University of Waterloo
CS240 - Spring 2023
Programming Question 2

Due Date: Wednesday July 19, 5pm

You should have submitted AID02 before the due date of this assignment. The agreement will indicate what you must do to ensure the integrity of your grade. If you are having difficulties with the assignment, course staff are there to help (provided it isn't last minute).

The Academic Integrity Declaration must be signed and submitted on time or the assessment will not be marked.

Please read `https://student.cs.uwaterloo.ca/~cs240/s23/assignments.phtml#guidelines` for guidelines on submission. Submit the file `trie.cpp` to Marmoset.

Late Policy: Assignments are due at 5:00pm, with the grace period until 11:59pm. Assignments submitted after 11:59pm on the due date will not be accepted but may be reviewed (by request) for feedback purposes only.

In this assignment, you will implement the basic functionalities of tries for strings made of characters a and b. We will use the implementation described in class, using a special character (of your choice) to indicate end-of-word (warning: the input strings we use in our tests do not include this special character; you have to manage it yourselves).

- You have to implement functions to insert, delete (as explained in class) and find strings. For these functions, you can assume that inputs are well-formed (non-empty strings made of a and b only).
  - If we try to insert a string s already in the trie, you should print
    `insert: string (print s) already in the trie`
    (with new line at the end) and continue execution. For instance, if $s = aa$, we want to see
    `insert: string aa already in the trie`
  - If we try to delete a string s not in the trie, you should print
    `delete: string (print s) not in the trie`
    (with new line at the end) and continue execution.
  - For find, if you find the string s in the trie, print
    `find: string (print s) found`
    else print

`find: string (print s) not found`
(with new line at the end) and continue execution.

These functions are defined in the class trie, but for the moment they do nothing.

- You also have to implement a pretty_print function. On input a trie with strings aa, $aaab$, b and ba, it should output the following
––*–aa
| └–*–*–aaab
└–*–b
  └–*–ba Note: we do not print the end-of-word character.

  Here is how the output is recursively defined. If the current node is empty, the output is the empty string. If your node contains a string, the output is this string. Else, call $P_\$$, P_a and P_b be the pretty prints corresponding to children \$, a and b. Then your output will be
*–(first line of $P_\$$)
| (…)
| (last line of $P_\$$)
├–(first line of P_a)
| (…)
| (last line of P_a)
└–(first line of P_b)
  (…)
  (last line of P_b)
  with a slight adjustment: if any one of $P_\$$, P_a or P_b is the empty string, skip it altogether, and do not print it. For instance, if $P_\$$ is the empty string, but P_a and P_b are not, the output is
*–(first line of P_a)
| (…)
| (last line of P_a)
└–(first line of P_b)
  (…)
  (last line of P_b)
  The exact characters you should use (*–├└|) are in the skeleton we provide.

- Finally, you have to implement a partial compression function (which will modify the current trie), where we prune branches that carry only one string. It is defined recursively as follows: at any node, prune recursively children with labels a and b (there is nothing to do for the \$ child). If all three children store in total one single string, the current node will hold this string and all children are deleted.

  After pruning, you should still be able to pretty print the trie, but we will not insert, find or delete in a pruned trie.

After completing the skeleton we give you, rename it and submit it as a file called `trie.cpp`. We give you a mostly empty class for tries, which you have to complete. You can add new classes / structs, but you **cannot** change main (we will auto-mark the assignments). You cannot use any pre-existing class for trees or tries.