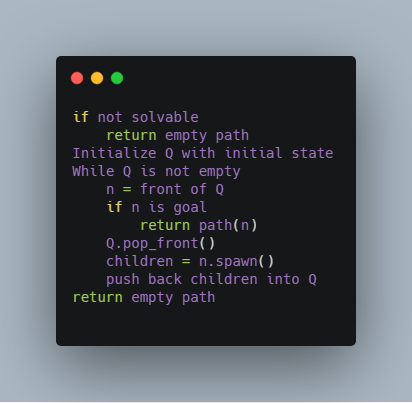
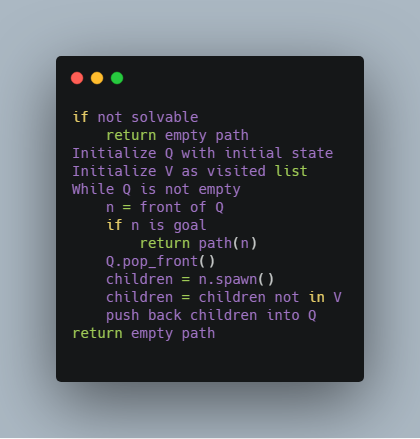
1. BFS without visited list
   1. State representation: std::string
   2. Q: std::deque<Node> ; Since deque can be worked on both beginning and end, it can be used as a Queue data structure.
   3. Expanded List: NA.
   4. Pseudo code



1.5 Extra work: NA

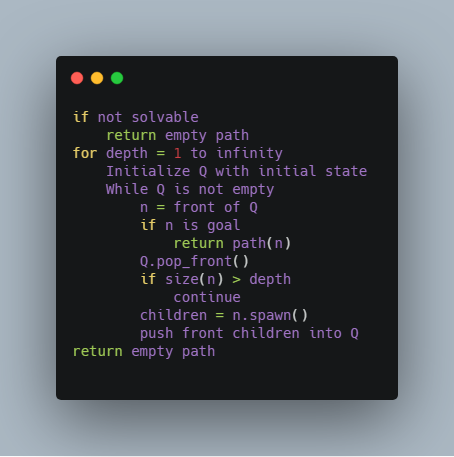
1. BFS with visited list
   1. State representation: std::string
   2. Q: std::deque<Node> ; Since deque can be worked on both beginning and end, it can be used as a Queue data structure.
   3. Expanded List: NA.
   4. Pseudo code



2.5.1 Original Hash Function

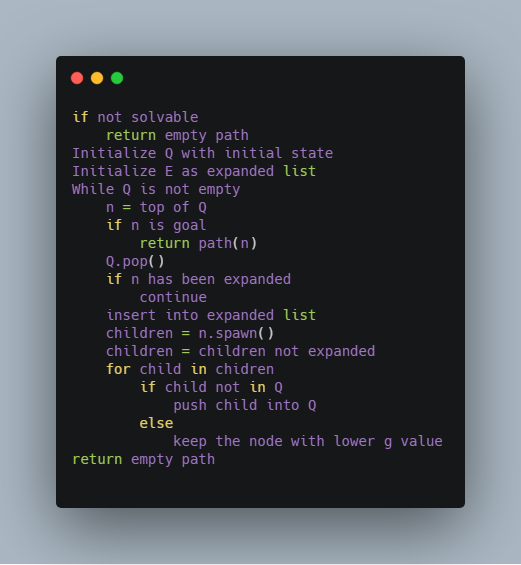
Original Hash Function with std::unordered\_set. Formula: hash\_value = std::atoi(state as string). Since this hash function simply parse state strings to int, such formula guarantee that no duplicates would be generated.

1. PDS without visited list
   1. State representation: std::string
   2. Q: std::deque<Node> ; Since deque can be worked on both beginning and end, it can be used as a Queue data structure.
   3. Expanded List: NA.
   4. Pseudo code



* 1. Extra work: NA

1. UC with expanded list
   1. State representation: std::string
   2. Q: PriorityQueue<Node>: original priority queue that allows access and delete elements inside and also keep all properties during such operations.
   3. Expanded List: std::unordered\_set<string, Hash>: std hash set with original hash function, so that search will take around O(1).
   4. Pseudo code



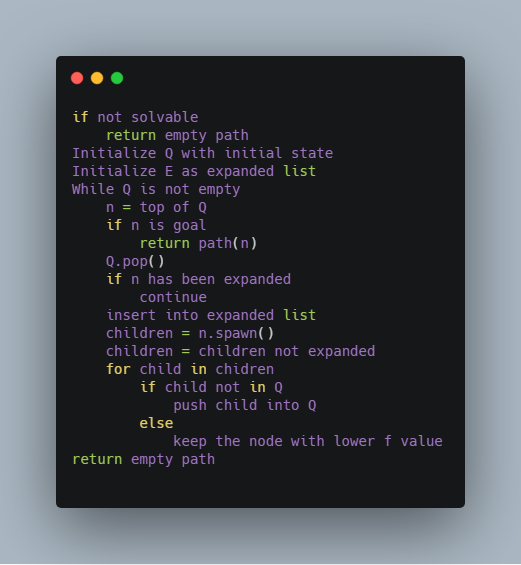
4.5.1 Original Hash Function

Original Hash Function with std::unordered\_set. Formula: hash\_value = std::atoi(state as string). Since this hash function simply parse state strings to int, such formula guarantee that no duplicates would be generated.

4.5.2 Original Heap

The original heap/priority queue is implemented by wrapping std::vector. By exposing the underlying std::vector, external code is able to access each element. Thus after locating the element by linear search, external code can call the public method to delete it. This delete method will delete it by swapping it with the last element in the heap and then call the heapify method to run through the target element in a top down way to maintain the heap property.

1. A star with expanded list
   1. State representation: std::string
   2. Q: PriorityQueue<Node>: original priority queue that allows access and delete elements inside and also keep all properties during such operations.
   3. Expanded List: std::unordered\_set<string, Hash>: std hash set with original hash function, so that search will take around O(1).
   4. Pseudo code



5.5.1 Original Hash Function

Same as part 4.5.1

5.5.2 Original Heap

Same as part 4.5.2