void LoginWindow::on_loginButton_clicked()

The Big Oh of this function is O(n) since it runs through an SQL query for our login screen. This function first starts by checking the username and passwords in SQL to verify a login attempt, then it runs a loop through the SQL query, and finally progresses on a successful login, or resets on unsuccessful login. This is O(n) because the loop is dependent on the query which acts a 'n'.

CityDistance queryDistance(QString start, QString end)

The Big Oh of this function is O(n) since it runs through an SQL query for our trip distance from start to end. This function takes input for a start and end value, then processes the values through an SQL query in a loop, before returning the calculated city distance. The function runs O(n) because the loop is dependent on the query which had values binded to it using the values passed to the function.

void ResultsWindow::setResults(std::vector<City>& loadedCities)

The Big Oh of this function is O(n^2) since it calculates the shortest distance of the cities and stores them in a table by using a nested for-loop. This function begins with creating the table to store the data and a vector with the cities, then runs through a nested for-loop to assign the data to the table while removing the already visited cities from the vector. The function runs O(n^2) because the nested for-loop at its worst time, runs at 'n' times in each loop.

City ResultsWindow::getClosestCity(std::vector<City> loadedCities, QString name)

The Big Oh of this function is O(n) since it linearly runs through a vector via a loop and returns the value matching the passed name. This function takes a vector of cities and a string for a name as inputs, which it then plugs into a for-loop which runs until the name of the city in the vector matches the passed in name. The function runs O(n) because the for-loop at its worst performance will have to go through the entire list of passed cities before finding the one that matches the name.

```
void CitySelectWindow::loadSelectedCities()
```

The Big Oh of this function is O(n^2) since it runs through a single for-loop, then through a nested for-loop. This function runs through a for-loop which erases non-selected cities, then proceeds to a nested for-loop where a vector for foods and distances is initialized for each city with the external loop, then populated with the internal. The function runs O(n^2) because with one for-loop and one nested for-loop, the equation O($n + n^2$) can be simplified to just O(n^2), and each loop runs the length of the cityNames vector.