P4- Boggle
The whole picture

• Demonstration of Boggle game
• Lexicon/dictionary
• Board
• What words are legal?
What do you need to do

• Design efficient data structures for the lexicon and board
  -“Efficient” means that inserting/searching for a given word in the lexicon is fast. Also, it should be efficient enough to look for a given word on the board.

• Implement the required functions
Data structures

• Your data structures should reside in two files, boggleutil.h, boggleutil.cpp

• The list of operations your data structures should support:
  – For lexicon: Insert, Find
  – For Board:
    1. void setBoard(unsigned int rows, unsigned int cols, string** diceArray);
    2. vector<int> isOnBoard(const string& key);
Data structures (cont’d)

• Lexicon: Ternary Trie
  – How to build the ternary trie? What do you need?
    • Remind how you build the previous trees (binary tree, huffman tree)

  – Insert() and Find()

• Or you can use your own data structure!
Data structures (cont’d)

• Board: one-D array
  – Advantage1: Quick search for neighbours.
    1 operation vs. 2 operations.
  – Advantage2: One integer for one position.
    If you use two-D array, you have to use two integers for one position.

vector<int> isOnBoard(const string& word_to_check)

• Or you can use your own data structure!
The next step

• Now you are done with the first part-data structures.
• The next step is to implement the required functions.
What are the required functions

- void buildLexicon(const set<string>& word_list);
- void setBoard(unsigned int rows, unsigned int cols, string** diceArray);
- bool getAllValidWords(unsigned int minimum_word_length, set<string>* words);
- bool isInLexicon(const string& word_to_check);
- vector<int> isOnBoard(const string& word_to_check);
- void getCustomBoard(string** &new_board, unsigned int *rows, unsigned int *cols);
- ~BaseBogglePlayer() {}
Details

- All the required functions reside in `boggleplayer.h` and `boggleplayer.cpp`.

- Some functions have already been implemented within the data structures while the other functions can be built on the operations supported by our data structures.
Files you need to submit (final version)

- boggleutil.h, boggleutil.cpp
- boggleplayer.h, boggleplayer.cpp
How to test your codes

• You should write your Makefile or compile your codes manually.
• For the second step, try to pass the test of bogtest.cpp, which is a non-GUI test.
• If you want, you can run the GUI application, details has been posted in the README file.