ITU Computer Engineering Department
BLG 223E Data Structures, Summer 2021
Homework #10
Due August 19, 2021 23:59

Definition

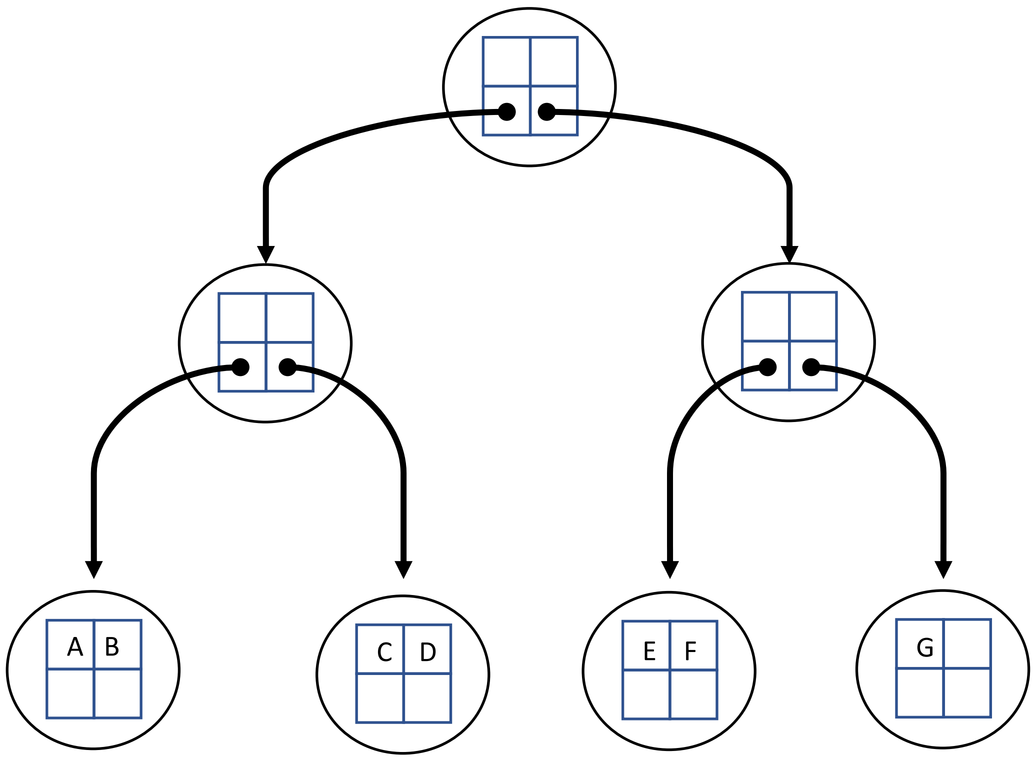
You are asked to implement a balanced radix tree vector (RBVector) that can hold strings. Please pay attention to the following details in your implementations:

- Your RB nodes should follow the following definition. You may add additional private methods/fields when necessary.

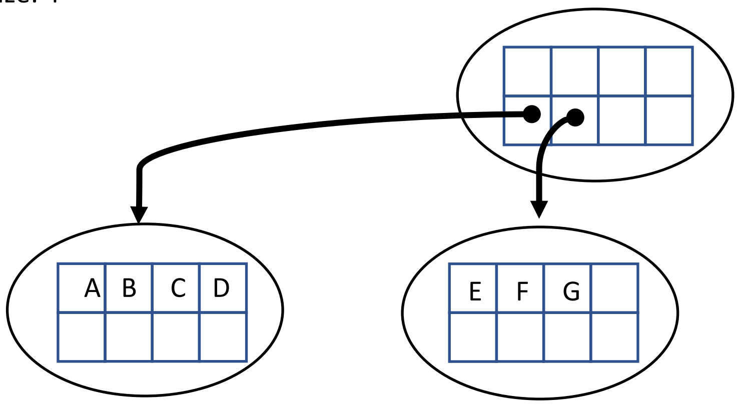
```
class RBNode{
    private:
        int branching_size;
        int cur_size;        //current number of elements in the
data array
        string* data;
        RBNode** next;
     public:
        //Other methods
};
```

- The amount of data that can be present in each node and number of children each node may have is designated at runing time. An example vector of 7 elements with branching sizes of 2, 4 and 8 separately is given below. Empty cells mean NULL pointers or empty strings.

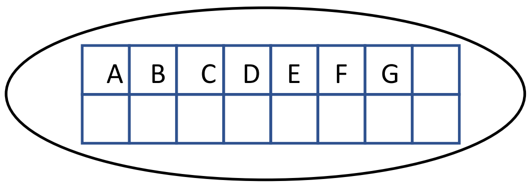
Branching size: 2



Branching size: 4



Branching size: 8



- The branching factor is always going to be a power of two.
- Your vector should support the following functions:
    - Push back (append at the end), update (change a value in an index), remove (remove a string), remove_from(remove the element in an index), find (return index of a string, -1 if not present), clear (reset the tree and free present nodes), print (print the elements in the vector by index order).
- Your tree should always keep the vector elements in the leaves.
- Your tree should add a node when required, but delete the nodes lazily.
    - For instance imagine that you have a branching factor of four and you have four elements in your root. When a new value is added resulting tree should have a root and first two child nodes of the root. The first child node should contain the original values where the second child node contains only the newly added value. In the example diagram with branching size 4 adding H and I to the tree will cause the third child to be constructed not the whole second level.
    - As another example, imagine that you have a branching factor of four and you have sixteen elements in your vector. There is not going to be any node removals until the tree depth might be shrinked, where for our case when there exists four elements in the vector. In the example diagram with branching size 2 removing F and G will not cause any node removals while removing E, F and G will cause the whole right subtree and the root to be removed.
- Your implementation **is not expected to** be immutable and **is not expected to** use a relaxed balanced radix tree.
- Your vector should be case sensitive, might contain duplicates, removing/finding a duplicate element results in removal/finding of the first occurence of the element.
- Removals from the vector should shift all the successors towards the removal index.

- You may check out the following resources but please keep in mind that the data structure in your homework might be quite different than both of them even though conceptually equivalent.
  - Whole speech: https://www.youtube.com/watch?v=sPhpelUfu8Q
  - Whole speech: https://www.youtube.com/watch?v=gx9T5HBYBqo
  - Section 2.1: https://infoscience.epfl.ch/record/213452 The rest of the paper is about relaxed case, which is not expected to be implemented.

Input-Output

Your program should accept a filename as a command line parameter. In the accepted file:

- First line is going to contain a single number followed by the string RBV that gives the branching size for your RB tree.
- Following lines may include one of the following commands:
  - PUSH keyword followed by **exactly** one string that should be added to your RBVector.
  - RM keyword followed by **exactly** one string that should be removed from your RBVector.
  - RMI keyword followed by **exactly** one integer that should be removed from the specific index of your RBVector.
  - UPDATE keyword followed by **exactly** one integer and **exactly** one string that the related value of the indice should be updated with the provided string.
  - FIND keyword followed by **exactly** one string finds the index of the string, return -1 if not present.
  - GNC keyword results in the output of the number of nodes present in your underlying tree.
  - PRINT keywrod results in the output of the elements in your vector printed by their index order.
  - CLEAR lears all the values and nodes from your tree.

Input will not contain syntactical errors. However it may contain removal/finding/updating of non-existent elements and trying to remove from an empty vector. See the provided test cases for examples.

Deliver

Please zip and deliver the directory structure defined below:

- HW10: Topmost folder, that will contain all the folders in your submission. No other files should be present under this folder in your submission.
- HW10/src: Contains all the *.cpp files
- HW10/src/main.cpp: Contains your main function.
- HW10/src/RBNode.cpp: Contains your RBTree's node class definitions.
- HW10/src/RBVector.cpp: Contains your vector class' definitions.
- HW10/include: Contains all the header files you use
- HW10/bin: An empty directory that will contain objective files when your project is compiled

Please check the calico test file in the homework definition to see how your files will be compiled and tested.

Restrictions and Guidelines

- Compilation environment: Only the code that can be compiled by the environment of the container definition provided in ninova will be accepted.
- Testing of your program will be performed using Calico (https://calico.readthedocs.io/en/latest/). Test cases that will be used to test your homework is provided as an attachment in ninova.
- **STL usage is not allowed at all cases.**
- **This homework is for individual submissions**. Any kind of code sharing or code adaptation from an external source is strictly forbidden. Submitted code will undergo a plagiarism check process and plagiarsim penalties may be given if the submitted code's similarity is approved by the instuctor.
- Make sure you write your name and number in all of the files of your project, in the following format:
  ```
  /* @Author
  Student Name:<studentname>
  Student ID :<studentid>
  Date:<date>*/
  ```
- Only electronic submissions through Ninova will be accepted no later than deadline
- Use comments wherever necessary in your code to explain what you did.