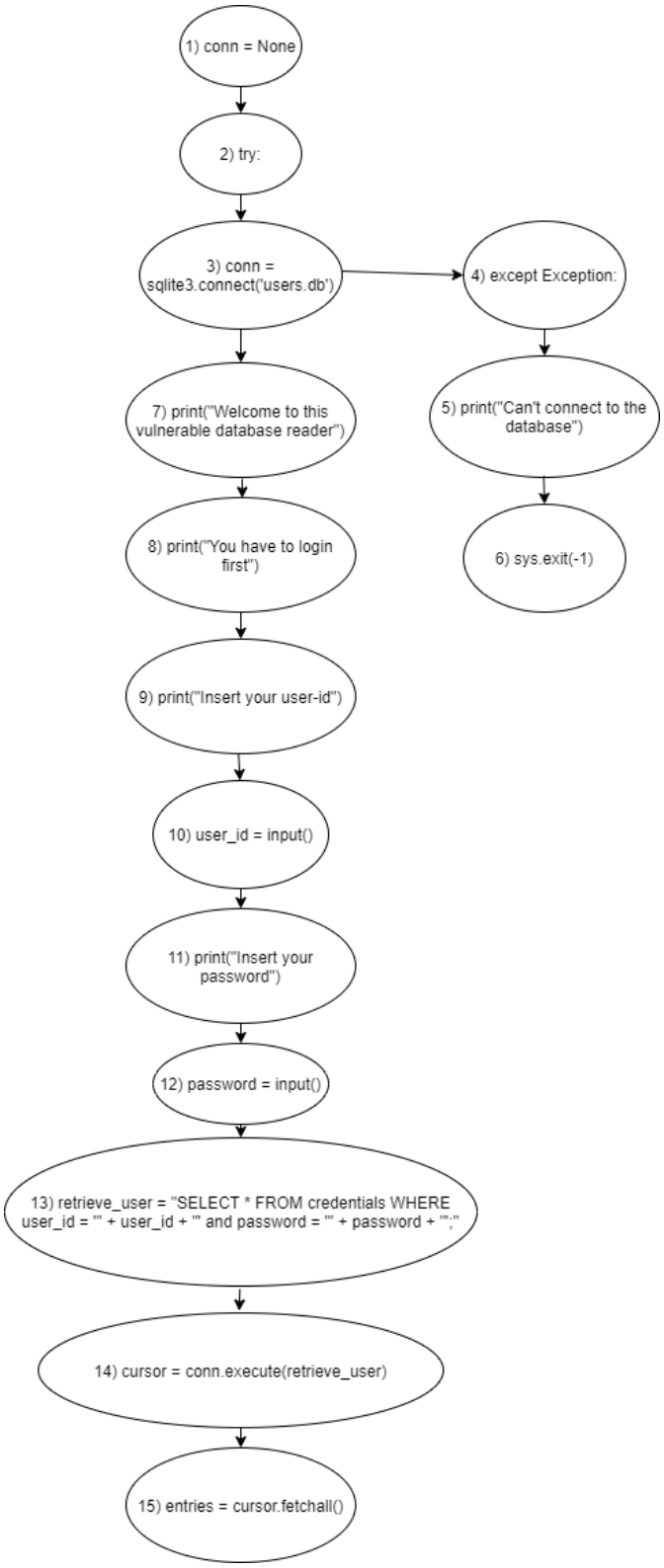
VINCI NICOLO' 220229

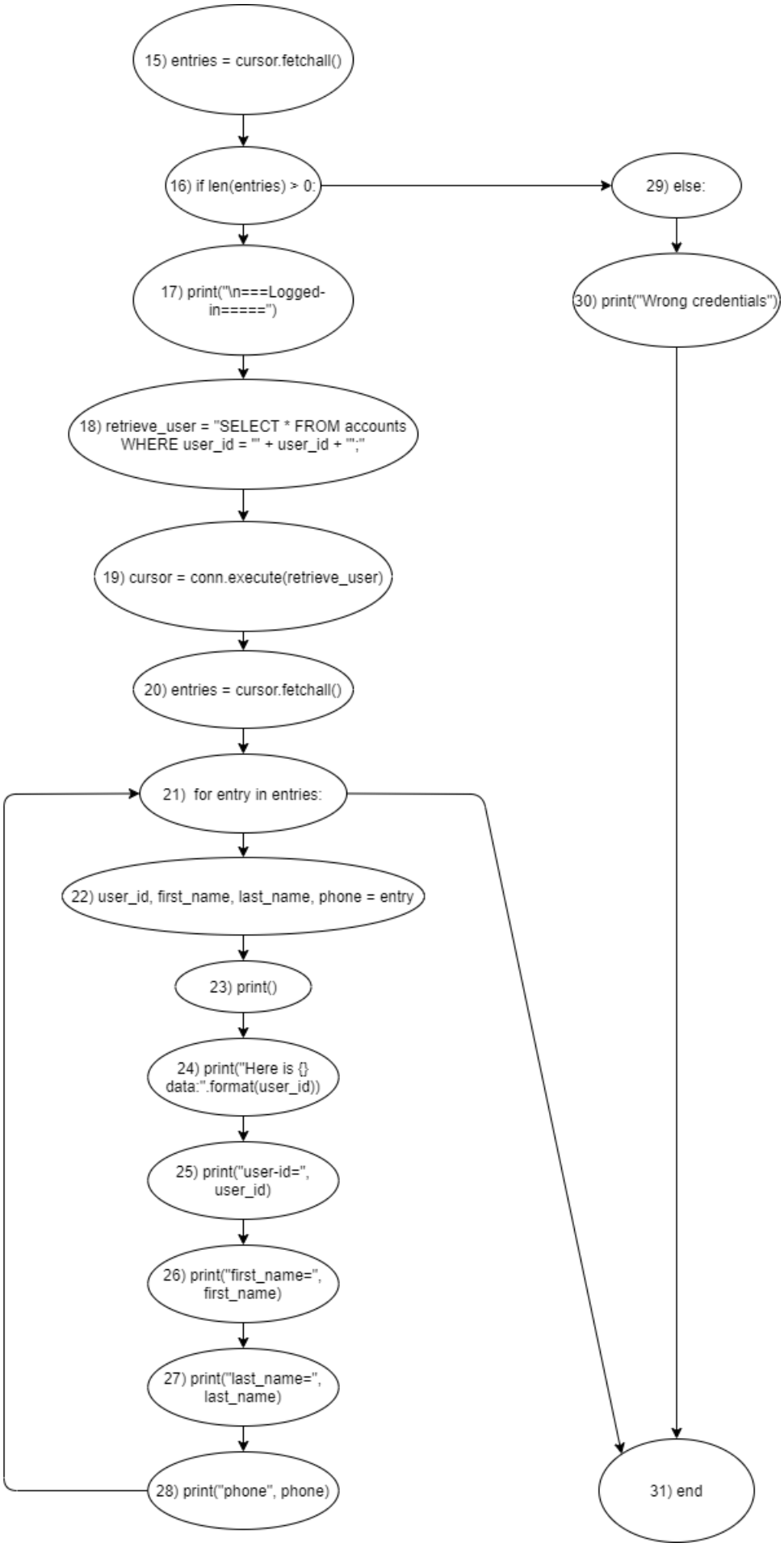
TAINT ANALYSIS

EXERCISE 1: sql-injection

First of all, I do not consider the first three lines for buildin the CFG, where there are only some imports.

STEP 0: draw CFG.




15) entries = cursor.fetchall()
16) if len(entries) > 0:
29) else:
17) print("\n===Logged-in=====")
30) print("Wrong credentials")
18) retrieve_user = "SELECT * FROM accounts WHERE user_id = "' + user_id + '";"
19) cursor = conn.execute(retrieve_user)
20) entries = cursor.fetchall()
21) for entry in entries:
22) user_id, first_name, last_name, phone = entry
23) print()
24) print("Here is {} data:".format(user_id))
25) print("user-id=", user_id)
26) print("first_name=", first_name)
27) print("last_name=", last_name)
28) print("phone", phone)
31) end

STEP 1 and STEP 2: compute GEN and KILL, initialize IN and OUT.

	GEN	KILL	IN	OUT
1	{}	{}	{}	{}
2	{}	{}	{}	{}
3	{}	{}	{}	{}
4	{}	{}	{}	{}
5	{}	{}	{}	{}
6	{}	{}	{}	{}
7	{}	{}	{}	{}
8	{}	{}	{}	{}
9	{}	{}	{}	{}
10	{user_id}	{}	{}	{user_id}
11	{}	{}	{}	{}
12	{password}	{}	{}	{password}
13	{[user_id->T | password->T]retreive_user}	{[user_id->F & password->F]retreive_user}	{}	{[user_id->T | password->T]retreive_user}
14	{[retrieve_user->T]cursor}	{[retrieve_user->F]cursor}	{}	{[retrieve_user->T]cursor}
15	{[cursor->T]entries}	{[cursor->F]entries}	{}	{[cursor->T]entries}
16	{}	{}	{}	{}
17	{}	{}	{}	{}
18	{[user_id->T]retrieve_user}	{[user_id->F]retrieve_user}	{}	{[user_id->T]retrieve_user}
19	{[retrieve_user->T]cursor}	{[retrieve_user->F]cursor}	{}	{[retrieve_user->T]cursor}
20	{[cursor->T]entries}	{[cursor->F]entries}	{}	{[cursor->T]entries}
21	{[entries->T]entry}	{[entries->F]entry}	{}	{[entries->T]entry}
22	{[entry->T]user_id, first_name, last_name, phone}	{[entry->F]user_id, first_name, last_name, phone}	{}	{[entry->T]user_id, first_name, last_name, phone}
23	{}	{}	{}	{}
24	{}	{}	{}	{}
25	{}	{}	{}	{}
26	{}	{}	{}	{}
27	{}	{}	{}	{}
28	{}	{}	{}	{}
29	{}	{}	{}	{}
30	{}	{}	{}	{}
31	{}	{}	{}	{}

STEP 3: compute IN and OUT with transfer function.

	GEN	KILL	IN	OUT
1	{}	{}	{}	{}
2	{}	{}	{}	{}
3	{}	{}	{}	{}
4	{}	{}	{}	{}
5	{}	{}	{}	{}
6	{}	{}	{}	{}
7	{}	{}	{}	{}
8	{}	{}	{}	{}
9	{}	{}	{}	{}
10	{user_id}	{}	{}	{user_id}
11	{}	{}	{user_id}	{user_id}
12	{password}	{}	{user_id}	{user_id, password}
13	{[user_id->T | password->T]retreive_user}	{[user_id->F & password->F]retreive_user}	{user_id, password}	{user_id, password, retrieve_user}
14	{[retrieve_user->T]cursor}	{[retrieve_user->F]cursor}	{user_id, password, retrieve_user}	{user_id, password, retrieve_user, cursor}
15	{[cursor->T]entries}	{[cursor->F]entries}	{user_id, password, retrieve_user, cursor}	{user_id, password, retrieve_user, cursor, entries}
16	{}	{}	{user_id, password, retrieve_user, cursor, entries}	{user_id, password, retrieve_user, cursor, entries}
17	{}	{}	{user_id, password, retrieve_user, cursor, entries}	{user_id, password, retrieve_user, cursor, entries}
18	{[user_id->T]retrieve_user}	{[user_id->F]retrieve_user}	{user_id, password, retrieve_user, cursor, entries}	{user_id, password, retrieve_user, cursor, entries}
19	{[retrieve_user->T]cursor}	{[retrieve_user->F]cursor}	{user_id, password, retrieve_user, cursor, entries}	{user_id, password, retrieve_user, cursor, entries}
20	{[cursor->T]entries}	{[cursor->F]entries}	{user_id, password, retrieve_user, cursor, entries}	{user_id, password, retrieve_user, cursor, entries}
21	{[entries->T]entry}	{[entries->F]entry}	{user_id, password, retrieve_user, cursor, entries}	{user_id, password, retrieve_user, cursor, entries, entry}

22	{[entry->T]user_id, first_name, last_name, phone}	{[entry->F]user_id, first_name, last_name, phone}	{user_id, password, retrieve_user, cursor, entries, entry}	{ user_id, password, retrieve_user, cursor, entries, entry, first_name, last_name, phone }
23	{}	{}	{ user_id, password, retrieve_user, cursor, entries, entry, first_name, last_name, phone }	{ user_id, password, retrieve_user, cursor, entries, entry, first_name, last_name, phone }
24	{}	{}	{ user_id, password, retrieve_user, cursor, entries, entry, first_name, last_name, phone }	{ user_id, password, retrieve_user, cursor, entries, entry, first_name, last_name, phone }
25	{}	{}	{ user_id, password, retrieve_user, cursor, entries, entry, first_name, last_name, phone }	{ user_id, password, retrieve_user, cursor, entries, entry, first_name, last_name, phone }
26	{}	{}	{ user_id, password, retrieve_user, cursor, entries, entry, first_name, last_name, phone }	{ user_id, password, retrieve_user, cursor, entries, entry, first_name, last_name, phone }

27	{}	{}	{ user_id, password, retrieve_user, cursor, entries, entry, first_name, last_name, phone }	{ user_id, password, retrieve_user, cursor, entries, entry, first_name, last_name, phone }
28	{}	{}	{ user_id, password, retrieve_user, cursor, entries, entry, first_name, last_name, phone }	{ user_id, password, retrieve_user, cursor, entries, entry, first_name, last_name, phone }
29	{}	{}	{user_id, password, retrieve_user, cursor, entries}	{user_id, password, retrieve_user, cursor, entries}
30	{}	{}	{user_id, password, retrieve_user, cursor, entries}	{user_id, password, retrieve_user, cursor, entries}
31	{}	{}	{user_id, password, retrieve_user, cursor, entries, entry}	{user_id, password, retrieve_user, cursor, entries, entry}

In line 14 , the program makes a tainted query due to the tainted variable “retrieve_user”, that it is tainted by user input variable “user_id” and “password”.

In line 19, it makes another tainted query due to user input variable “user_id” that is tainted.

In lines 24, 25, 26, 27, 28, it prints some tainted variables due to tainted variable “entry”.

STEP 4: iterates at least once to check the tainted variable.

	GEN	KILL	IN	OUT
1	{}	{}	{}	{}
2	{}	{}	{}	{}
3	{}	{}	{}	{}
4	{}	{}	{}	{}
5	{}	{}	{}	{}
6	{}	{}	{}	{}
7	{}	{}	{}	{}
8	{}	{}	{}	{}
9	{}	{}	{}	{}
10	{user_id}	{}	{}	{user_id}
11	{}	{}	{user_id}	{user_id}
12	{password}	{}	{user_id}	{user_id, password}
13	{[user_id->T | password->T]retreive_user}	{[user_id->F & password->F]retreive_user}	{user_id, password}	{user_id, password, retrieve_user}
14	{[retrieve_user->T]cursor}	{[retrieve_user->F]cursor}	{user_id, password, retrieve_user}	{user_id, password, retrieve_user, cursor}
15	{[cursor->T]entries}	{[cursor->F]entries}	{user_id, password, retrieve_user, cursor}	{user_id, password, retrieve_user, cursor, entries}
16	{}	{}	{user_id, password, retrieve_user, cursor, entries}	{user_id, password, retrieve_user, cursor, entries}
17	{}	{}	{user_id, password, retrieve_user, cursor, entries}	{user_id, password, retrieve_user, cursor, entries}
18	{[user_id->T]retrieve_user}	{[user_id->F]retrieve_user}	{user_id, password, retrieve_user, cursor, entries}	{user_id, password, retrieve_user, cursor, entries}
19	{[retrieve_user->T]cursor}	{[retrieve_user->F]cursor}	{user_id, password, retrieve_user, cursor, entries}	{user_id, password, retrieve_user, cursor, entries}
20	{[cursor->T]entries}	{[cursor->F]entries}	{user_id, password, retrieve_user, cursor, entries}	{user_id, password, retrieve_user, cursor, entries}

21	{[entries->T]entry}	{[entries->F]entry}	{
			user_id,
			password,
			retrieve_user,
			cursor,
			entries, entry,
			first_name,
			last_name,
			phone
			}
				{
				user_id,
				password,
				retrieve_user,
				cursor,
				entries, entry,
				first_name,
				last_name, phone
				}

22	{[entry->T]user_id,
	first_name, last_name,
	phone}
		{[entry->F]user_id,
		first_name, last_name,
		phone}
			{
			user_id,
			password,
			retrieve_user,
			cursor,
			entries, entry,
			first_name,
			last_name,
			phone
			}
				{
				user_id,
				password,
				retrieve_user,
				cursor,
				entries, entry,
				first_name,
				last_name, phone
				}

23	{}	{}	{
			user_id,
			password,
			retrieve_user,
			cursor,
			entries, entry,
			first_name,
			last_name,
			phone
			}
				{
				user_id,
				password,
				retrieve_user,
				cursor,
				entries, entry,
				first_name,
				last_name, phone
				}

24	{}	{}	{
			user_id,
			password,
			retrieve_user,
			cursor,
			entries, entry,
			first_name,
			last_name,
			phone
			}
				{
				user_id,
				password,
				retrieve_user,
				cursor,
				entries, entry,
				first_name,
				last_name, phone
				}

25	{}	{}	{
			user_id,
			password,
			retrieve_user,
			cursor,
			entries, entry,
			first_name,
			last_name,
			phone
			}
				{
				user_id,
				password,
				retrieve_user,
				cursor,
				entries, entry,
				first_name,
				last_name, phone
				}

26	{}	{}	{
			user_id,
			password,
			retrieve_user,
			cursor,
			entries, entry,
			first_name,
			last_name,
			phone
			}
				{
				user_id,
				password,
				retrieve_user,
				cursor,
				entries, entry,
				first_name,
				last_name, phone
				}

27	{}	{}	{
			user_id,
			password,
			retrieve_user,
			cursor,
			entries, entry,
			first_name,
			last_name,
			phone
			}
				{
				user_id,
				password,
				retrieve_user,
				cursor,
				entries, entry,
				first_name,
				last_name, phone
				}

28	{}	{}	{
			user_id,
			password,
			retrieve_user,
			cursor,
			entries, entry,
			first_name,
			last_name,
			phone
			}
				{
				user_id,
				password,
				retrieve_user,
				cursor,
				entries, entry,
				first_name,
				last_name, phone
				}

29	{}	{}	{user_id, password, retrieve_user, cursor, entries}	{user_id, password, retrieve_user, cursor, entries}
30	{}	{}	{user_id, password, retrieve_user, cursor, entries}	{user_id, password, retrieve_user, cursor, entries}

31	{}	{}	{
			user_id,
			password,
			retrieve_user,
			cursor,
			entries, entry,
			first_name,
			last_name,
			phone
			}
				{
				user_id,
				password,
				retrieve_user,
				cursor,
				entries, entry,
				first_name,
				last_name, phone
				}

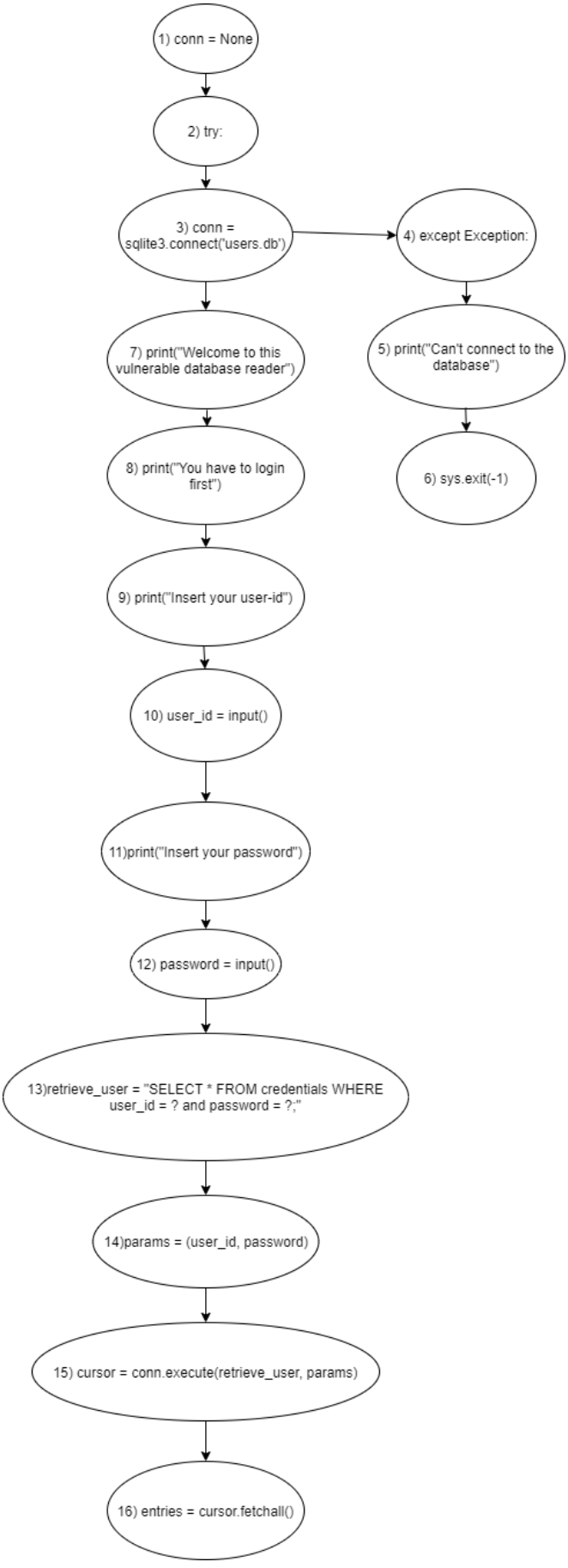
The node 21 is the only changed. It gets in input the output of node 20 and the output of node 28, the latter has some tainted variables in output now. Instead, in step 3 the output of node 28 comes to step 2 (initialization) that was empty.

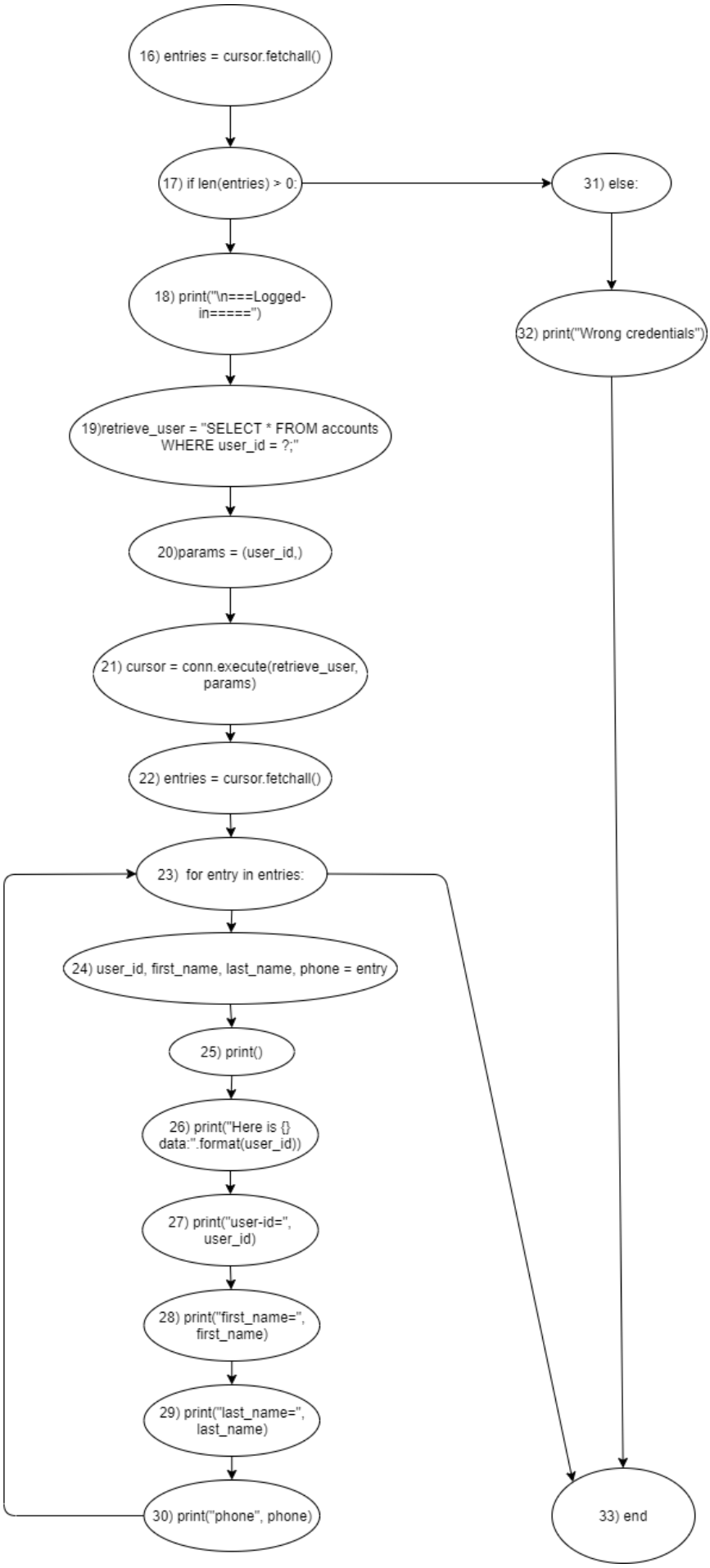
Anyway, the program makes two tainted queries to database and fetches the results, without sanitization of the two input variables.

FIXED sql-injection:

I have to check the two input variables to avoid a SQL injection attack. One solution may be to do parameterized query. So, I introduce the variable “params” to intepret the user input as a string so that the user can not concatenate his input with the query, because the query is built at run time binding the original query with parameters.

STEP 0: draw CFG.




16) entries = cursor.fetchall()
17) if len(entries) > 0:
31) else:
18) print("\n===Logged-in=====")
32) print("Wrong credentials")
19)retrieve_user = "SELECT * FROM accounts WHERE user_id = ?;"
20)params = (user_id,)
21) cursor = conn.execute(retrieve_user, params)
22) entries = cursor.fetchall()
23) for entry in entries:
24) user_id, first_name, last_name, phone = entry
25) print()
26) print("Here is {} data:".format(user_id))
27) print("user-id=", user_id)
28) print("first_name=", first_name)
29) print("last_name=", last_name)
30) print("phone", phone)
33) end

STEP 1 and STEP 2: compute GEN and KILL, initialize IN and OUT.

	GEN	KILL	IN	OUT
1	{}	{}	{}	{}
2	{}	{}	{}	{}
3	{}	{}	{}	{}
4	{}	{}	{}	{}
5	{}	{}	{}	{}
6	{}	{}	{}	{}
7	{}	{}	{}	{}
8	{}	{}	{}	{}
9	{}	{}	{}	{}
10	{user_id}	{}	{}	{user_id}
11	{}	{}	{}	{}
12	{password}	{}	{}	{password}
13	{}	{}	{}	{}
14	{[user_id->T | password->T]params}	{[user_id->F & password->F]params}	{}	{[user_id->T | password->T]params}
15	{}	{cursor, params}	{}	{}
16	{[cursor->T]entries}	{[cursor->F]entries}	{}	{[cursor->T]entries}
17	{}	{}	{}	{}
18	{}	{}	{}	{}
19	{}	{}	{}	{}
20	{[user_id->T]params}	{[user_id->F]params}	{}	{[user_id->T]params}
21	{}	{cursor, params}	{}	{}
22	{[cursor->T]entries}	{[cursor->F]entries}	{}	{[cursor->T]entries}
23	{[entries->T]entry}	{[entries->F]entry}	{}	{[entries->T]entry}
24	{[entry->T]user_id, first_name,last_name,phone}	{[entry->F]user_id, first_name,last_name,phone}	{}	{[entry->T]user_id, first_name,last_name,phone}
25	{}	{}	{}	{}
26	{}	{}	{}	{}
27	{}	{}	{}	{}
28	{}	{}	{}	{}
29	{}	{}	{}	{}
30	{}	{}	{}	{}
31	{}	{}	{}	{}
32	{}	{}	{}	{}
33	{}	{}	{}	{}

STEP 3: compute IN and OUT with transfer function.

	GEN	KILL	IN	OUT
1	{}	{}	{}	{}
2	{}	{}	{}	{}
3	{}	{}	{}	{}
4	{}	{}	{}	{}
5	{}	{}	{}	{}
6	{}	{}	{}	{}
7	{}	{}	{}	{}
8	{}	{}	{}	{}
9	{}	{}	{}	{}
10	{user_id}	{}	{}	{user_id}
11	{}	{}	{user_id}	{user_id}
12	{password}	{}	{user_id}	{user_id, password}
13	{}	{}	{user_id, password}	{user_id, password}
14	{[user_id->T | password->T]params}	{[user_id->F & password->F]params}	{user_id, password}	{user_id, password, params}
15	{}	{cursor, params}	{user_id, password, params}	{user_id, password}
16	{[cursor->T]entries}	{[cursor->F]entries}	{user_id, password}	{user_id, password}
17	{}	{}	{user_id, password}	{user_id, password}
18	{}	{}	{user_id, password}	{user_id, password}
19	{}	{}	{user_id, password}	{user_id, password}
20	{[user_id->T]params}	{[user_id->F]params}	{user_id, password}	{user_id, password, params}
21	{}	{cursor, params}	{user_id, password, params}	{user_id, password}
22	{[cursor->T]entries}	{[cursor->F]entries}	{user_id, password}	{user_id, password}
23	{[entries->T]entry}	{[entries->F]entry}	{user_id, password}	{user_id, password}
24	{[entry->T]user_id, first_name,last_name,phone}	{[entry->F]user_id, first_name,last_name,phone}	{user_id, password}	{user_id, password}
25	{}	{}	{user_id, password}	{user_id, password}
26	{}	{}	{user_id, password}	{user_id, password}
27	{}	{}	{user_id, password}	{user_id, password}

28	{}	{}	{user_id, password}	{user_id, password}
29	{}	{}	{user_id, password}	{user_id, password}
30	{}	{}	{user_id, password}	{user_id, password}
31	{}	{}	{user_id, password}	{user_id, password}
32	{}	{}	{user_id, password}	{user_id, password}
33	{}	{}	{user_id, password}	{user_id, password}

Now, the program makes safe queries, because the two queries are built binding the input variables as parameters with the original query (Line 13, 21).

STEP 4: iterates at least once to check the tainted variable.

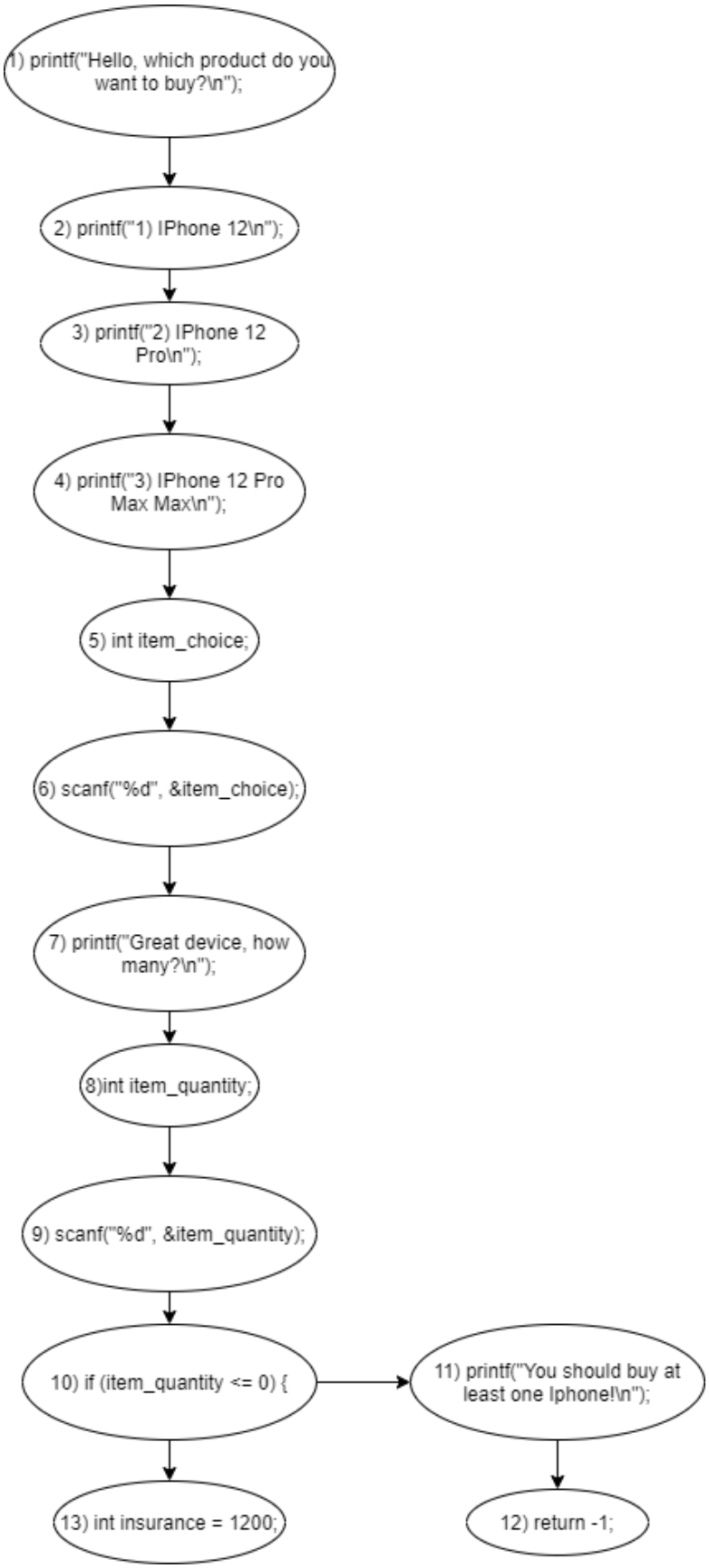
	GEN	KILL	IN	OUT
1	{}	{}	{}	{}
2	{}	{}	{}	{}
3	{}	{}	{}	{}
4	{}	{}	{}	{}
5	{}	{}	{}	{}
6	{}	{}	{}	{}
7	{}	{}	{}	{}
8	{}	{}	{}	{}
9	{}	{}	{}	{}
10	{user_id}	{}	{}	{user_id}
11	{}	{}	{user_id}	{user_id}
12	{password}	{}	{user_id}	{user_id, password}
13	{}	{}	{user_id, password}	{user_id, password}
14	{[user_id->T | password->T]params}	{[user_id->F & password->F]params}	{user_id, password}	{user_id, password, params}
15	{}	{cursor, params}	{user_id, password, params}	{user_id, password}
16	{[cursor->T]entries}	{[cursor->F]entries}	{user_id, password}	{user_id, password}
17	{}	{}	{user_id, password}	{user_id, password}
18	{}	{}	{user_id, password}	{user_id, password}
19	{}	{}	{user_id, password}	{user_id, password}
20	{[user_id->T]params}	{[user_id->F]params}	{user_id, password}	{user_id, password, params}
21	{}	{cursor, params}	{user_id, password, params}	{user_id, password}
22	{[cursor->T]entries}	{[cursor->F]entries}	{user_id, password}	{user_id, password}
23	{[entries->T]entry}	{[entries->F]entry}	{user_id, password}	{user_id, password}
24	{[entry->T]user_id, first_name,last_name,phone}	{[entry->F]user_id, first_name,last_name,phone}	{user_id, password}	{user_id, password}
25	{}	{}	{user_id, password}	{user_id, password}
26	{}	{}	{user_id, password}	{user_id, password}
27	{}	{}	{user_id, password}	{user_id, password}

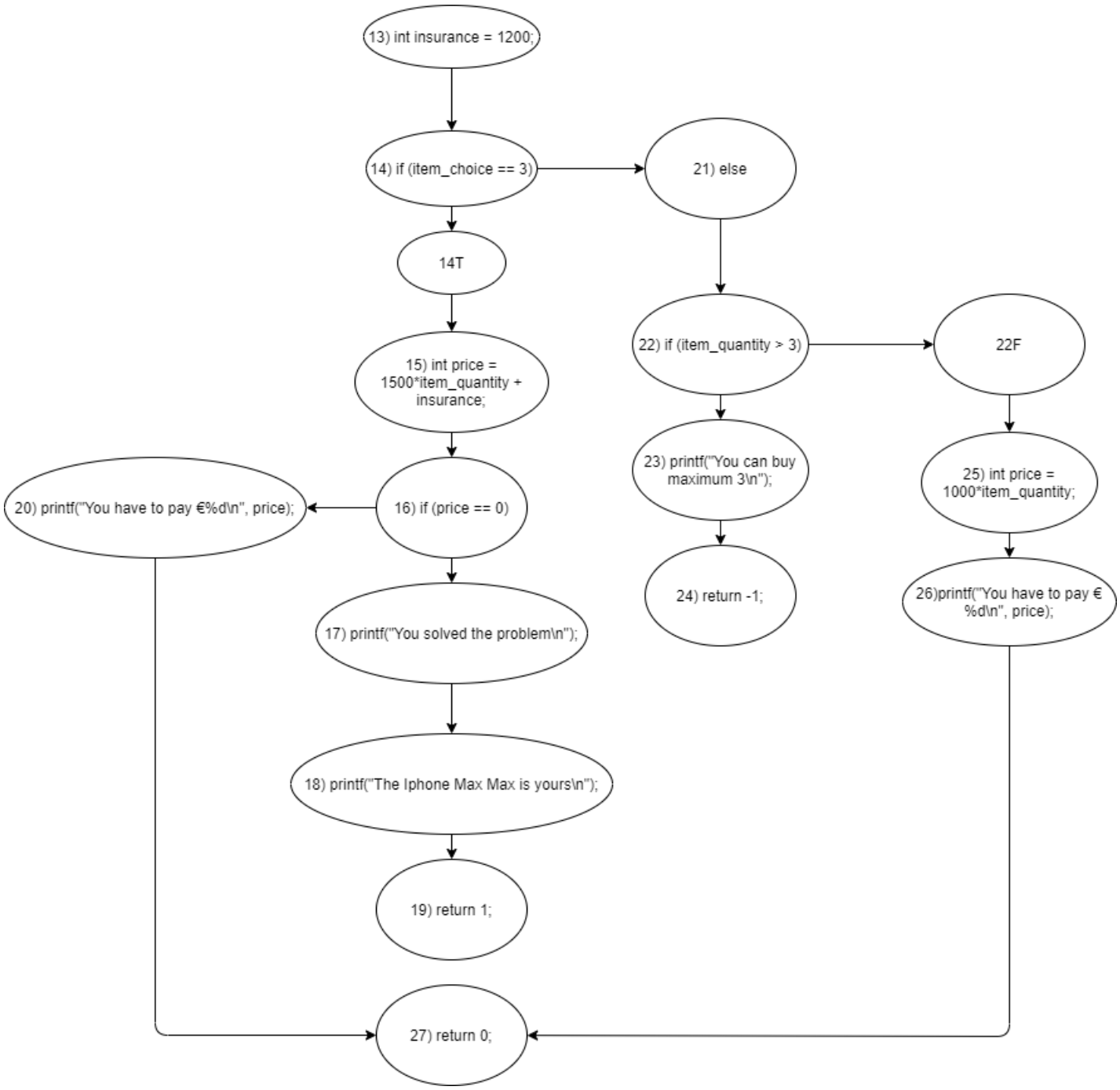
28	{}	{}	{user_id, password}	{user_id, password}
29	{}	{}	{user_id, password}	{user_id, password}
30	{}	{}	{user_id, password}	{user_id, password}
31	{}	{}	{user_id, password}	{user_id, password}
32	{}	{}	{user_id, password}	{user_id, password}
33	{}	{}	{user_id, password}	{user_id, password}

The table remains the same as the step 3.

EXERCISE 2: integer_overflow

STEP 0: draw CFG.




13) int insurance = 1200;
14) if (item_choice == 3)
21) else
14T
22) if (item_quantity > 3)
22F
15) int price = 1500*item_quantity + insurance;
23) printf("You can buy maximum 3\n");
25) int price = 1000*item_quantity;
20) printf("You have to pay €%d\n", price);
16) if (price == 0)
24) return -1;
26)printf("You have to pay € %d\n", price);
17) printf("You solved the problem\n");
18) printf("The Iphone Max Max is yours\n");
19) return 1;
27) return 0;

STEP 1 and STEP 2: compute GEN and KILL, initialize IN and OUT.

	GEN	KILL	IN	OUT
1	{}	{}	{}	{}
2	{}	{}	{}	{}
3	{}	{}	{}	{}
4	{}	{}	{}	{}
5	{}	{}	{}	{}
6	{item_choice}	{}	{}	{item_choice}
7	{}	{}	{}	{}
8	{}	{}	{}	{}
9	{item_quantity}	{}	{}	{item_quantity}
10	{}	{}	{}	{}
11	{}	{}	{}	{}
12	{}	{}	{}	{}
13	{}	{}	{}	{}
14	{}	{}	{}	{}
14T	{}	{item_choice}	{}	{}
15	{[item_quantity->T]price}	{[item_quantity->F]price}	{}	{[item_quantity->T]price}
16	{}	{}	{}	{}
17	{}	{}	{}	{}
18	{}	{}	{}	{}
19	{}	{}	{}	{}
20	{}	{}	{}	{}
21	{}	{}	{}	{}
22	{}	{}	{}	{}
22F	{}	{item_quantity}	{}	{}
23	{}	{}	{}	{}
24	{}	{}	{}	{}
25	{[item_quantity->T]price}	{[item_quantity->F]price}	{}	{[item_quantity->T]price}
26	{}	{}	{}	{}
27	{}	{}	{}	{}

STEP 3: compute IN and OUT with transfer function.

	GEN	KILL	IN	OUT
1	{}	{}	{}	{}
2	{}	{}	{}	{}
3	{}	{}	{}	{}
4	{}	{}	{}	{}
5	{}	{}	{}	{}
6	{item_choice}	{}	{}	{item_choice}
7	{}	{}	{ item_choice }	{ item_choice }
8	{}	{}	{ item_choice }	{ item_choice }
9	{item_quantity}	{}	{ item_choice }	{ item_choice, item_quantity}
10	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
11	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
12	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
13	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
14	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
14T	{}	{item_choice}	{ item_choice, item_quantity}	{ item_quantity}
15	{[item_quantity->T]price}	{[item_quantity->F]price}	{item_quantity}	{ item_quantity, price}
16	{}	{}	{item_quantity, price}	{item_quantity, price}
17	{}	{}	{item_quantity, price}	{item_quantity, price}
18	{}	{}	{item_quantity, price}	{item_quantity, price}
19	{}	{}	{item_quantity, price}	{item_quantity, price}
20	{}	{}	{item_quantity, price}	{item_quantity, price}
21	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
22	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
22F	{}	{item_quantity}	{ item_choice, item_quantity}	{ item_choice}
23	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
24	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
25	{[item_quantity->T]price}	{[item_quantity->F]price}	{ item_choice}	{ item_choice}
26	{}	{}	{ item_choice}	{ item_choice}
27	{}	{}	{ item_choice, item_quantity, price}	{ item_choice, item_quantity, price}

In this first iteration, I can observe from line 15 that the program makes an operation with a tainted input variable “item_quantity” and it could overflow. Moreover, in line 20 the program prints the tainted variable “price” that it is tainted due to the input variable “item_quantity”.

Instead, in the line 25 the program makes an operation with the input variable “item_quantity” that it is untainted due to the check at line 22 and the initial check at line 10.

STEP 4: iterates at least once to check the tainted variable.

	GEN	KILL	IN	OUT
1	{}	{}	{}	{}
2	{}	{}	{}	{}
3	{}	{}	{}	{}
4	{}	{}	{}	{}
5	{}	{}	{}	{}
6	{item_choice}	{}	{}	{item_choice}
7	{}	{}	{ item_choice }	{ item_choice }
8	{}	{}	{ item_choice }	{ item_choice }
9	{item_quantity}	{}	{ item_choice }	{ item_choice, item_quantity}
10	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
11	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
12	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
13	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
14	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
14T	{}	{item_choice}	{ item_choice, item_quantity}	{item_quantity}
15	{[item_quantity->T]price}	{[item_quantity->F]price}	{item_quantity}	{item_quantity, price}
16	{}	{}	{item_quantity, price}	{item_quantity, price}
17	{}	{}	{item_quantity, price}	{item_quantity, price}
18	{}	{}	{item_quantity, price}	{item_quantity, price}
19	{}	{}	{item_quantity, price}	{item_quantity, price}
20	{}	{}	{item_quantity, price}	{item_quantity, price}
21	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
22	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
22F	{}	{item_quantity}	{ item_choice, item_quantity}	{ item_choice}

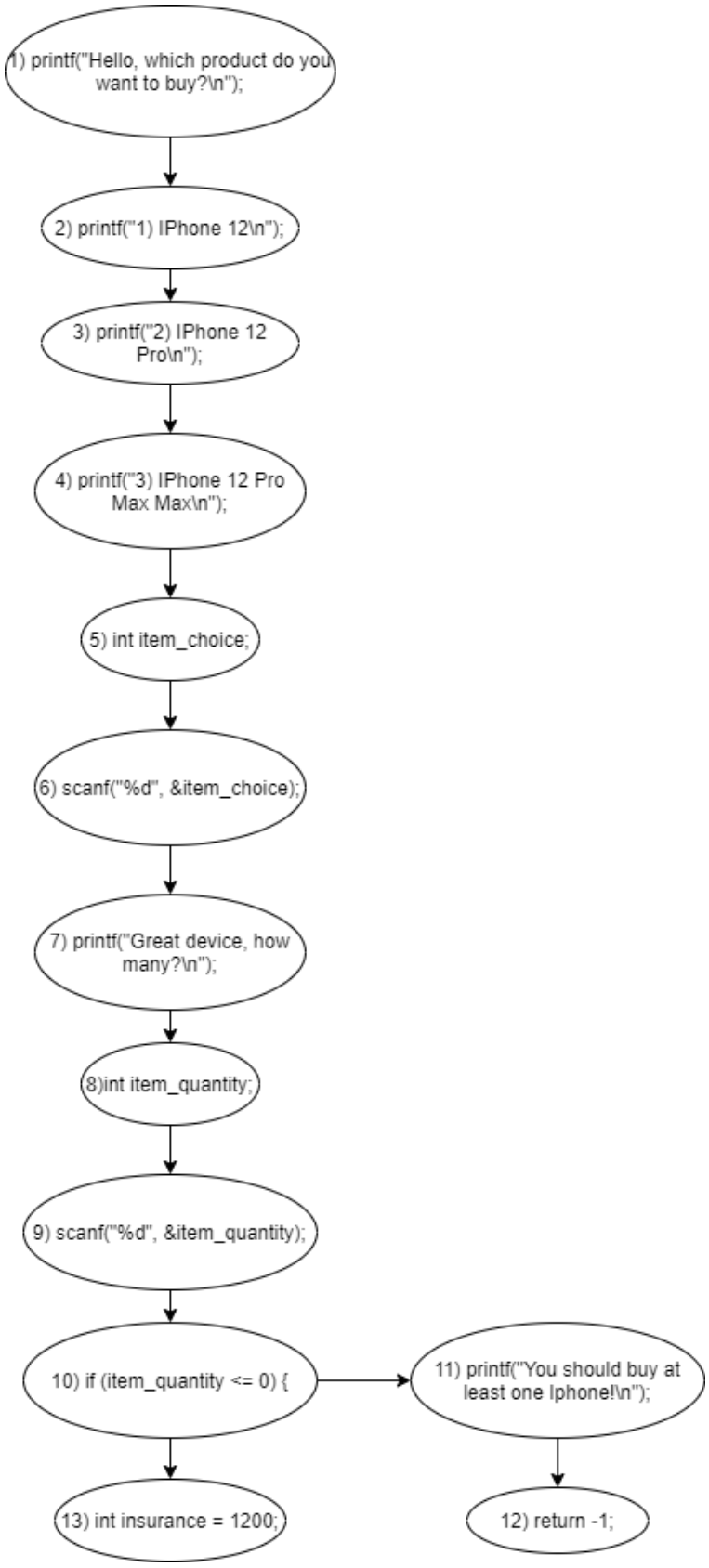
23	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
24	{}	{}	{ item_choice, item_quantity}	{ item_choice, item_quantity}
25	{[item_quantity->T]price}	{[item_quantity->F]price}	{ item_choice}	{ item_choice}
26	{}	{}	{ item_choice}	{ item_choice}
27	{}	{}	{ item_choice, item_quantity, price}	{ item_choice, item_quantity, price}

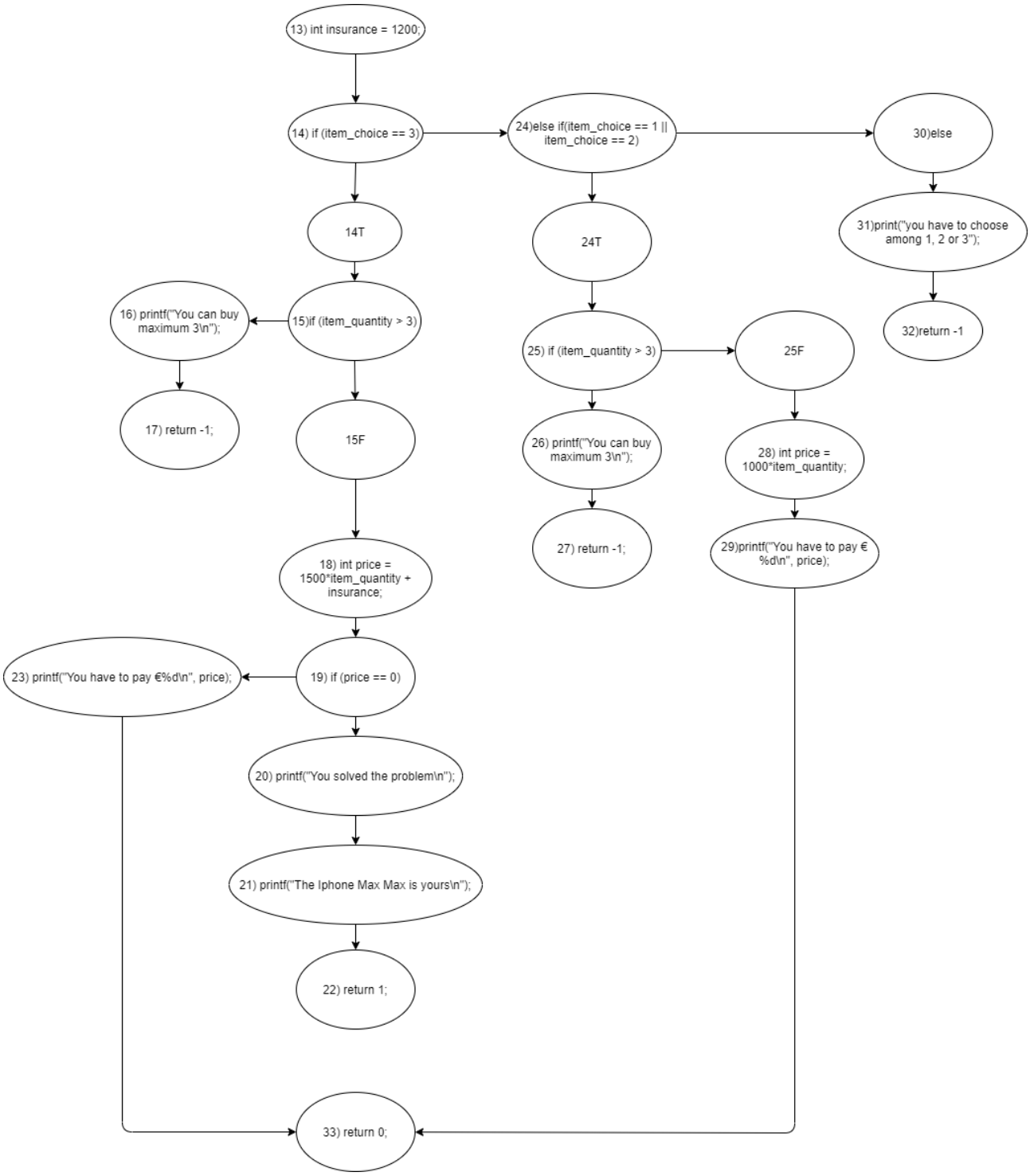
In the step 4 the table remains the same as step 3.

FIXED integer_overflow:

To fix the program, I should check the input variable “item_quantity” before the program makes the operation at line 15 for avoiding the integer overflow. So, I choose that an user can buy at most 3 IPhone 12 Pro Max Max. To sanitize all the script, I should check the input variable “item_choice” so that the user can insert only 1, 2 or 3.

STEP 0: draw CFG.




13) int insurance = 1200;
14) if (item_choice == 3)
24)else if(item_choice == 1 || item_choice == 2)
30)else
14T
24T
31)print("you have to choose among 1, 2 or 3");
16) printf("You can buy maximum 3\n");
15)if (item_quantity > 3)
25) if (item_quantity > 3)
25F
32)return -1
17) return -1;
15F
26) printf("You can buy maximum 3\n");
28) int price = 1000*item_quantity;
27) return -1;
29)printf("You have to pay € %d\n", price);
18) int price = 1500*item_quantity + insurance;
23) printf("You have to pay €%d\n", price);
19) if (price == 0)
20) printf("You solved the problem\n");
21) printf("The Iphone Max Max is yours\n");
22) return 1;
33) return 0;

STEP 1 and STEP 2: compute GEN and KILL, initialize IN and OUT.

	GEN	KILL	IN	OUT
1	{}	{}	{}	{}
2	{}	{}	{}	{}
3	{}	{}	{}	{}
4	{}	{}	{}	{}
5	{}	{}	{}	{}
6	{item_choice}	{}	{}	{item_choice}
7	{}	{}	{}	{}
8	{}	{}	{}	{}
9	{item_quantity}	{}	{}	{item_quantity}
10	{}	{}	{}	{}
11	{}	{}	{}	{}
12	{}	{}	{}	{}
13	{}	{}	{}	{}
14	{}	{}	{}	{}
14T	{}	{item_choice}	{}	{}
15	{}	{}	{}	{}
15F	{}	{item_quantity}	{}	{}
16	{}	{}	{}	{}
17	{}	{}	{}	{}
18	{[item_quantity->T]price}	{[item_quantity->F]price}	{}	{[item_quantity->T]price}
19	{}	{}	{}	{}
20	{}	{}	{}	{}
21	{}	{}	{}	{}
22	{}	{}	{}	{}
23	{}	{}	{}	{}
24	{}	{}	{}	{}
24T	{}	{item_choice}	{}	{}
25	{}	{}	{}	{}
25F	{}	{item_quantity}	{}	{}
26	{}	{}	{}	{}
27	{}	{}	{}	{}
28	{[item_quantity->T]price}	{[item_quantity->F]price}	{}	{[item_quantity->T]price}
29	{}	{}	{}	{}
30	{}	{}	{}	{}
31	{}	{}	{}	{}
32	{}	{}	{}	{}
33	{}	{}	{}	{}

STEP 3: compute IN and OUT with transfer function.

	GEN	KILL	IN	OUT
1	{}	{}	{}	{}
2	{}	{}	{}	{}
3	{}	{}	{}	{}
4	{}	{}	{}	{}
5	{}	{}	{}	{}
6	{item_choice}	{}	{}	{item_choice}
7	{}	{}	{item_choice}	{item_choice}
8	{}	{}	{item_choice}	{item_choice}
9	{item_quantity}	{}	{item_choice}	{item_choice, item_quantity}
10	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}
11	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}
12	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}
13	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}
14	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}
14T	{}	{item_choice}	{item_choice, item_quantity}	{item_quantity}
15	{}	{}	{item_quantity}	{item_quantity}
15F	{}	{item_quantity}	{item_quantity}	{}
16	{}	{}	{item_quantity}	{item_quantity}
17	{}	{}	{item_quantity}	{item_quantity}
18	{[item_quantity->T]price}	{[item_quantity->F]price}	{}	{}
19	{}	{}	{}	{}
20	{}	{}	{}	{}
21	{}	{}	{}	{}
22	{}	{}	{}	{}
23	{}	{}	{}	{}
24	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}
24T	{}	{item_choice}	{item_choice, item_quantity}	{item_quantity}
25	{}	{}	{item_quantity}	{item_quantity}
25F	{}	{item_quantity}	{{item_quantity}	{}
26	{}	{}	{item_quantity}	{item_quantity}
27	{}	{}	{item_quantity}	{item_quantity}
28	{[item_quantity->T]price}	{[item_quantity->F]price}	{}	{}
29	{}	{}	{}	{}
30	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}
31	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}

32	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}
33	{}	{}	{}	{}

Now, as you can see, in line 18 the program makes a mathematical operation with untainted user input “item_quantity”, thanks to the new check at line 15. Moreover, at line 23 the program does not print a tainted variable like before. So, now there is no more risk of integer overflow. Besides, I also check “item_choice”.

STEP 4: iterates at least once to check the tainted variable.

	GEN	KILL	IN	OUT
1	{}	{}	{}	{}
2	{}	{}	{}	{}
3	{}	{}	{}	{}
4	{}	{}	{}	{}
5	{}	{}	{}	{}
6	{item_choice}	{}	{}	{item_choice}
7	{}	{}	{item_choice}	{item_choice}
8	{}	{}	{item_choice}	{item_choice}
9	{item_quantity}	{}	{item_choice}	{item_choice, item_quantity}
10	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}
11	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}
12	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}
13	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}
14	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}
14T	{}	{item_choice}	{item_choice, item_quantity}	{item_quantity}
15	{}	{}	{item_quantity}	{item_quantity}
15F	{}	{item_quantity}	{item_quantity}	{}
16	{}	{}	{item_quantity}	{item_quantity}
17	{}	{}	{item_quantity}	{item_quantity}
18	{[item_quantity->T]price}	{[item_quantity->F]price}	{}	{}
19	{}	{}	{}	{}
20	{}	{}	{}	{}
21	{}	{}	{}	{}
22	{}	{}	{}	{}
23	{}	{}	{}	{}
24	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}
24T	{}	{item_choice}	{item_choice, item_quantity}	{item_quantity}
25	{}	{}	{item_quantity}	{item_quantity}
25F	{}	{item_quantity}	{{item_quantity}	{}
26	{}	{}	{item_quantity}	{item_quantity}
27	{}	{}	{item_quantity}	{item_quantity}
28	{[item_quantity->T]price}	{[item_quantity->F]price}	{}	{}
29	{}	{}	{}	{}
30	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}
31	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}

32	{}	{}	{item_choice, item_quantity}	{item_choice, item_quantity}
33	{}	{}	{}	{}

In the step 4 the table remains the same as step 3.