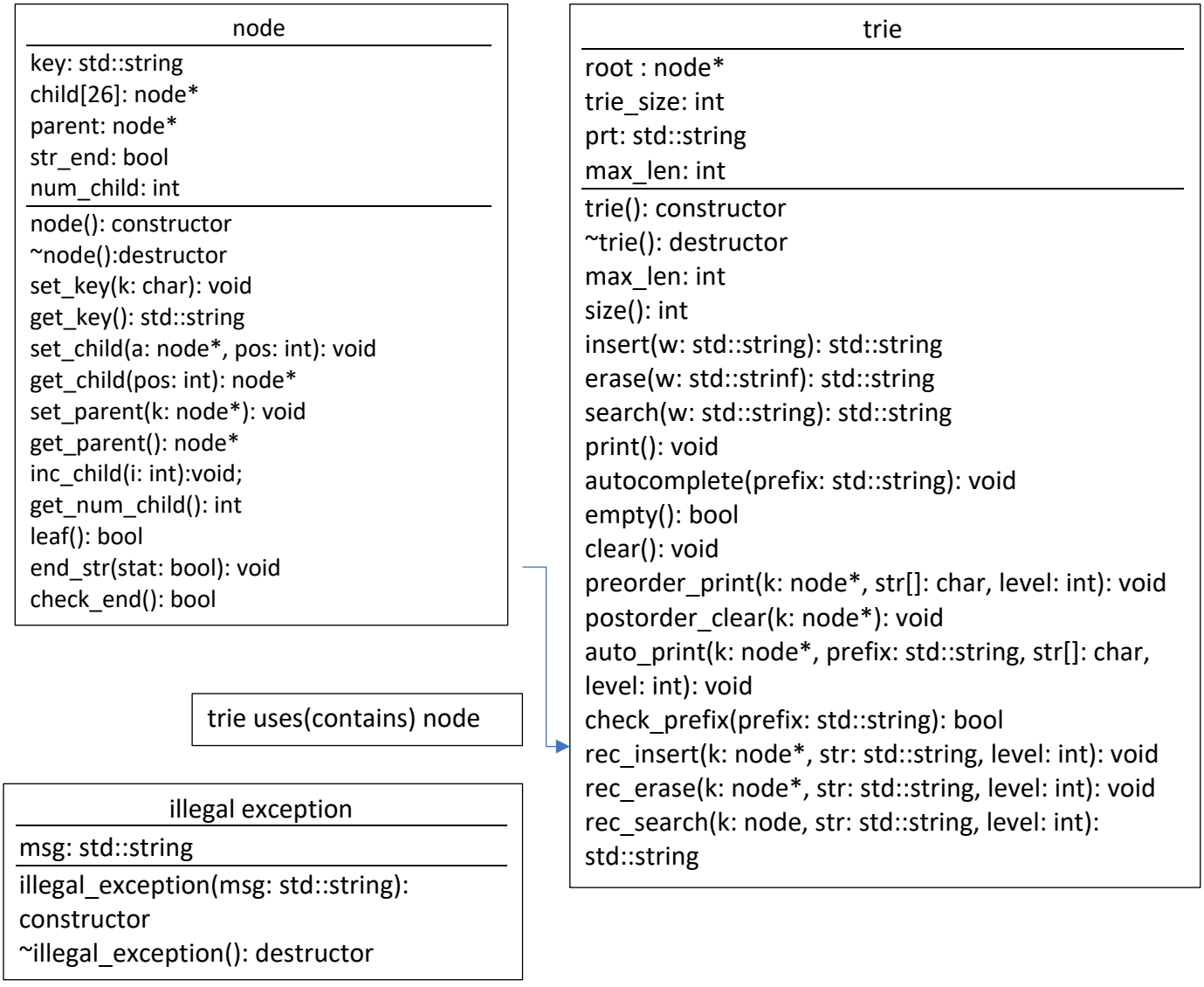
Project 2

Ji Wu

I designed three classes for this project: 1. node 2. trie 3. illegal_exception
The trie class will form a trie(tree-like structure) containing multiple node structure in it. Each node structure holds the key value that can be insert to the trie.

- The node class represent each node of the trie. A node holds a key value and its child and parent pointers so that multiple nodes can be connected together. The child pointer is a pointer array of size 26 so that each node can have up to 26 children. A node also contains a bool variable "str_end" to indicate if the node represents the end of a word and a int variable "num_child" to count the number of children of the node. Member functions can be used to modify and return the value of each private member.
- The trie class represent the body of the trie. It has a root pointer to represent the root of the trie. It contains a int variable "trie_size" to indicate the number of words inserted into the trie. A string variable will be used for printing. The member functions can manipulate the trie including insert, erase, search, print, autocomplete, empty, and clear. Some recursive functions that will be used to implement the above functions are also declared in the class.
- The illegal exception class is used to throw exceptions when the input parameter for search, insert and erase is not 26 lower case alphabets.



Detail on design decision:
node:

- Constructor: initialize the node value, key will be set to “empty” and can be modify the later using set_key function. All entries of child pointer and parent pointer will be set to nullptr. num_child will be set to 0 and str_end to false.
- Destructor will be used to deallocate memory if needed, in this case, no memory needs to be deallocated.
- No operator was overridden.
- Since we do not want the parameter to change during the program, all parameters should be passed using const and reference.

trie:

- Constructor: initialize the private variables. root pointer will allocate memory to be new node. trie_size will be set to 0 and max_len will be set to 0.
- Destructor will be used to deallocate memory if needed. Here root pointer and insertion to the trie request memory, so destructor will call clear() and deallocate the root pointer.
- No operator was overridden.
- The str[] in print and auto_print will need to change during the corresponding function. Other than those two parameters, all other parameters will be passed using const and reference since we do not want then to change during the program.

Test cases:
I tested each function separately and then do comprehensive tests on them. I tested how my program react to illegal arguments.

- Consider input for insertion, erase, search in format “str1 str2”, blank is not lower case alphabet so should make sure this throws illegal argument.
- Consider string “the” and “there” in the trie, when erasing one of them, the other one should still exist in the trie.
- Consider string “there” in the trie but “the” is not. If we search for “the”, it should return not found although “there” contains “the”.
- Consider “the” and “there” in the trie, when we autocomplete ther*, only thoughts should be printed.
- Repeatedly insert, erase or clear the trie and call empty and size function to see if they return the expected number of the words.

Sample test cases posted on Learn was also used to test the program.

Performance:

- empty() and size() could be done in O(1). We only need to check if root is a leaf node or trie_size is 0 for empty(). For size(), I increase the private variable trie_size every time a word is successfully inserted and decrease or set to 0 when erasing or clearing, only need to return trie_size value.
- Insert, erase and search depend on the size of the target string O(n), because we only need to traverse the same number of times as the string size to insert all alphabet to the trie using recursion.
- Print, autocomplete and clear depend on the size of the trie O(N). Because those function requires printing of the trie. The number of times traverse need to be depend on how big the trie is or how big the trie is on a specific route. Recursion was used for traversal.