Candidate number: 197348

Code Comments

Your code is split into multiple modules, which is good. Your header files need to have #include guards. Looking at queue.h, nice layout with the variable names inside your structure and your function prototypes all in a nice column which makes it much easier to read and it looks much more professional. You have got a structure to represent a queue, and it is typedef-ed. In runOneSimulation.h, your variable names inside your result array are uninformative. Good use of typedef there and #define to create your own Boolean. I can see in runSimulations.h you intend to use the GSL random number generator. In queue.c, good error processing for malloc. You're freeing memory when you remove something from the queue. Note that pop is a stack-related term rather than a queue-related term, but no matter. In runOneSimulation.c, you have got makeResultArray, and I understand why you're doing it, but again, these variable names are not terribly helpful. In line 25, if you have got a function called oppositeValue and the comment is "switching light to opposite signal", why not just call the function changeTrafficLights? You output a load of values. That is good. In the runOneSimulation function, Interesting in Lines 86 and 87: firstly, you have got your traffic lights left and right, where you only actually need one of those to tell which one is green, but secondly, you have made them chars, so they take up a small amount of memory, but then you have given them the character values '0' and '1' where you could just give them the numerical value 0 and 1. Line 98 is your main while loop starting. Notice that your first four if statements all start with "if iteration ¿ 500". You could just have a single if around all of rather than duplicating that. In fact, it is easier to say if its rate is less than (or equal to) five hundred because nearly everything happens in every iteration. Now here's a chunk of code which is effectively changing the traffic lights, so let's have a function called changeTrafficLights. And again, you have got these various chunks of code that are replicated for left and right, particularly around the if statement in line 172. So let's have separate functions. You then make sure that the queues are empty, which they should be at the end, but it is good to verify that you free them and then you calculate and return results. Looking at runSimulations.c, good to check that argc is 5. I don't think Line 14 is doing what you think it is doing. Apart from anything else, I could enter the value 'A'. I think your negative values check in line 30, you probably want to not have a zero value either, certainly not for the traffic light periods. You are seeding the random number generator correctly, just the once and then you're running 100 iterations, getting the results and remembering to free the random number generator at the end. So this is a very good effort. I think if you just put some more functions in here and think about the logic around the main loop, this would condense that while loop down and make it much easier to work out what is going on.

Report

There is a full and complete design section, well done. I note the last paragraph of the report, but you definitely do need to validate! You include some experiments with some discussion.

Marks: 85%