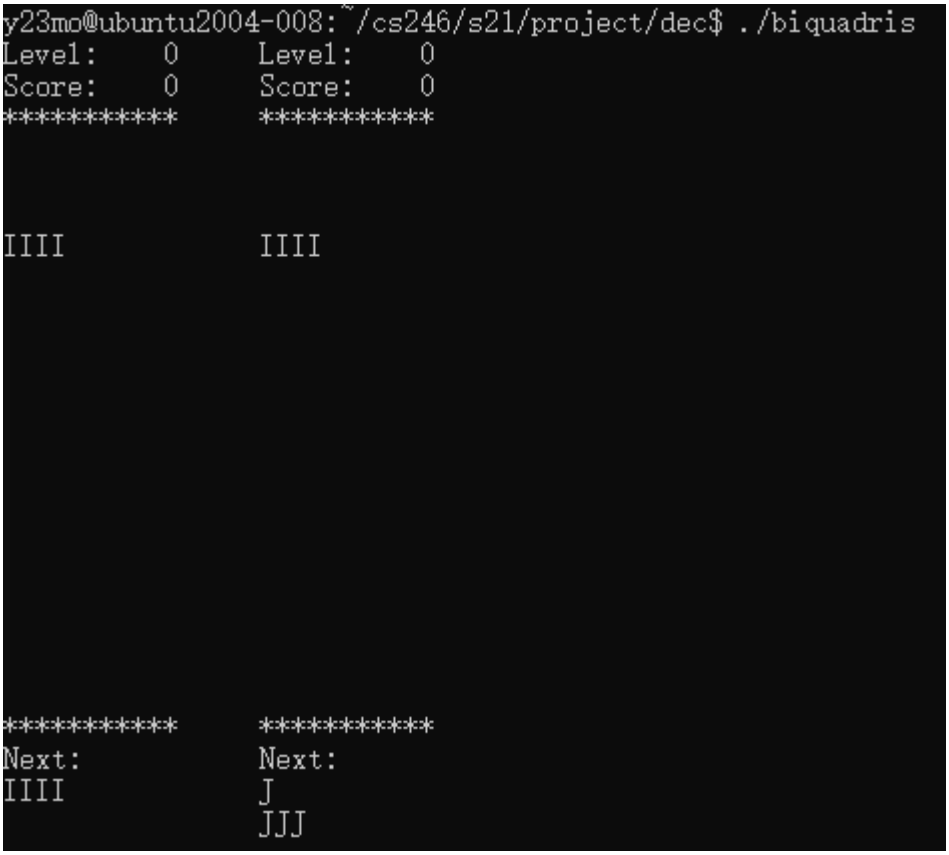
Program Demonstration

By Yilin Mo and Yixing Zhang

1. Start the program

To start the program, enter make command, and then enter ./biquradris.
After the program starts, it will print the game boards of two players on standard output, with the first block of each player already generated at the default positions. The size of each player's board is 18 rows x 11 columns, and the first 3 rows are reserved for possible rotations. The boards are displayed below.



2. Level

My group only implemented level 0 of biquadris(read in blocks from a file), so actions in level 1-4 are not valid.

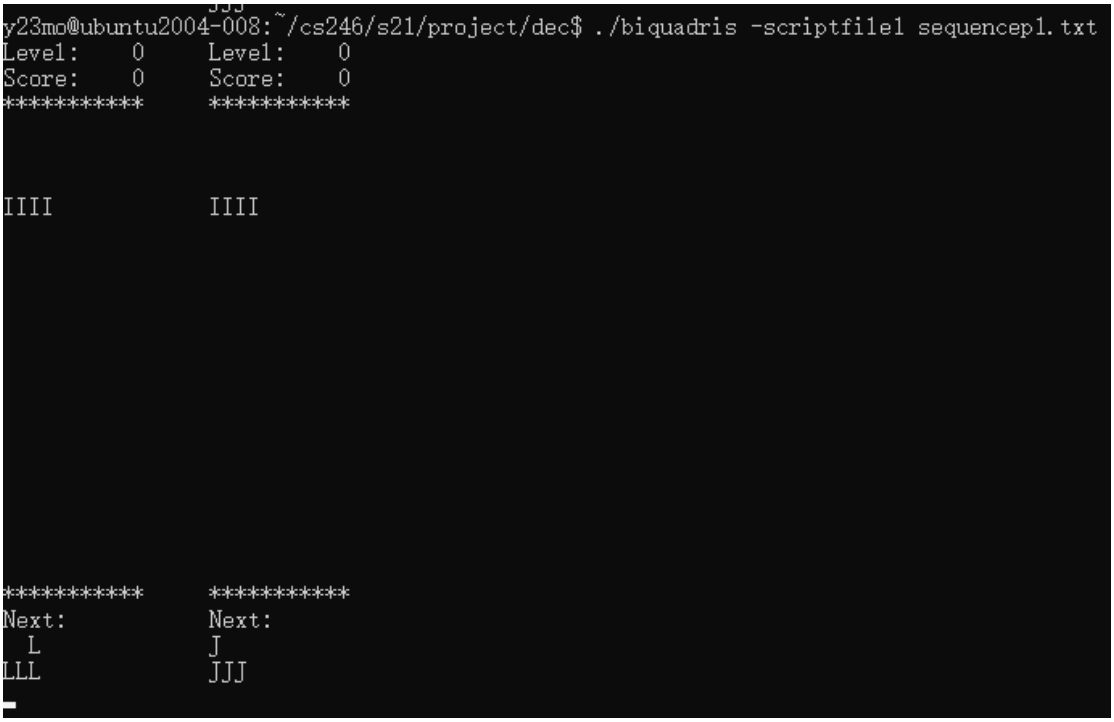
3. Command-line arguments

My group implemented -scriptfile1, -scriptfile2, and -startlevel. Other arguments are not valid.

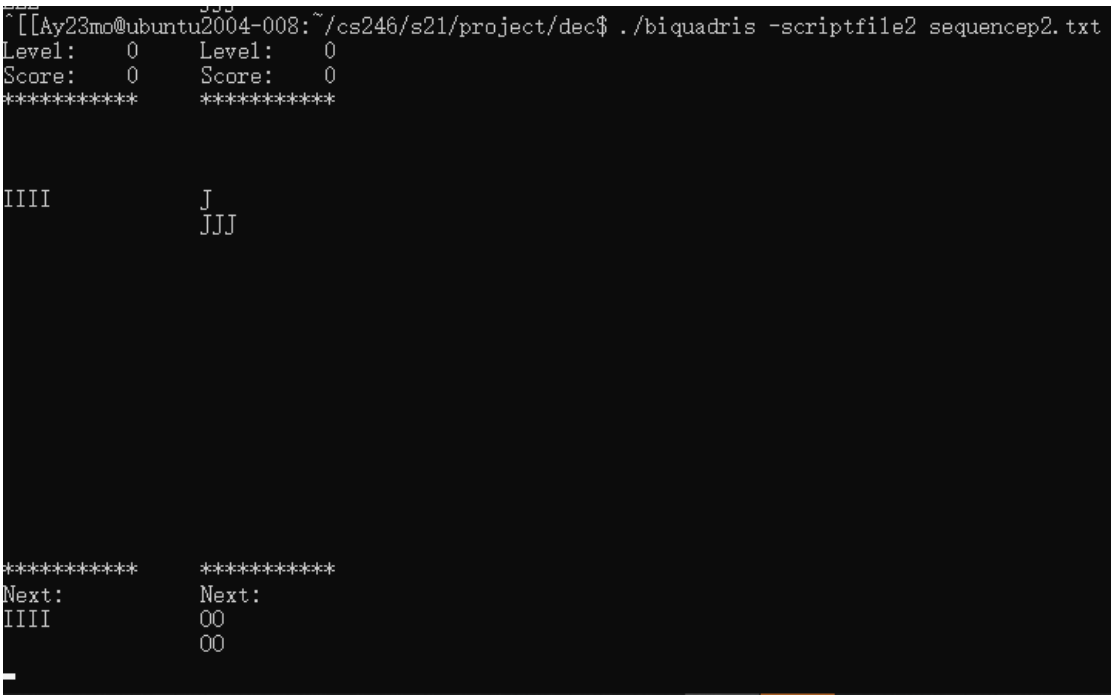
1) -scriptfile1 and -scriptfile2

-scriptfile1 xxx replaces "sequence1.txt" with xxx as the file of input block sequence for player1. The example below replaces "sequence1.txt" with

“sequencep1.txt” for player1.



-scriptfile2 xxx replaces “sequence2.txt” with xxx as the file of input block sequence for player2. The example below replaces “sequence2.txt” with “sequencep2.txt” for player2.



You can add both -scriptfile1 xxx and -scriptfile2 xxx on the command line. This action replaces input block file for both players. If neither argument is added, the program will take input from sequence1.txt for player1 and sequence2.txt for player2. We have provided these two files. sequence1.txt has input: I I I T and sequence2.txt has input: I J L O. If the program reaches the

end of such file, it will go back to the first input in the file and loop through the file again.
We have also provided sequencep1.txt and sequencep2.txt in the zip file for you to test on yourself. sequencep1.txt has input: I L S T and sequence2.txt has input: J O Z.

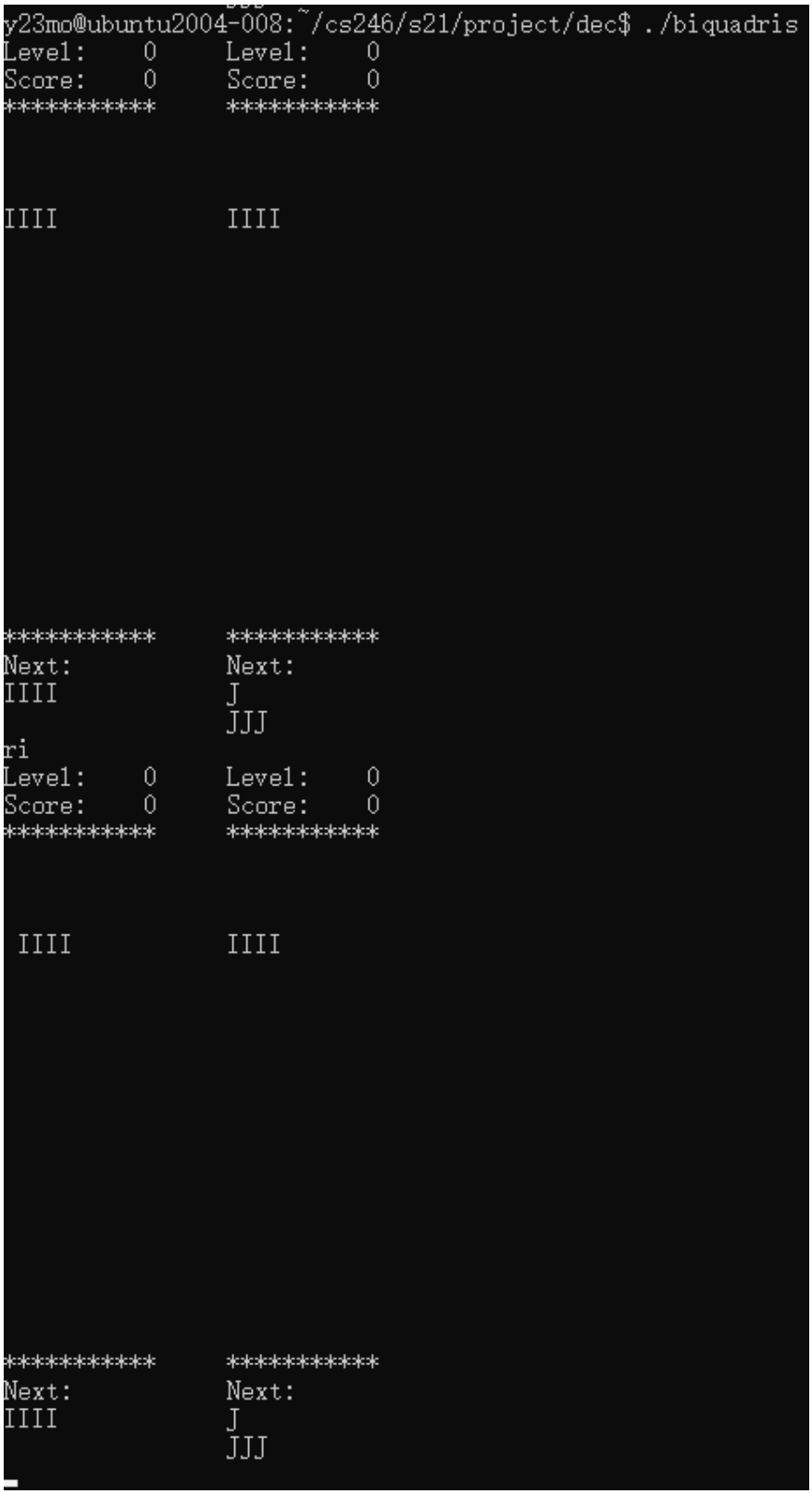
2) -startlevel

-startlevel xxx accepts only integers from 0 to 4 inclusively. Other integers are not valid. The program will change the level of both players from default 0 to xxx. However, since we did not implement level 1 to level 4, this argument only changes the levels of both players on the displayed board, and the program still behaves as if in level 0. An example is provided below:

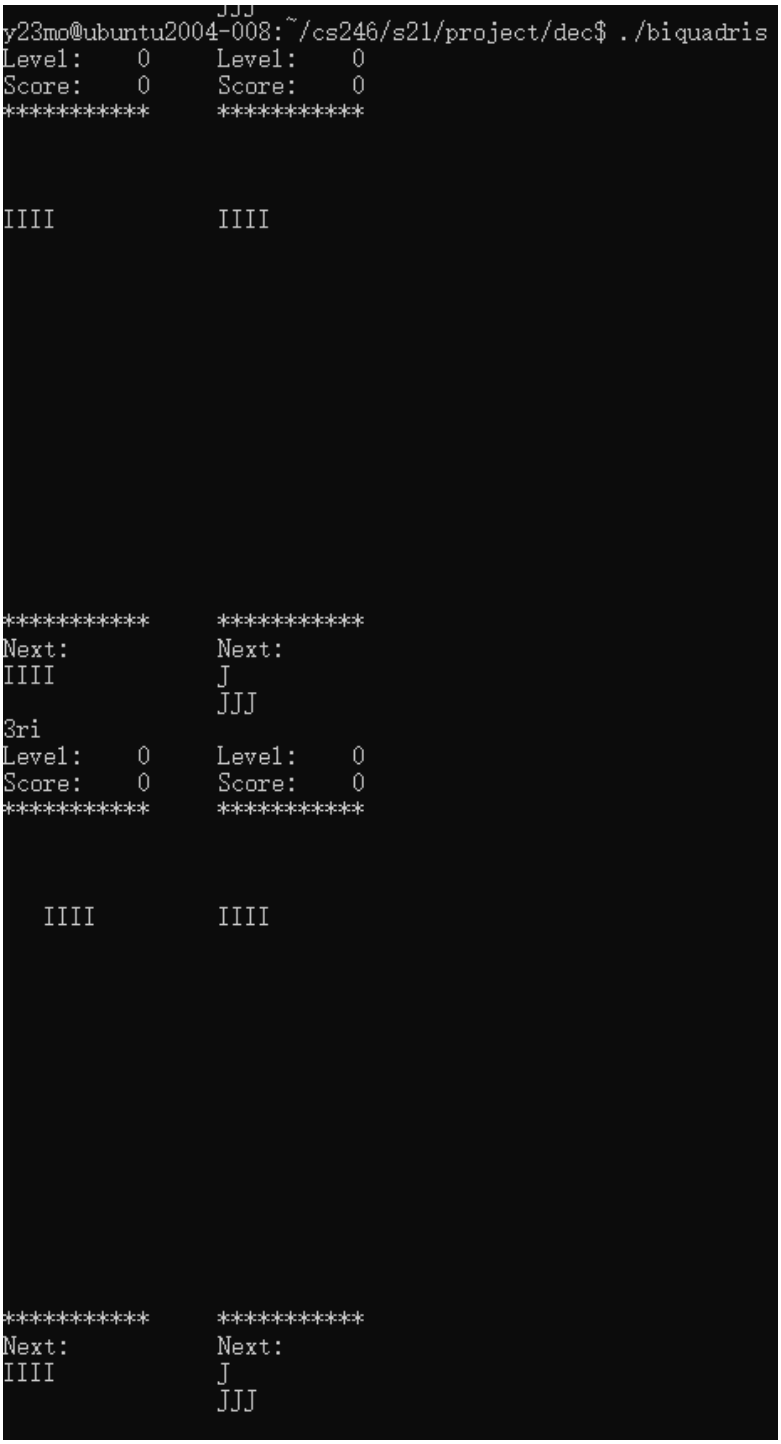
4. Command interpreter

We implemented the following commands: left, right, down, drop, levelup, leveldown, sequence, and restart. Other commands are not valid. All of these commands except sequence can take simplified form(e.g. left can be typed as lef). In addition, all of these commands except sequence can have multiplier prefix. However, for command restart, if the multiplier is more than 1, the board only restarts for one time. Also note that drop command with multiplier will drop consecutive blocks of that player, and then enter the other player's round. For example, 3dr drops the current block and 2 more consecutive blocks of only the player at that round. Then the other player's round begins.

Below is an example of simplified command:

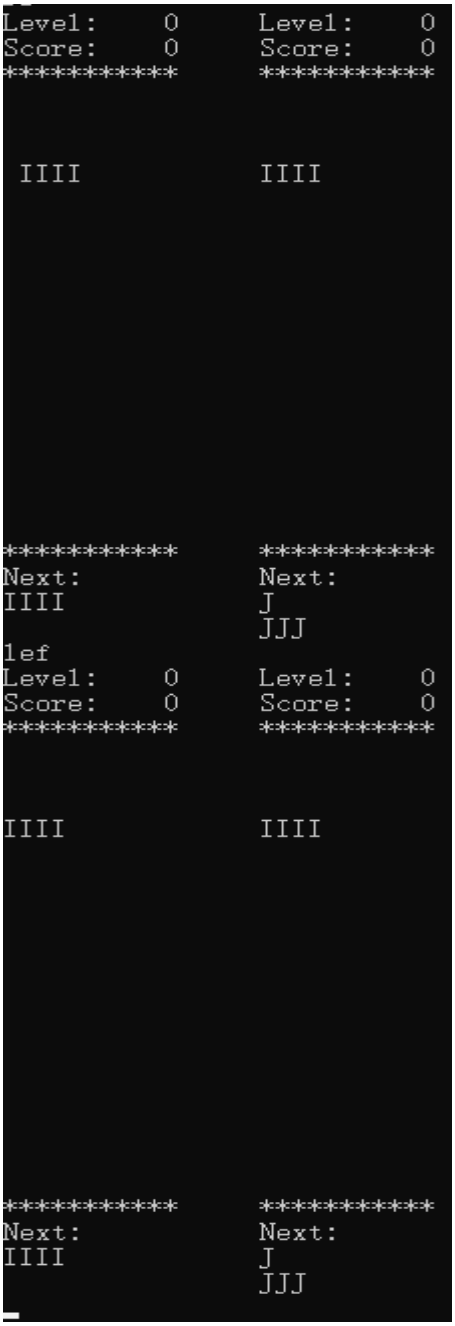


Below is an example of 3ri(right with multiplier 3):



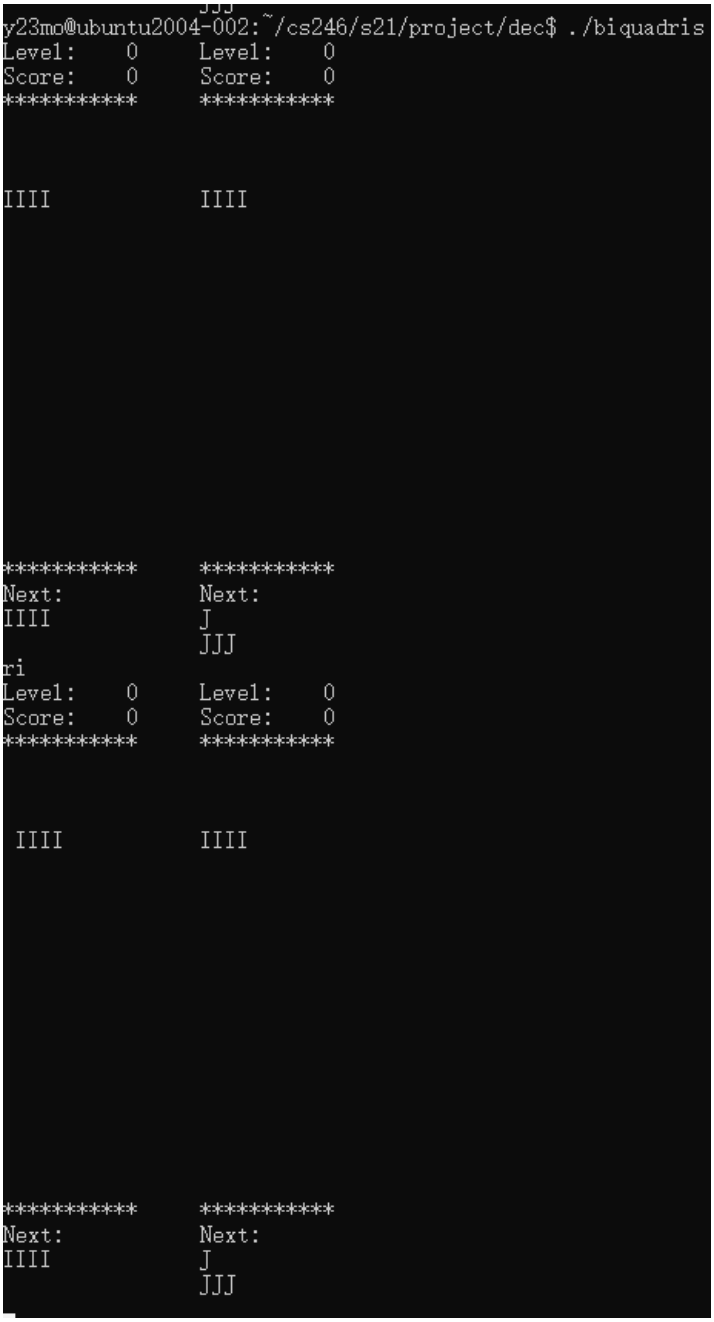
1) left

left command moves the current block left by one cell, if the block's left side is not edge and there is no block at the immediate left side of the current block. Otherwise, it does nothing. Below is an example of left:



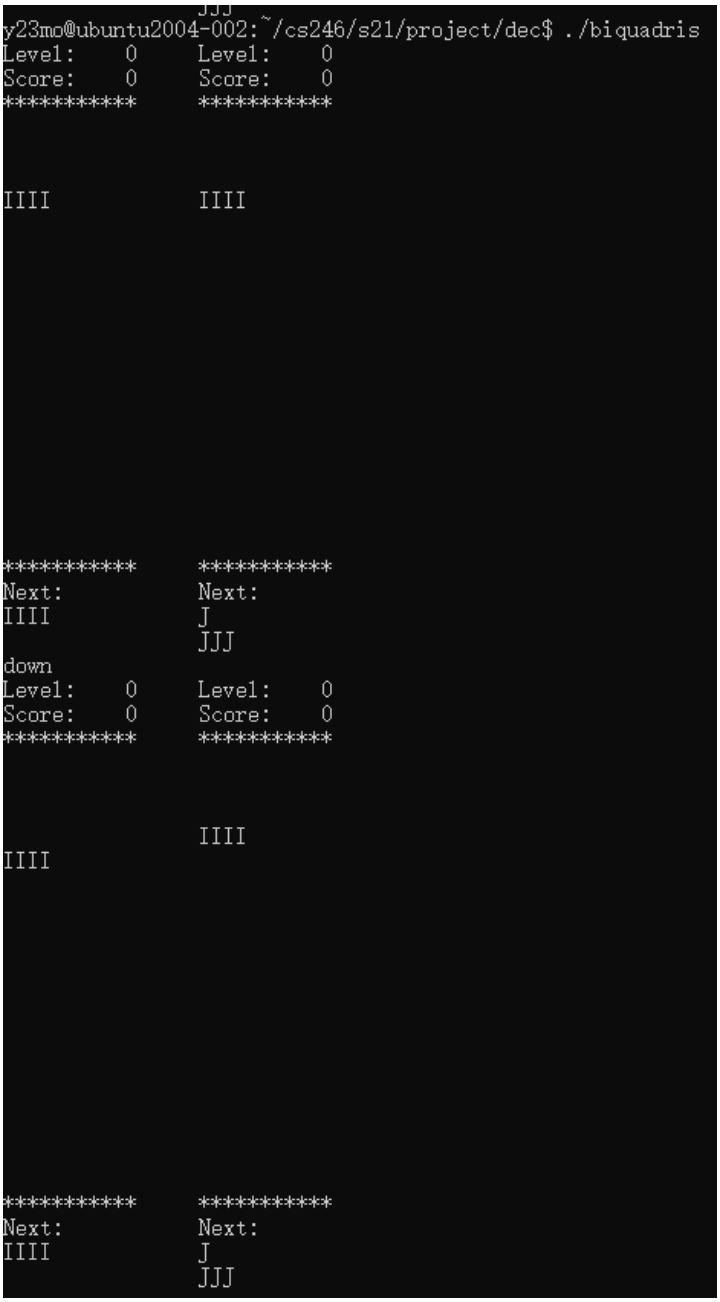
2) right

right command moves the current block right by one cell, if the block's right side is not edge and there is no block at the immediate right side of the current block. Otherwise, it does nothing. Below is an example of right:



3) down

down command moves the current block down by one cell, if the block's bottom side is not edge and there is no block at the immediate bottom side of the current block. Otherwise, it does nothing. Below is an example of down:



4) drop

drop command moves the current block down until it cannot be further moved downward. Then the program calls Grid::rowScore() to add possible earned score as the result of filling rows, calls Grid::rowsDelete() to delete filled rows and add possible earned score as the result of fully clearing blocks. Then the programs calls Grid::update() to update the current block and next block information. Then the program calls Grid::setDefault() to set the block of the next round at the default position. In the end, the program changes turn so that the round of the other player begins. Below is an example of drop:

5) levelup and leveldown

levelup increases the level of only the player at that round by 1, while leveldown decreases the level of only the player at that round by 1. For example, in the round of player1, levelup increases only the level of player1 by 1, while leveldown decreases only the level of player1. by 1. Note that if the player's original level is already 4, levelup dose nothing; if the player's original level is already 0, leveldown dose nothing. Since we did not implement level 1 to level 4, these commands only changes the level of one

player on the displayed board, and the program still behaves as if in level 0.

6) Sequence

Command sequence xxx uses the commands in file xxx instead of reading from standard input. This command does not support simplified form and multiplier prefix. For example, if we have command sequence a.txt, and a.txt contains command ri ri down. Then command sequence is equal to moving the current block right by 2 units and then drop it. We have provided a file called commands.txt in the zip file for you to test on your own, which contains all sorts of simplified commands and commands with multiplier prefix. The content of commands.txt is: 0re 1rest 3restart 0ri 1rig 3right 0lef 1left 3left 0do 1dow 3down 0dr 1dro 3drop 0levelu 1levelu 3levelup 0leveld 1leveldo 3leveldown

7) Restart

Restart clears the board, score, level, and block sequence of only the player at that round, builds a new empty board for that player, assigns the score of 0 and the level of 0 for that player, and uses the block sequence from the beginning of that player's original input block sequence file. Then the first block is set at the default position. It is still that player's round. For example, player1 with input block sequence file a.txt enters command restart at its round. Then only player1's board is deleted and replaced with an new empty board; Only player1's score and level are cleared as 0; player1's block sequence is counted from the beginning of a.txt. The first block in a.txt is set at the default position on player1's new board. It is still player1's round. Below is an example of restart(restart the board at right):

5. Display

We only implemented the text display, so the program has no graphical display.

6. Scoring

Players have two ways to earn score: by filling entire rows with blocks and by completely clearing blocks from the board. When a player enters drop command, after the current block is dropped, the program scans the player's board to find

filled rows and add score. The score added is equal to the number of deleted rows squared. Then the program calls Grid::rowsDelete() to delete the filled rows one by one and moves the upward blocks downward. When the program deletes rows, if it finds that some about-to-delete cells are the last cells of their blocks, the score is added again, one for each fully deleted block. We have provided a command sequence in score.txt to simulate such scoring action. You only need to start the program with no command-line argument, and enter *sequence score.txt* to view the result(In the last step, one row at the left board is cleared, along with two I blocks, so 3 marks are added). The content of score.txt is: dr dr 4ri dr dr dr dr 20ri dr dr dr dr 4ri dr. In addition, the hi score is tracked every time when a player enters drop command.

7. End the program

Players have two ways to end the program:

1. Enter EOF(end of file) to end the program at any time when the program executes. To enter EOF, simply enter Ctrl+D on the keyboard.
2. Keep playing the game until one player's blocks have piled up so high that the new block cannot fit into the default position. Then the other player wins, and the game ends.