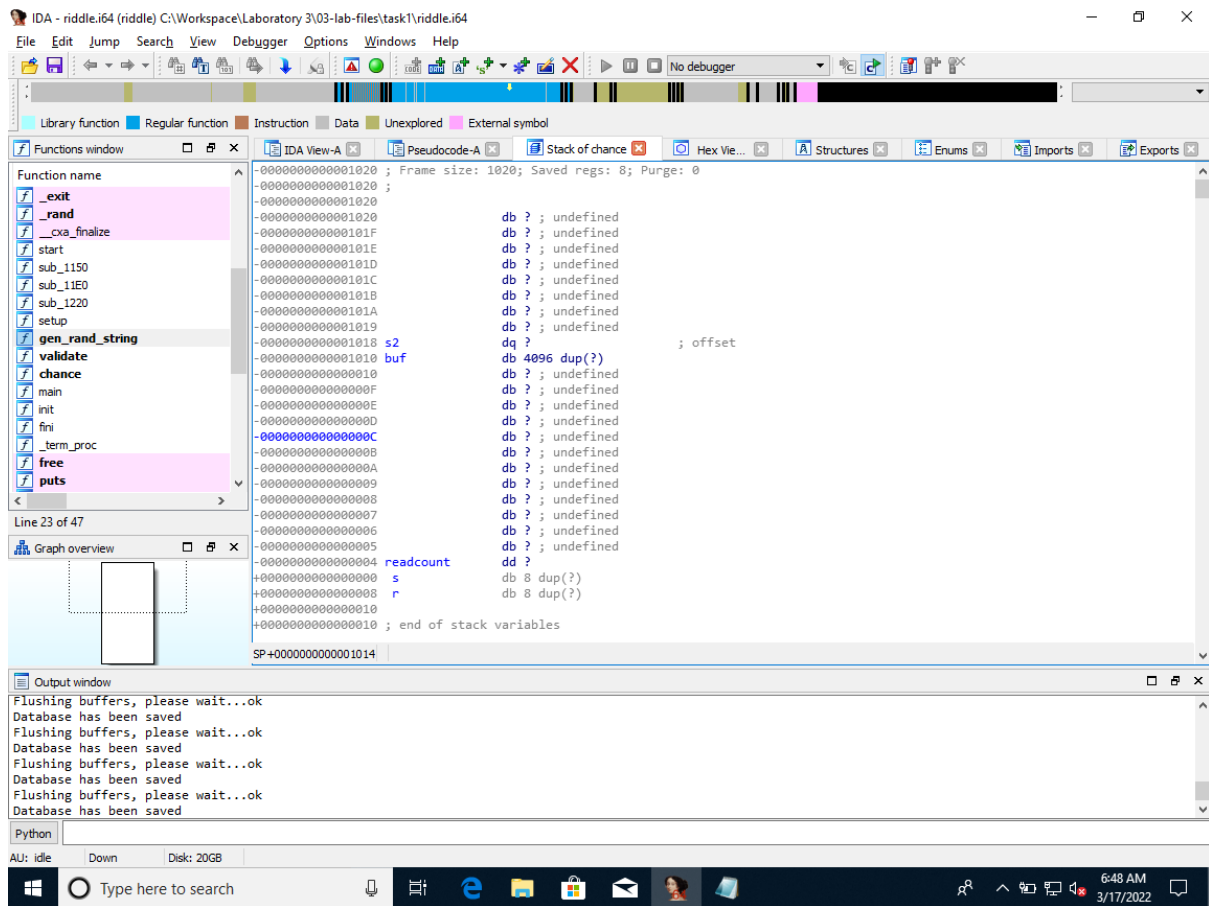


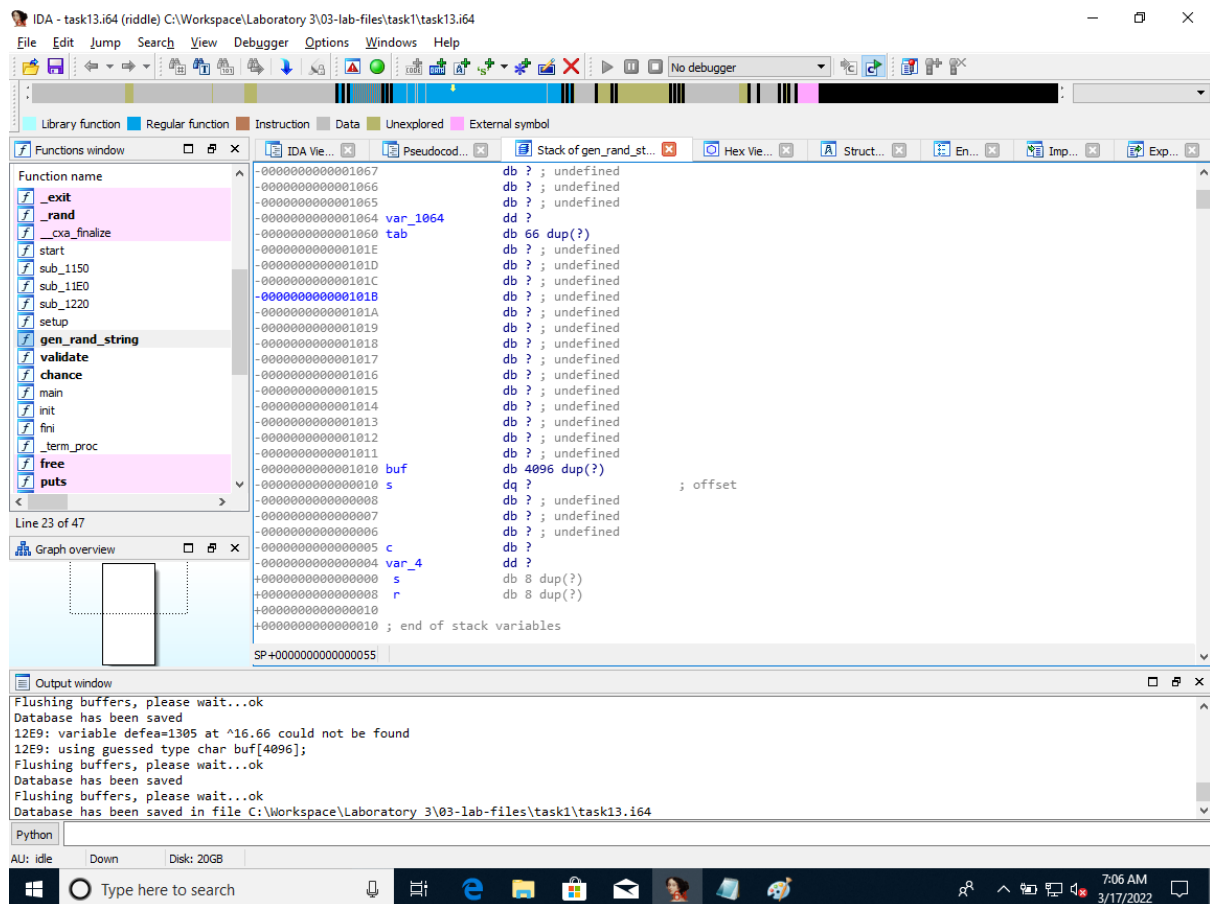
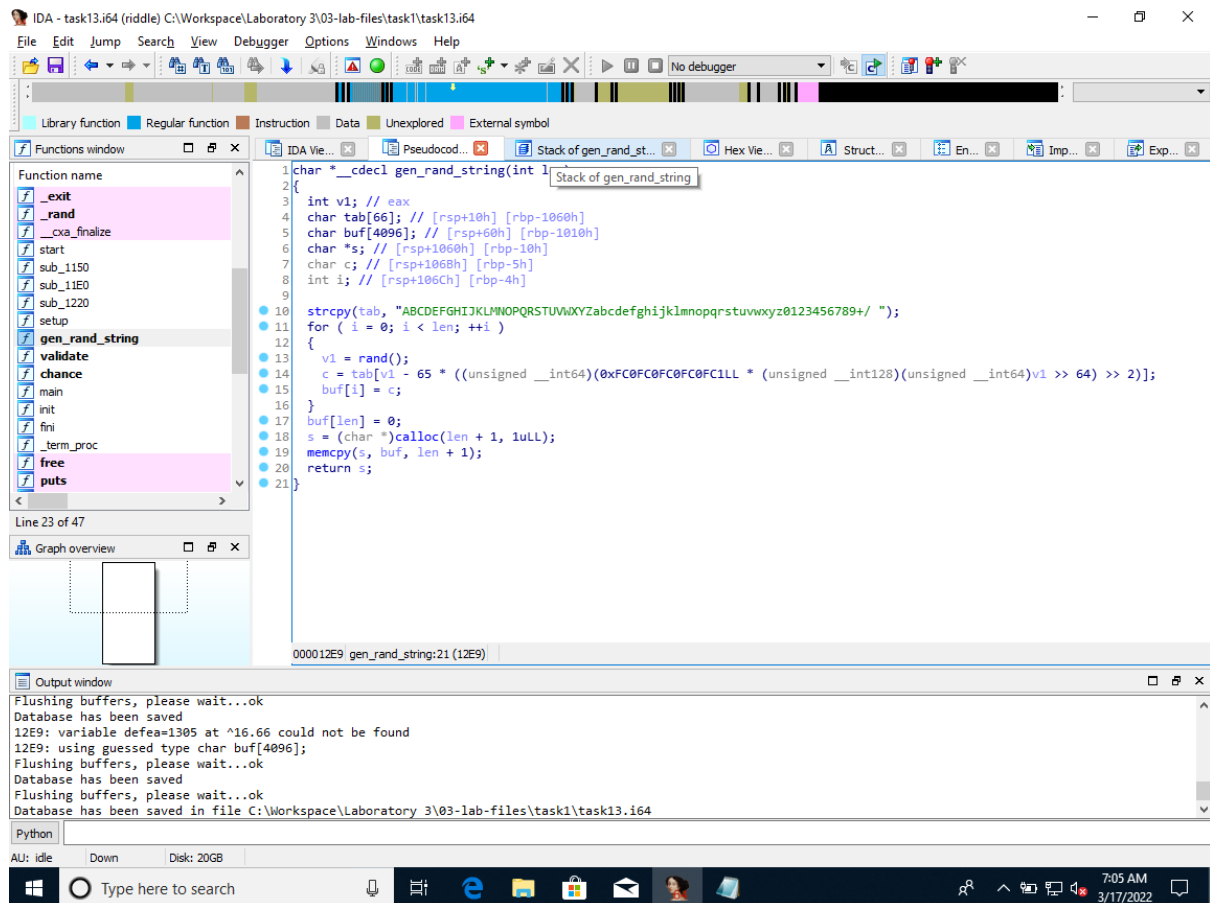


- Rename and retype the stack variables (including the arrays) in **chance()** and **gen_rand_string()** (2p)

The IDA file containing the renaming and retyping of the variables from chance and gen_rand_string is task13.i64. Below, there are screenshots from IDA for each function and for the stack of each function.



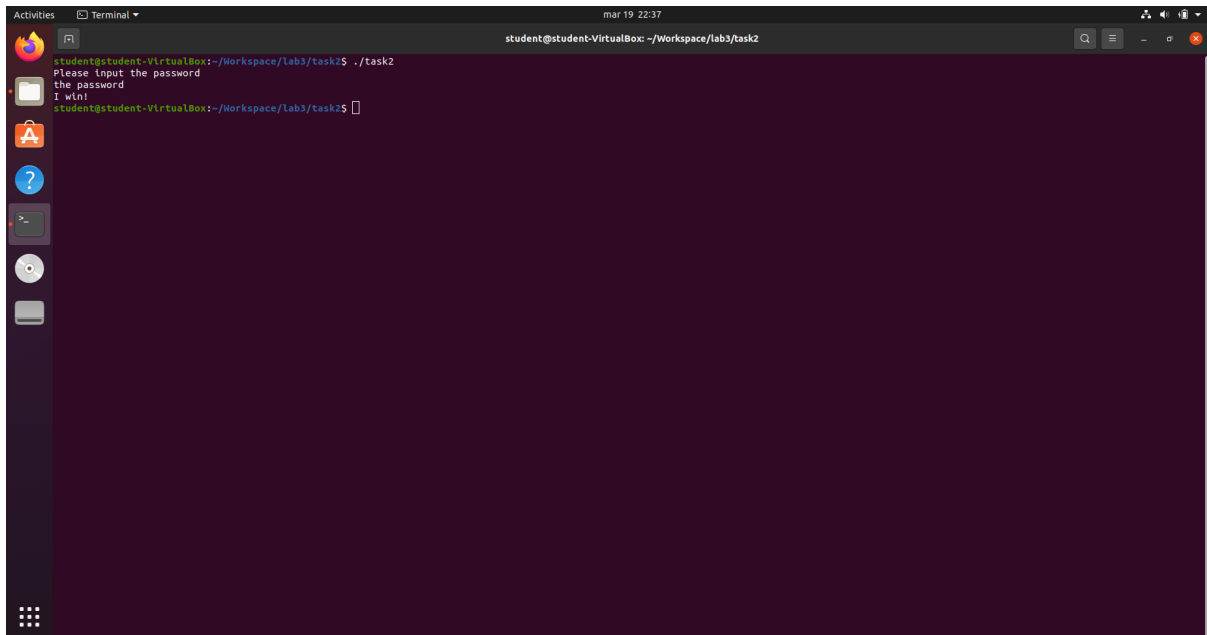




Task 2: Statically linked crackme - graybox analysis (dynamic + static)

- Run the program once, note any strings. Go to the **.rodata** segment (**Ctrl-s**) and find any/all of the strings. Using the xref functionality, determine where the **main()** function is. **(1p)**

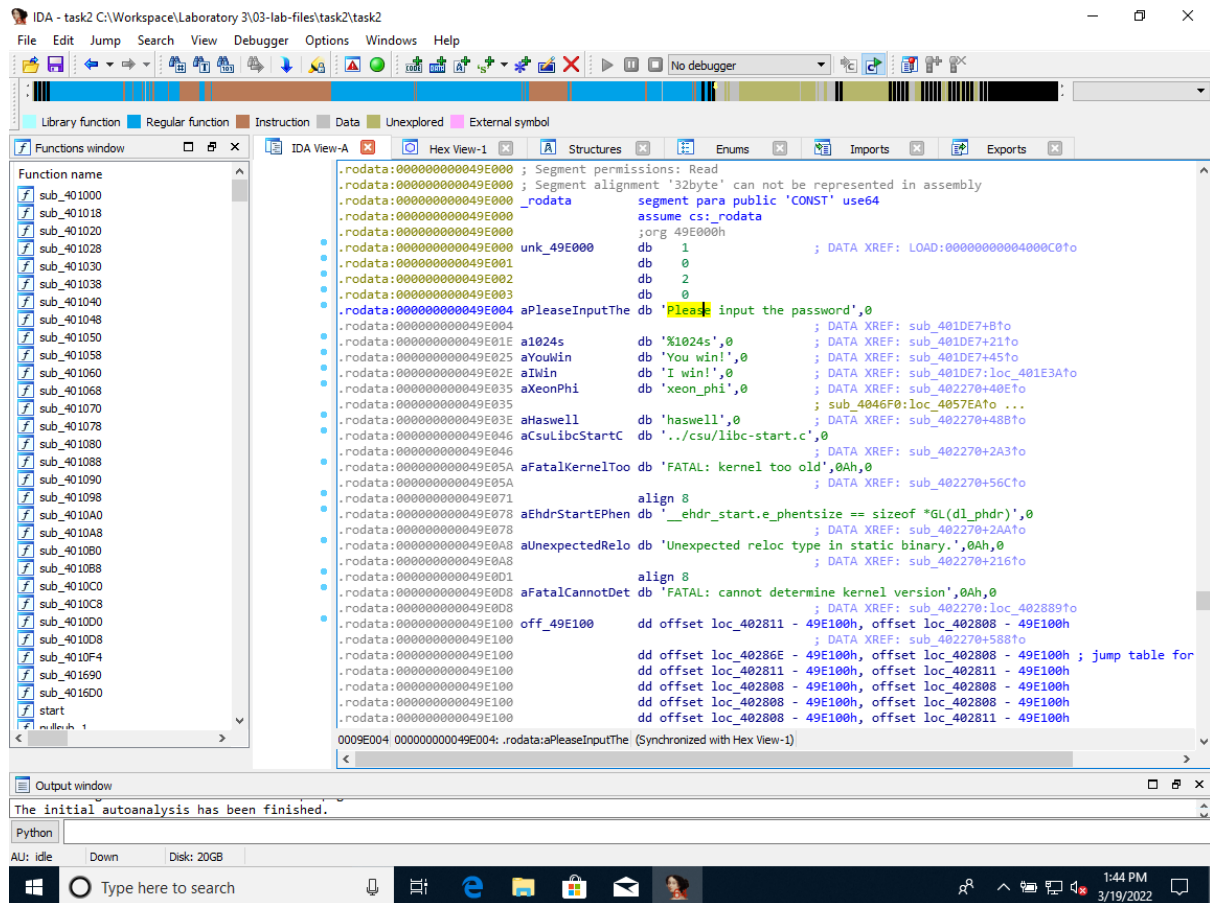
After I ran the program I noted two strings: "Please input the password", "I win!".



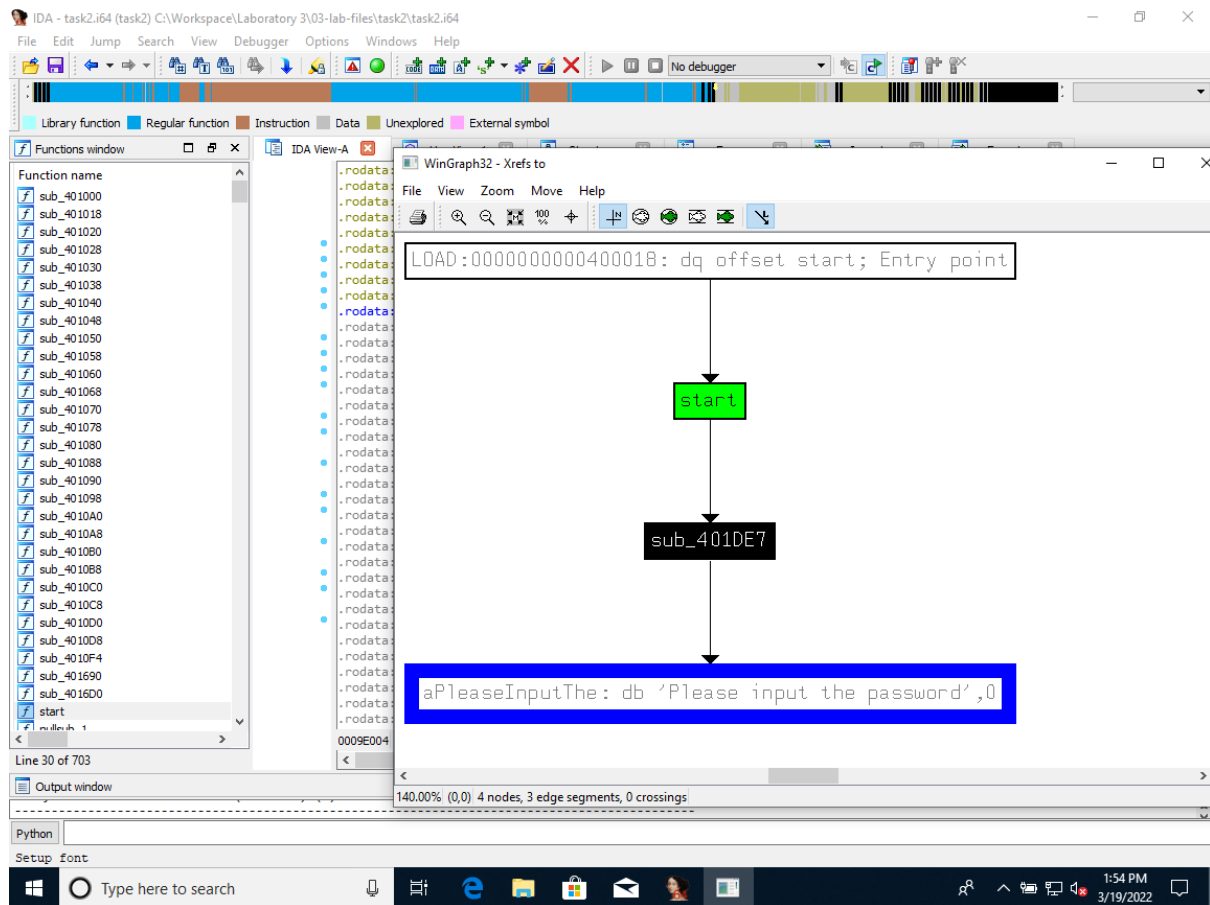
The screenshot shows a terminal window titled "Terminal" with a dark background. The prompt is "student@student-VirtualBox: ~/Workspace/Lab3/task2". The user has entered the command `./task2`. The program's output is as follows:

```
student@student-VirtualBox:~/Workspace/Lab3/task2$ ./task2
Please input the password
the password
I win!
student@student-VirtualBox:~/Workspace/Lab3/task2$
```

Afterwards, I looked in the .rodata segment, where I identified the previously mentioned strings as well as the string: "You win!".

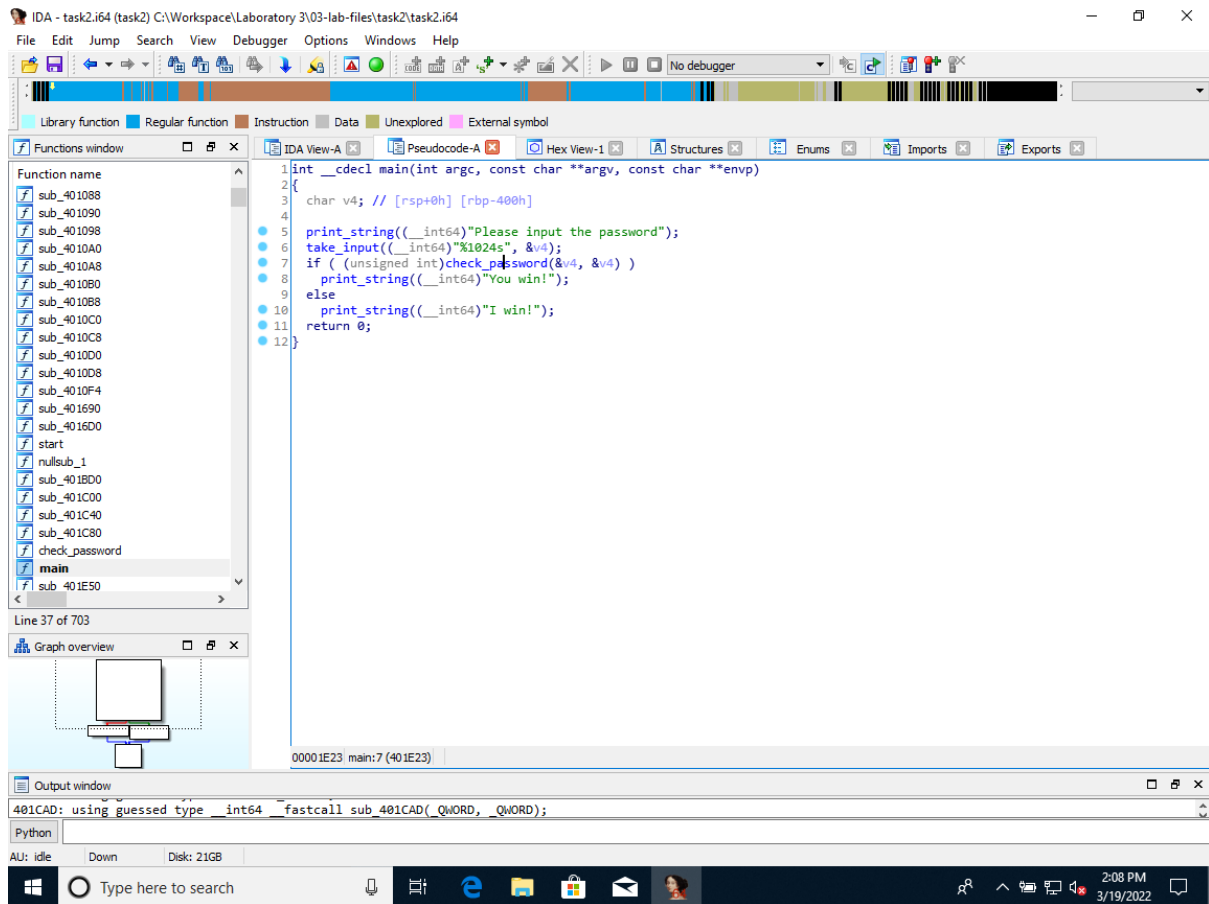


The next step was to use the xref functionality on the strings, which gives us a graph with the function calls until the usage of the string. The first function after the “start”, in our case sub401DE7, is the main function.



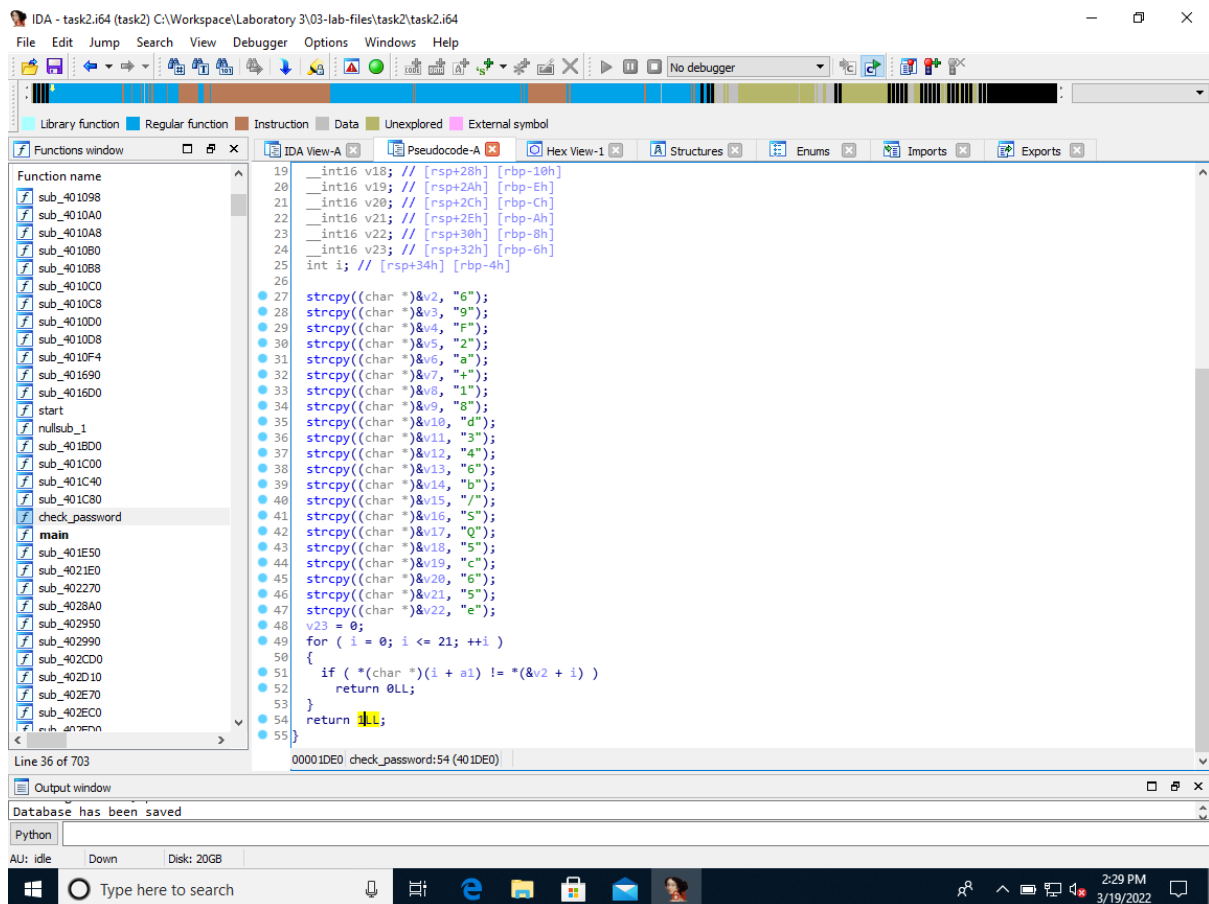
- Rename all the functions in **main()** and determine the password checking function. (1p)

After looking just inside the main function, I was able to tell what was the purpose of each function (I didn't look inside individual functions). The following screenshot shows the main function after the renamings.



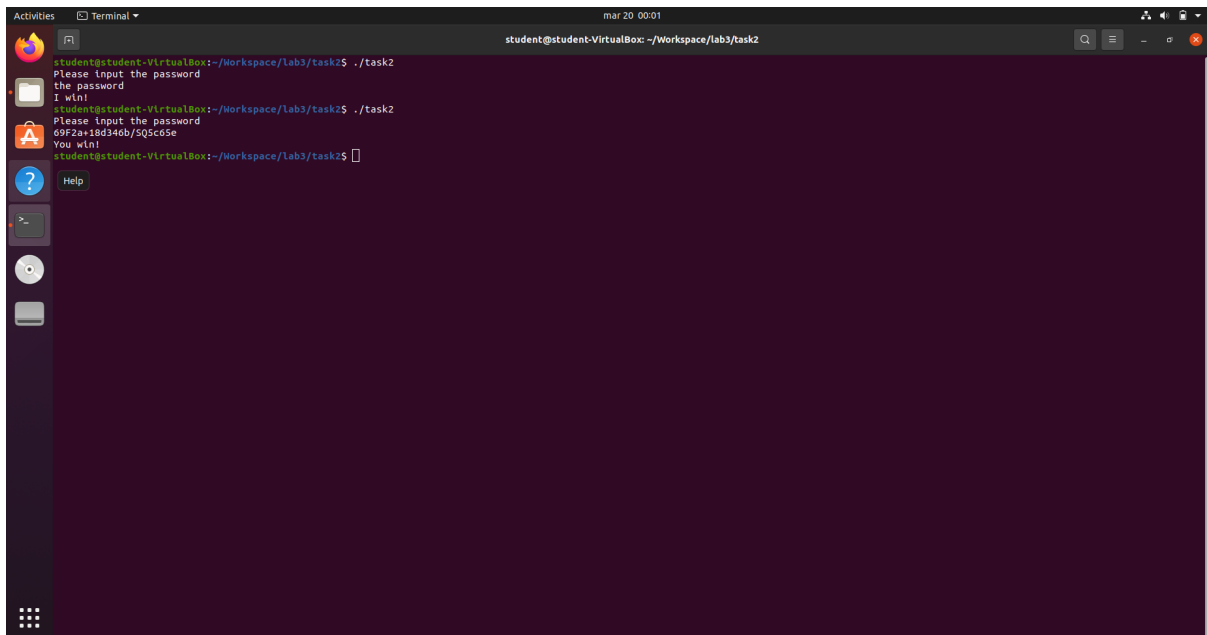
- In the password checking function, observe how the correct password is generated; we want to make this function more legible.
- Go to the location of any **word_.....** variable in IDA-view and find the location of the start of the alphabet and redeclare that address as a wide C string (**Edit->Strings->Unicode**).
- Again, in the password checking function, observe how the right-hand-sides look now. Redeclare the alphabet with the “const” modifier at the beginning. This should collapse the function and reveal the correct password. Finally check that the password is accepted (**2p**)

After looking at the stack of the function that checks the password, I observed that there were some addresses that hold the alphanumeric characters. After following the steps in the requirement, I was able to uncover the password as “69F2a+18d346b/SQ5c65e”.



The IDA file for this task can be found in the file task23.i64.

In the end, I introduced the discovered password to confirm its validity; the output is shown in the following screenshot.



The screenshot shows a terminal window titled "Terminal" with the date and time "mar 20 00:01". The user is logged in as "student" on a machine named "student-VirtualBox", and the current directory is "~/Workspace/lab3/task2". The terminal displays the following sequence of commands and outputs:

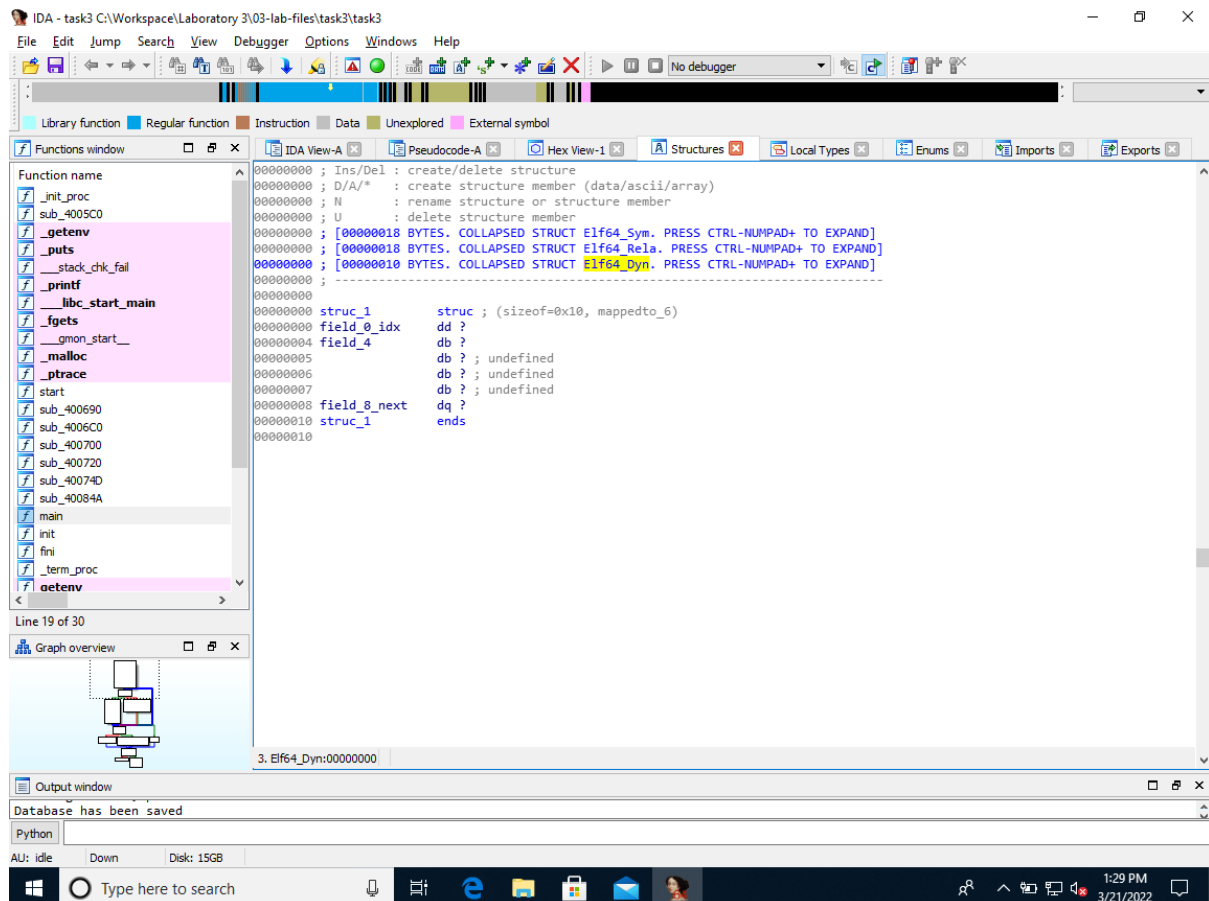
```
student@student-VirtualBox:~/Workspace/lab3/task2$ ./task2
Please input the password
the password
I win!
student@student-VirtualBox:~/Workspace/lab3/task2$ ./task2
Please input the password
69f2a18d340b/5Q5c65e
You win!
student@student-VirtualBox:~/Workspace/lab3/task2$
```

The terminal window has a dark purple background. On the left side, there is a vertical dock with several application icons, including a question mark icon and a "Help" button. The top of the window shows the "Activities" menu and system status icons.

Task 3: Data Structures

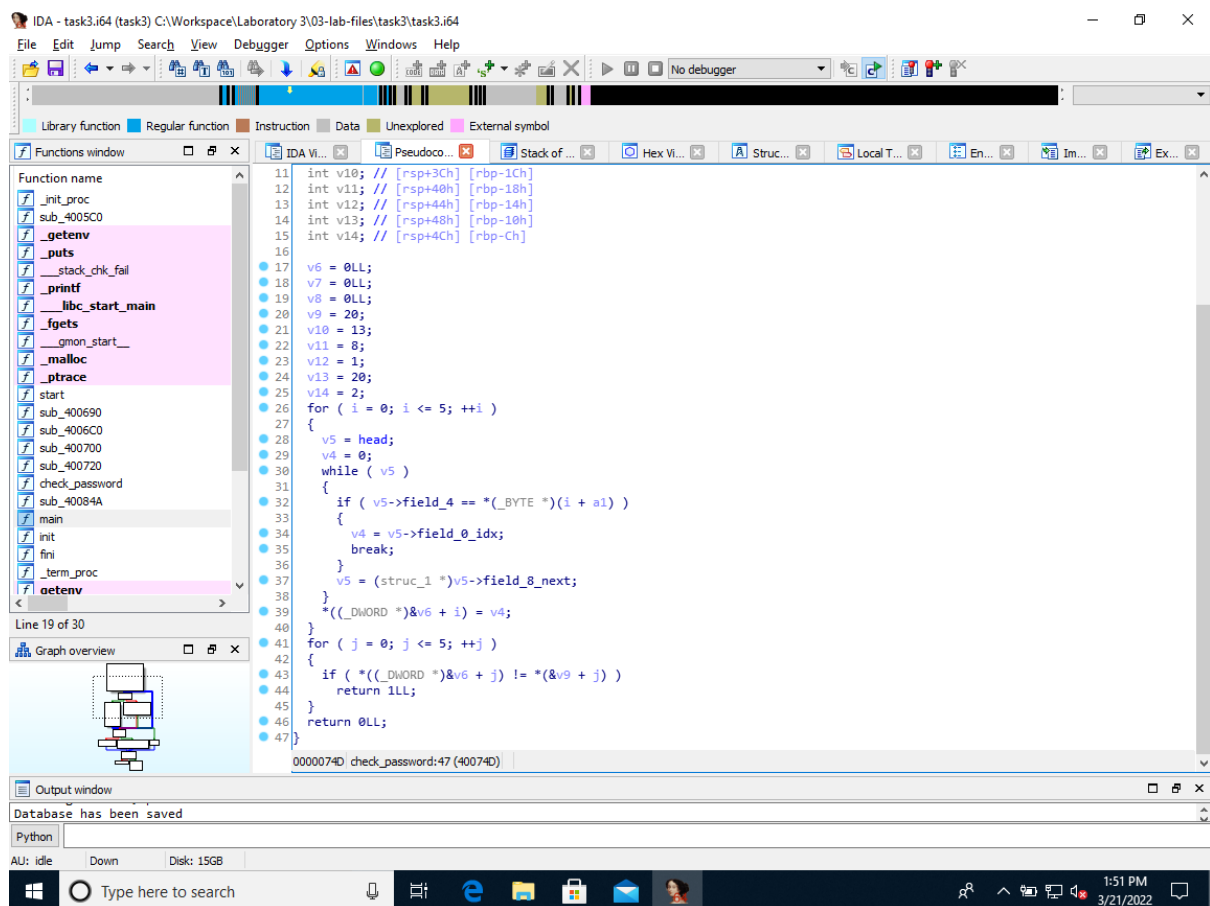
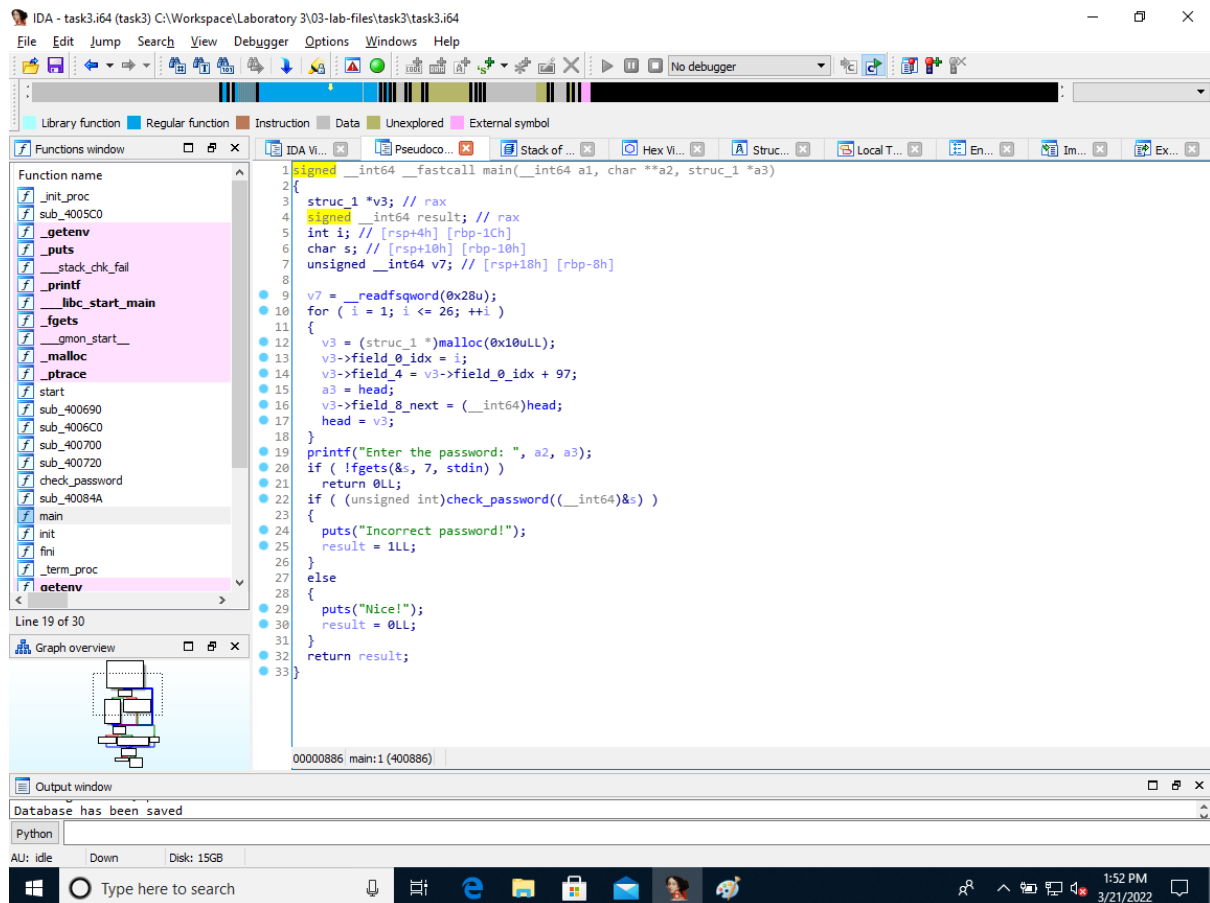
- Use the **Structures** tab and create the following list structure (also declare field_8_next as a **struc_1*** pointer) (2p)

The following screenshot shows the created structure, which can also be seen in the file task31.i64.



- In **main**, cast the buffer returned from `malloc` and the head of the list to this struct type and propagate in the **password checking function**, renaming and retyping where necessary (2p)

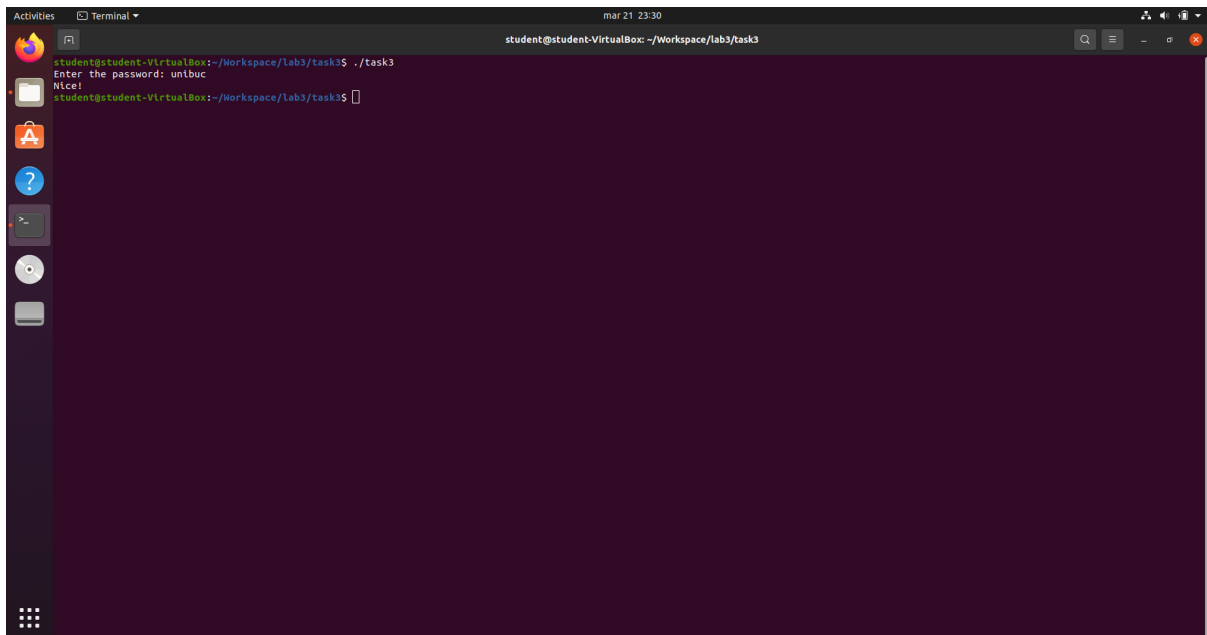
The IDA file containing the renaming and retypings is called task32.i64. The result after the usage of the previously created structure, can also be seen in the following screenshots with the two functions (`main` and `check_password`).



- Describe what the code does and figure out the correct password **(2p)**

In the main function, before reading the output, a single linked list is created having elements connect ASCII characters with indices ((1,b); (2,c); ...; (25,z); (26,{)) (only characters between 98 and 123 are added to the list). The first 6 characters from the input are read and are passed to the password_checking function. Using the list, each character from the input is converted into a number (v4 from the loop). Using these numbers the values from the stack v6, v7 and v8 are set. Afterwards their values are compared with the constants from the stack v9, v10, v11, v12, v13, v14 (their data types are irrelevant since the access is made through pointers).

Now, looking at the initial values from the stack, we have v9=20, v10=13, v11=8, v12=1, v13=20, v14=2. Referring back to the values from the linked list, we obtain the password “unibuc”, which is confirmed by running the program (as in the screenshot below).



```
student@student-VirtualBox: ~/Workspace/lab3/task3
student@student-VirtualBox:~/Workspace/lab3/task3$ ./task3
Enter the password: unibuc
Nice!
student@student-VirtualBox:~/Workspace/lab3/task3$
```