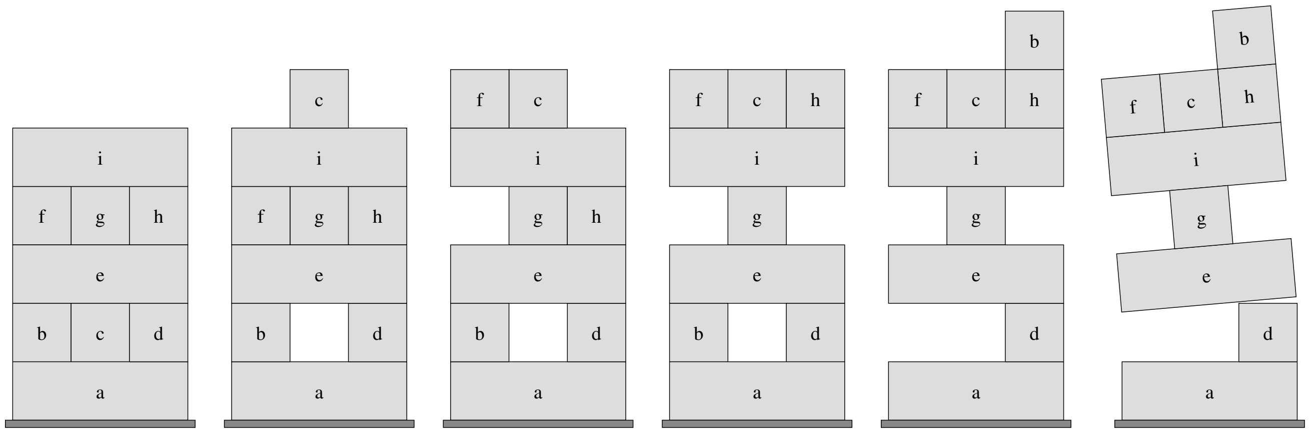
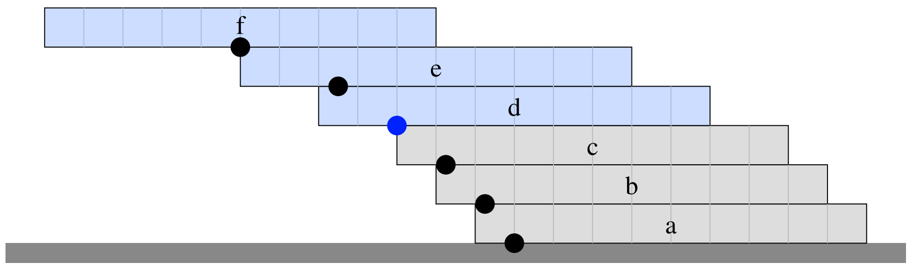
CSCI-1200 Data Structures — Fall 2024
Homework 5 — Jenga Blocks Stability

Ben Bitdiddle has teamed up with his cousin Brent Brickbuster (who is a mechanical engineering major) to create a 2D computer game that simulates the 3D physical block-building game, *Jenga.* A simplified illustration of the game is shown below. We start with a large collection of rectangular bricks sized 3x1x1 units. The game starts by building a tower shown in the leftmost image with 3 bricks on each level alternating the direction of the bricks (parallel with the page of this document, then perpendicular to the page). Then the players take alternating turns removing one brick and placing it on the top of the tower. This sample game ends when a player attempts to remove brick `b`, but the tower is no longer stable and balanced. The tower of blocks starts to lean and then spectacularly collapses! You can read more about the original game of *Jenga* here: https://en.wikipedia.org/wiki/Jenga.



Brent explains the physics of the block stacking game to Ben using the cantilevered stack of six blocks below right as an example. We work from top down to determine stability of each component of the structure. Brick `f` is stable and balanced at the top of the pile because exactly half of its weight is supported from below by brick `e`. NOTE: The situation is very precarious... a tiny breeze could cause brick `f` to lean and fall! The *center of mass* of brick `f` is visualized as a black dot. Each brick in the diagram is stable if the *combined center of mass* of that brick plus all bricks that rest on it from above is supported from below, by either another brick or the ground.

We determine that brick `d` is stable by finding all bricks that rest on it (`e` and `f`), and computing the combined center of mass (visualized with a dark blue dot) of those three bricks (shaded in light blue). Because the blue dot is *just barely* supported from below by brick `c` we say that component is stable.

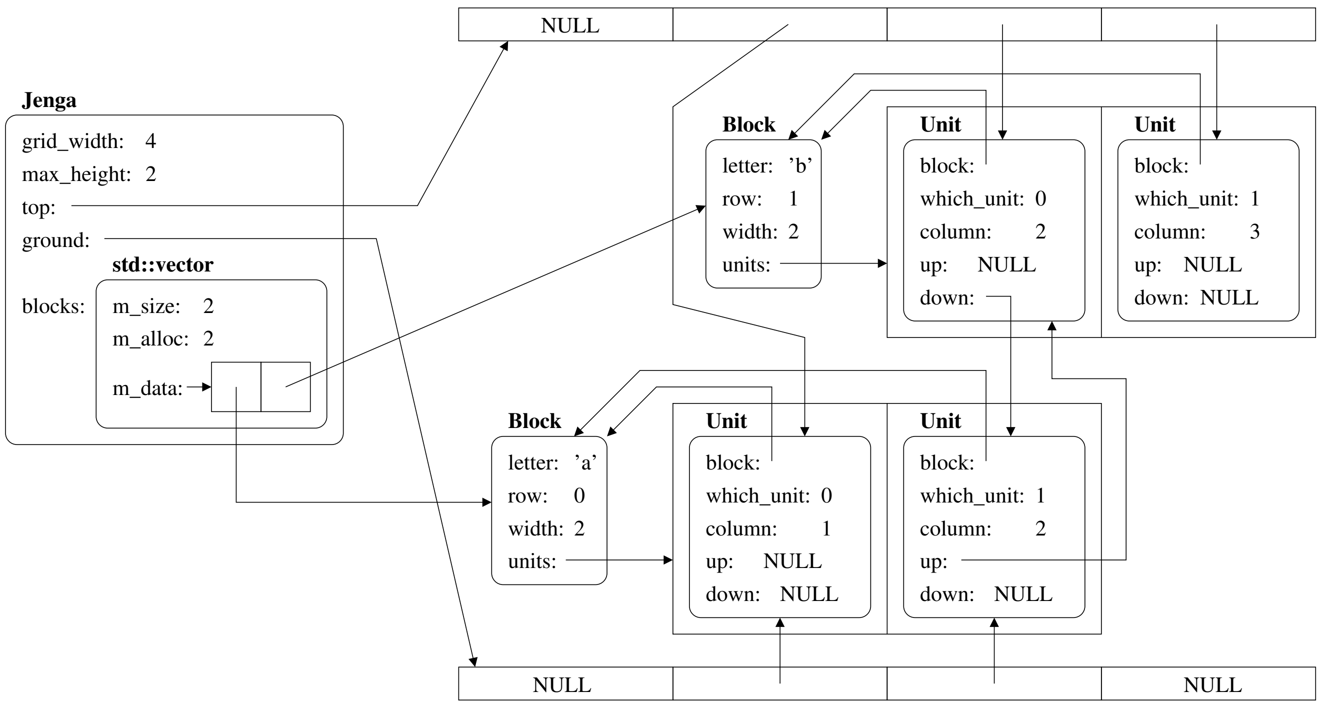


If the center of mass of each brick plus the bricks above it is supported, then the entire tower is stable. We can compute the center of mass of these rectangular solids by breaking each larger brick into squares of uniform size and mass and then simply averaging their positions.

Every brick in Ben and Brent's 2D Jenga game will be a simple rectangle with integer dimensions. The height will always be 1 unit tall – *if they are not square, the longer dimension will be placed horizontally.* The blocks will always be placed such that the left (and right) edges align with the 1 unit spacing on the horizontal axis.

The Linked Jenga Data Structure

Ben is excited to use this block stacking game as a chance to practice a modified linked-list implementation in C++. He has drawn the the diagram below showing how the `Jenga` class will serve as the overall manager for the tower configuration. The `Block` class will store the position of each rectangular brick in the diagram. Each `Block` object will store one or more `Unit` objects. Study this diagram carefully, and deduce the type of every member variable. Your implementation must follow this design exactly, with the same member variables (match the name and type exactly). You are not allowed to modify or add any additional variables. All of your member variables must be private. Note that there is only one STL `vector` object in the design. You should not use any other STL container (or the `Vec` or `ds_list` classes we discussed in lecture and lab).



Provided Code - Jenga Print, Interactive Mode, and Testing Framework

To further specify the required interface for the `Jenga`, `Block`, and `Unit` classes, and to aid in your development and testing, we provide the `main.cpp` and `jenga_print.cpp` files. You must use these files. You should not modify these files except to uncomment the tests as you progress with your implementation and add your own tests where specified. *Please read through the entire handout, and study the provided code before beginning your implementation.*

The code to `print` an ASCII art representation of the `Jenga` board is provided for you. There are optional arguments controlling the `print` function: increasing the resolution of the ASCII art grid, visualizing the center of mass of each brick, and labeling the `up` and `down` pointer links between `Units`.

The program is launched with a single `string` command line argument: which test case to run. The provided tests, which include the examples shown in this handout, demonstrate the required member functions for the `Jenga`, `Block`, and `Unit` classes. If no command line argument is provided, the program will run in *interactive mode* – taking commands from `std::cin` – giving an approximate preview of the physical simulation for Ben and Brent's Jenga game – but without the game rules.

Extra Credit: Physics and Static Indeterminacy

The *Maximal Overhang* problem asks "How far can a stack of n identical blocks be made to hang over the edge of a table?" The optimal answer when we allow only 1 block per level/row is well-understood (it is similar to our *cantilever* example above, except the blocks are not restricted to integer positions on the horizontal axis). The physics of the problem are more complicated in both theory and in physical reality when we consider designs with multiple bricks per row. See the images and discussion on the first four pages of the following paper:

"Maximum overhang", M. Paterson, Y. Peres, M. Thorup, P. Winkler and U. Zwick, American Mathematical Monthly, Volume 116, #9, November 2009, pages 763-787.

When one brick in a Jenga block construction has direct contact below with two or more other bricks, we likely do not have a unique solution for the forces necessary to support the structure. *In fact, this is why Jenga players usually have multiple choices for which brick they will move next!* To demonstrate that a specific tower structure is stable, we only need to identify one valid set of forces that shows stability. For extra credit, you can begin to explore the complex physics of block stacking by detecting these cases of *static indeterminacy* and correctly determining the overall stability of the structure. Work through the provided extra credit test cases, but also make your own!

Additional Requirements, Hints, and Suggestions

The use of lists or vectors or other STL containers in this assignment is restricted. You must implement the data structure exactly as diagrammed, with the specified member variables and types. The `Jenga` class has one STL `vector` as part of the member variable representation. The `Jenga::print` functions use an STL `vector` of STL `string` to prepare the ASCII art visualization of the current tower. You should avoid using additional instances of `vector` or `list` (or our `Vec` or `ds_list` classes from lecture and lab) and instead complete the implementation by navigating the links between `Unit` and `Block` objects and using recursion. *NOTE: If you tackle the extra credit portion of the assignment, you are allowed to use an STL* `vector` *or* `list` *in a minor way for the extra credit physics and stability analysis – but you should not change the data structure diagram on the previous page.*

Unlike our last couple homeworks, you are not asked to analyze the Big O Notation of your implementation. The choice of data structures for this homework were designed for for convenient navigation *and* to give you practice constructing, connecting, modifying, and destroying linked structures.

Submitty will again be configured to use Dr. Memory to check your program for memory errors and memory leaks. Be sure to use Dr. Memory or Valgrind on your local machine as you develop to catch these problems early. Use good coding style when you design and implement your program. Be sure to make up new test cases and don't forget to comment your code! Please use the provided template `README.txt` file for any notes you want the grader to read. You must do this assignment on your own, as described in the "Collaboration Policy & Academic Integrity" handout. If you did discuss this assignment, problem solving techniques, or error messages, etc. with anyone, please list their names in your `README.txt` file.