***Core Pollutant Functions:***

* *See flowchart for comprehensive overview of nodes and links:* [*https://www.lucidchart.com/invitations/accept/319e57a8-ba2b-4886-af6a-30d6fda83719*](https://www.lucidchart.com/invitations/accept/319e57a8-ba2b-4886-af6a-30d6fda83719)
* *See flowchart for comprehensive overview of LIDs:* [*https://www.lucidchart.com/invitations/accept/d0471daf-39ed-4b14-8826-492996a84f70*](https://www.lucidchart.com/invitations/accept/d0471daf-39ed-4b14-8826-492996a84f70)

***Code Modification Options for Nodes & Links  
Pros & Cons:***

1. Setting removal R in the node following the methods created for link target\_setting
   1. Pros:
      1. Allows SWMM to route pollutants
      2. Allows SWMM to do other pollutant accumulations/mass balance
      3. No timestep issues
      4. Do not need setter for pollutant concentration
      5. Bypasses SWMM’s mathexpr\_eval() function which limits the math functions available
   2. Cons:
      1. Need to make R a global variable for both nodes and links
      2. getRemoval() called by treatmnt\_treat() checks if R between 0 and 1, so either need to write a different function for erosion or rewrite this function
2. Pollutant setter using the global variable newQual
   1. Pros:
      1. simple function to write
      2. can directly set pollutant concentration if you know it from treatment or sensor data
   2. Cons:
      1. Would need to build our own mass balance because will bypass SWMM’s internal mass balance
      2. Requires creating new global setter variables for both links and nodes
3. Bypass treatmnt\_treat() with an externally defined treatment function
   1. Pros:
      1. Uses global variables newQual
      2. Works for adding more pollutants to the network (like for erosion)
   2. Cons:
      1. It calls an error reporting function and mass balance equation, so we would need to make sure we still call that in our version
      2. Requires creating a new global variable for custom treatment for both links and nodes
4. Pollutant setter buts creates new global function externalQual.
   1. Pros:
      1. Simple to write – reuse existing code but switch newQual to externalQual
      2. allow you to directly set pollutant concentration if you know it (but could add a separate setter for that if that’s someone’s goal, for example, if they have pollutant data already for their network)
      3. Works for adding more pollutants to the network (like for erosion)
      4. allows for SWMM’s internal mass balance to occur
      5. Can replicate this method for LIDs
   2. Cons:
      1. Requires creating two new global variables for both links and nodes

***SWMM Code Modifications for Nodes & Links:***

* ~~src/objects.h:~~
  + ~~Lines 514-517: added the objects <int externalTreatment> , <double\* externalQual>, <double\* C\_in>, <double\* C\_2> for nodes~~
  + ~~Lines 675-676: added the objects <int externalTreatment> and <double\* C\_1> for links~~
* ~~src/toolkitAPI.c:~~
  + ~~Lines 950-981: added swmm\_getNodeC2() to get C\_2~~
  + ~~Lines 983-1014: added swmm\_getNodeCin() to get Cin~~
  + ~~Lines 1016-1046: added swmm\_setNodePollutant() to set externalQual and set externalTreatment == 1~~
  + ~~Lines 1016-1042: added swmm\_getNodeHRT() to get storage node HRT~~
  + ~~Lines 1048-1078: added swmm\_getLinkC2() to get C\_2~~
  + ~~Lines 1081-1111: added swmm\_setLinkPollutant() to set externalQual and set externalTreatment == 1~~
* ~~src/treatmnt.c:~~
  + ~~Line 290-343: added new function < void treatmnt\_custom(int j, double q, double v, double tStep) >~~
* ~~src/project.c:~~
  + ~~Lines 1041-1043: added memory for externalQual, C\_in, and C\_2 for nodes~~
  + ~~Lines 1054-1056: added memory for externalQual and C\_2 for links~~
* ~~src/funcs.h:~~
  + ~~Line 238: added < void treatmnt\_custom(int node, double q, double v, double tStep); >~~
* ~~src/qualrout.c:~~
  + ~~Line 119: added if statement so treatmnt\_setInflow is called if externalTreatment == 1~~
  + ~~Line 133-140: added < else { treatmnt\_custom(j, qIn, vAvg, tStep); } >~~
  + ~~Line 219-246: added <~~ ~~Node[j].C\_2[p] = Node[j].newQual[p];> and <~~ ~~Node[j].C\_2[p] = 0.0;> so can get C\_2 a a global variable when node is not a storage node~~
  + ~~Line 464: added < Node[j].C\_2[p] = c2; > so can get C\_2 as global variable~~
  + ~~Line 334: added < Link[i].C\_2[p] = c2; > so can get C\_2 as global variable to findLinkQual()~~
  + ~~Lines 343-358 : added if statement in findLinkQual() to set newQual to equal externalQual if   
    externalTreatment ==1 and there is water in the link~~

***~~PySWMM Code Modifications:~~***

* ~~swmm5.py: nodes~~
  + ~~Lines 2051-2062: added function getNodeC2()~~
  + ~~Lines 2064-2075: added function getNodeCin()~~
  + ~~Lines 2077-2089: added function getNodeHRT()~~
  + ~~Lines 2104-2114: added function setNodePollutant()~~
* ~~swmm5.py: links~~
  + ~~Lines 2091-2102: added function getLinkC2()~~
  + ~~Lines 2116-2126: added function setLinkPollutant()~~

***Testing Code:***

* Before I can test, need to download new SWMM C code:
  + cd into SWMM folder
  + type <rm -r CMakeFiles> enter
  + type <rm CMakeCache.txt> enter
  + type <cmake CMakeLists.txt
  + > enter
    - that remakes the Makefile which tells C compiler which files to compile
  + type <make> and enter
    - tells you what errors there are
  + go to lib folder, libswmm5.so is the file will all my SWMM functions
    - copy and paste libswmm5.so into PySWMM macros folders
* Node Tests:
  + Test one tank with constant inflow and constant pollutant load
    - Test the treatments that are available in SWMM: constant effluent concentration, percent removal, and nth order reaction
    - Confirm PySWMM matches SWMM results: concentration (by graphing), mass balance (report file)
    - Confirm getters and setters work
  + Test one tank with variable inflow and variable pollutant load
    - Test the treatments that are available in SWMM: constant effluent concentration, percent removal, and nth order reaction
    - Confirm PySWMM matches SWMM results: concentration (by graphing), mass balance (report file)
    - Confirm getters and setters work
  + Several tanks and only change 2 tanks with variable inflow and variable pollutant load
    - Test the treatments that are available in SWMM: constant effluent concentration, percent removal, and nth order reaction
    - Confirm PySWMM matches SWMM results: concentration (by graphing), mass balance (report file)
    - Confirm getters and setters work
* Link Tests:
  + Used SWWM example input file for testing
  + Cannot do treatment in links, so I cannot test like testing nodes, instead can only check mass balance
  + Tested the following scenarios:
    - Constant inflow and constant pollutant load:
      * setting constant concentration in link
      * use getter to get link concentration run calculation and set new concentration (both smaller (percent removal treatment) and larger (erosion) concentrations)
      * use getter to get link concentration run calculation and set new concentration and then run treatment at downstream node (had to change junction to storage node in order to run node treatment)
    - Variable inflow and variable pollutant load:
      * setting constant concentration in link
      * use getter to get link concentration run calculation and set new concentration (both smaller (percent removal treatment) and larger (erosion) concentrations)
      * use getter to get link concentration run calculation and set new concentration and then run treatment at downstream node (had to change junction to storage node in order to run node treatment)

**TO-DO LIST:**

**PySWMM:**

1. add getters and setters for pollutants in swmm.py – follow all the notions of PySWMM’s code
2. add unit tests
3. add getters and setters for pollutants in node and link classes

**SWMM:**

1. clean code with getters and setters – follow all the notions of SWMM’s code
2. add units tests
3. ensure we do not break any prior functionality
4. SWMM has multiple branches, which branch to put pull request on? – ask Bryant in raised issue
   1. Add changes directly into KLabUM code
   2. Will push KLabUM code to SWMM

**Toolbox:**

1. Finish building out all my functionality – this will confirm we have everything we need in SWMM/PySWMM
2. Need to come up with a fancy name for toolbox: needs to make sense, make sure library name is available (https://pypi.org/)
3. what do I want my user to do? (basic users vs users who want to make customize equations, etc.)
   1. default functionality
   2. custom functionality