Memory Check Report

Starting the memory check using valgrind, we decided to check each window by itself to not get overwhelmed.

We started with login and highscore window: valgrind showed leaks that were caused by the creation of widgets that were not dealocated at the end. After some research, we deduced that they should actually be deleted, that is because the parent of both of these windows is the Qwidget class, it has a destructor that deallocates all widgets. So we fixed the problem by calling the destructor after we close the window using “this→close(); this→~login()” for example.

We then worked on the signUp window, since it’s parent is Qwidget, we called the destructor after calling “this→close()”, as we did previously for login and highscoreWindow.

When it came to the game killCovid, we had to call the destructor for all the windows with Qwidget as a parent. But when it came to the game scene, we realized that we were only removing items from the scene instead of calling “delete”. For example: when the syringe would collide with a virus, we removed the virus from the scene but did not call delete.

For Othello, we did not face the issues that we had in KillCovid. That is because all pictures that were created are stored in a Qvector, when the pictures are removed from the scene, we also call vector.erase() that deallocates the erased elements.

In conclusion, the memory check did not show us any invalid write, invalid read, etc… . The only problems we faced were dangling pointers, we fixed those using “delete” and the destructor of each window.

NOTE: Since the main source file in our application is main.cpp, the file creates an instance of Qapplication that gets executed. Qapplication automatically deallocates memory if the user decided to suddenly stop the app by pressing the red X button on the top right.