**Bonus**

Memory management via vector smart pointers

We applied unique\_ptr to prevent memory leak. For how they have been implemented, please check Design part of this report. One challenge during the implementation is distinguishing when to use raw pointer to create a has relationship and when to take ownership with unique\_ptr. A Block has a field of vector of cell pointers to form a “has a” relationship. When the Block is destroyed no Cells should ever be destroyed from the Board. Thus raw pointers need to be used for this purpose. Another challenge during the implementation is to accommodate the constantly changing current and next block. We have to utilize methods such as “reset” and “move” to transfer property ownership and facilitate calls to dtors.

Next block when level up and level down

As required by the guideline, leveling up or down the block showing as next still comes next but subsequent blocks are generated using the new level. The slight issue with this is that after changing the level, the player has to do two drop to start experiencing the features of that level, which creates a feeling of lag. So we have decided to implement a feature that keeps the next block’s type while giving the next block properties of the next level when leveling up or down is called to enhance player experience. The challenge of implementing this feature is to be able to create any block type we desire for any level generator. So we added a parameter in the generate method in the abstract super Level class and implemented this for all Level subclasses which inherits this method. When a ‘N’ is given to the parameter, the generate method will return a block according to the generation specification of that level, otherwise when a Block type is given, it will generate a specific Block of that type which also has the properties of that level.