Please enter your name here:

Please enter your Academic Declaration here:

Description of the coursework: SkraBBKle

Logistics

On Moodle, behind the “Assessment” tile, you will find information about submission deadlines. You will also find an invitation link for this coursework assignment. To get access to your GitHub Classroom repository with your initial files, you will need to visit the invitation link on Moodle and accept the invitation. You can then pull from/push to this GitHub repository as usual.

Overview

This coursework takes inspiration from the board game Scrabble, although it has certain differences with the standard Scrabble game. To reduce your workload, *SkraBBKle* has some simplification over Scrabble, but also some generalisations. The game will be played between a **Human player** and a **Computer player**. However, the board size will be S x S, where S is an integer number between 11 and 26, instead of the usual 15 x 15. (We only set the limit 26 to avoid visualisation/printing problems for large boards.)

The rules of SkraBBKle have been inspired by the ones available here for the Scrabble game.

The goal of the game is to populate the board with a grid of tiles, similar to a crossword, so that all words on the board are from a pre-defined word list. Both players have a rack of up to 7 tiles which is refilled from a tile bag after every move. Each tile has a letter and a numerical value.

When the game is started, the user is asked whether to load a specific board from the file system or whether to use the default board. The default board corresponds to the Scrabble board.

After that, both players get 7 randomly chosen tiles from the tile bag. The human player is then shown the (currently empty) board and their own tiles. The human player plays the first move.

Notation and Symbols

The columns will be designated by lower-case letter characters from a to z and the rows by numbers from 1 to 26. The leftmost column is a and the top row is 1.

Gameplay

1. The first move is done by the human player, by combining two or more of their letters to a word. The word is placed on the board to read either to the right or downwards, with one letter on the centre square. Like in Scrabble, diagonal words are not possible in SkraBBKle. Whenever a tile is placed on a square, the letter and the value of the tile replace the square on the board. If the value `S` from the intro is an even number, there are four candidates for the role of “centre square” that are at minimal distance from the “true” centre of the board (the point where the four squares meet). In SkraBBKle, the top left of these four squares (i.e., the one with minimal column and row) is the centre square for the purposes of the first move.

   For example, on a board with `S = 16`, there are four squares at the same minimal distance from the “true” centre point of the board: h8, h9, i8, and i9. Each of these “candidates” for the role of centre square has one column and one row too many, either before or after the candidate. To make unambiguous which square must be used for the first move, “the top left of these four squares” (in the example: h8) is chosen for SkraBBKle as the “centre square”.

2. The game computes the score resulting from the move. As long as the tile bag is not empty, the player who just made a move will have their tile rack topped up with tiles taken from the tile bag so that it has 7 tiles again.

3. The next move is the computer player’s. From now on, the computer player and the human player take turns with their moves until one player has no more tiles on their rack or no more moves are possible.

4. The player whose turn it is adds one or more letters to the letters already on the board to form a new word. All letters played in a move must be placed in one row to the right or downwards and contribute to a new word. It is allowed to skip occupied positions (for example, one may extend `NO` to `SNOW` by adding a `S` at the beginning and at the same time a `W` at the end). In SkraBBKle, every move may lead to only one occurrence of a new or changed word on the board. In contrast to the original Scrabble game, it is not allowed to place a word at right angles to a word already on the board without an overlap, nor to place a complete word parallel immediately next to a word already played.

5. Once a tile has been used in a move, its position on the board stays unchanged.

6. There are special tiles, called wildcards or blanks, where the letter is initially not given, but can be chosen by the player as needed. In our game, the choice is entered as a small letter. As soon as the choice has been made for a given wildcard, it stays fixed for the rest of the game. The value of a wildcard in SkraBBKle is 5. When a wildcard has been played, its letter used for display on the board is the chosen small letter - the choice lasts for the whole game.

7. A player is allowed to pass the current turn (i.e., not make a move and allow the next player to continue).

8. In SkraBBKle, it is not allowed to exchange tiles from the tile rack with tiles from the tile bag.

9. Moves must always lead to words that are in the game’s word list. (This is checked by the game, and invalid moves are rejected by the game. The word list can be found in the file `resources/wordlist.txt`.)

10. The game ends when the tile bag is empty and one of the player has an empty tile rack. The game also ends if both players pass twice in a row.

Scoring

1. The score for each move is calculated as the sum of the letter values in the word created or modified in the move, plus the extra points obtained (or lost!) from tiles placed on *premium squares*.
2. **Premium letter squares** have an integer number of at least `-9` and at most 99 as a factor (specifically, 0 or negative values are possible). When a tile is placed on a premium letter square, the score for the tile is its value multiplied by the factor of the premium letter square. A premium letter square has the shape `(x)` if the factor x is a single character and the shape `(xy` if the factor xy has two characters.
3. **Premium word squares** also have an integer number of at least `-9` and at most 99 as a factor (specifically, 0 or negative values are possible). When a move places a tile on a premium word square, the factor of the premium word square will be multiplied with the score obtained for the word otherwise. If a move uses several premium word squares, the effect is cumulative (for example, when we use a premium word square with factor 4 and a second premium word square with factor 5 in the same move, the resulting factor for the word score would be `4*5 = 20`). Premium word squares are applied only *after* premium letter squares. A premium word square has the shape `{x}` if the factor x is a single character and the shape `{xy` if the factor xy has two characters.
4. A square cannot be at the same time both a premium letter square and a premium word square. There can also be squares that are not premium letter squares or premium word squares. Such squares are displayed as ` . ` (i.e., a space, then a dot, then a space).
5. Letter and word premium squares are applied only in a single move. As soon as they have been covered by a tile, in later moves this tile will count at its face value (i.e., the score will not be affected by the premium formerly obtained from covering the tile).
6. Woo-hoo! If a player manages to play all 7 tiles in one move, they are awarded an extra score of 75 points in SkraBBKle. This extra score is added only after all the other calculations for the current move are done (so also with a premium word square involved in the move, the player would still get only 75 extra score points).
7. At the end of the game, at least one player will have unplayed tiles. Each player's score is reduced by the sum of the values of their own unplayed tiles.

Winning player

The player who has a higher score at the end of the game wins. If the scores are equal, the game is declared a draw.

Initial tile bag

SkraBBKle is played with a fixed initial tile bag. It contains the individual tiles with the following quantities and values (which are slightly different from Scrabble):

8 x [A1]
2 x [B3]

2 x [C3]
4 x [D2]
10 x [E1]
3 x [F4]
4 x [G2]
3 x [H4]
8 x [I1]
1 x [J9]
1 x [K6]
4 x [L1]
2 x [M3]
7 x [N1]
7 x [O1]
2 x [P3]
1 x [Q12]
6 x [R1]
4 x [S1]
6 x [T1]
5 x [U1]
2 x [V4]
1 x [W4]
1 x [X9]
2 x [Y5]
1 x [Z11]

Here, the line

8 x [A1]

means: “there are 8 tiles with the letter A and the value 1”

Moreover, there are two wildcards “[_5]” of value 5 in the initial tile bag.

Example play

For example, after starting the game, the human player may have the following interaction with the game:

```
============                   ============
============ S k r a B B K l e ============
============                   ============

Would you like to _l_oad a board or use the _d_efault board?
Please enter your choice (l/d): d

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  . {12 .  . (2) .  .  . {3} .  .  . (2) .  . {8}  1
 2  .  . {2} .  .  . (3) .  .  . (3) .  .  . {2} .   2
 3  .  .  . {2} .  .  . (2) . (2) .  .  . {2} .  .   3
 4  . (2) .  . {2} .  .  . (2) .  .  . {2} .  . (2)  4
 5  .  .  .  .  . {2} .  .  .  .  . {0} .  .  .  .   5
 6  .  . (3) .  .  . (3) .  .  . (3) .  .  . (3) .   6
 7  .  .  . (2) .  .  . (2) . (2) .  .  . (2) .  .   7
 8 (3){-3 .  . (2) .  .  . {2} .  .  . (-4 .  . {-2  8
 9  .  .  . (2) .  .  . (2) . (2) .  .  . (2) .  .   9
10  .  . (3) .  .  . (3) .  .  . (3) .  .  . (3) .  10
11  .  .  .  .  . {2} .  .  .  .  . {2} .  .  .  .  11
12  . (2) .  . {-1 .  .  . (0) .  .  . {-1 .  . (2) 12
13  .  .  . {2} .  .  . (2) . (9) .  .  . {2} .  .  13
14  .  . {2} .  .  . (3) .  .  . (3) .  .  . {2} .  14
15  . {9} .  . (2) .  .  . {3} .  .  . (2) .  . {16 15
16  .  .  .  .  .  .  .  . (3) .  .  .  .  .  .  .  16

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

Would you like to play an _o_pen or a _c_losed game?
Please enter your choice (o/c): o
OPEN GAME: The computer's tiles:
OPEN GAME: [E1], [U1], [D2], [I1], [_3], [I1], [P3]
It's your turn! Your tiles:
[T1], [N1], [C3], [N1], [I1], [U1], [D2]
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles
and lower-case letters are wildcards.
Entering "," passes the turn.
DINT,h6
The move is:    Word: DINT at position h6
Human player score:    7
Computer player score: 0

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  . {12 .  . (2) .  .  . {3} .  .  . (2) .  . {8}  1
 2  .  . {2} .  .  . (3) .  .  . (3) .  .  . {2} .   2
 3  .  .  . {2} .  .  . (2) . (2) .  .  . {2} .  .   3
 4  . (2) .  . {2} .  .  . (2) .  .  . {2} .  . (2)  4
 5  .  .  .  .  . {2} .  .  .  .  . {0} .  .  .  .   5
 6  .  . (3) .  .  . (3)D2  .  . (3) .  .  . (3) .   6
 7  .  .  . (2) .  .  . I1  . (2) .  .  . (2) .  .   7
 8 (3){-3 .  . (2) .  . N1 {2} .  .  . (-4 .  . {-2  8
 9  .  .  . (2) .  .  . T1  . (2) .  .  . (2) .  .   9
10  .  . (3) .  .  . (3) .  .  . (3) .  .  . (3) .  10
11  .  .  .  .  . {2} .  .  .  .  . {2} .  .  .  .  11
12  . (2) .  . {-1 .  .  . (0) .  .  . {-1 .  . (2) 12
13  .  .  . {2} .  .  . (2) . (9) .  .  . {2} .  .  13
14  .  . {2} .  .  . (3) .  .  . (3) .  .  . {2} .  14
15  . {9} .  . (2) .  .  . {3} .  .  . (2) .  . {16 15
16  .  .  .  .  .  .  .  . (3) .  .  .  .  .  .  .  16

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p
```

```
OPEN GAME: The computer's tiles:
OPEN GAME: [E1], [U1], [D2], [I1], [_3], [I1], [P3]
It's the computer's turn!
The move is:    Word: UPItIED at position 8g
Human player score:    7
Computer player score: 91

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  . {12 .  . (2) .  .  . {3} .  .  . (2) .  . {8}  1
 2  .  . {2} .  .  . (3) .  .  . (3) .  .  . {2} .   2
 3  .  .  . {2} .  .  . (2) . (2) .  .  . {2} .  .   3
 4  . (2) .  . {2} .  .  . (2) .  .  . {2} .  . (2)  4
 5  .  .  .  .  . {2} .  .  .  .  . {0} .  .  .  .   5
 6  .  . (3) .  .  . (3)D2  .  . (3) .  .  . (3) .   6
 7  .  .  . (2) .  .  . I1  . (2) .  .  . (2) .  .   7
 8 (3){-3 .  . (2) . U1 N1 P3 I1 t3 I1 E1 D2  . {-2  8
 9  .  .  . (2) .  .  . T1  . (2) .  .  . (2) .  .   9
10  .  . (3) .  .  . (3) .  .  . (3) .  .  . (3) .  10
11  .  .  .  .  . {2} .  .  .  .  . {2} .  .  .  .  11
12  . (2) .  . {-1 .  .  . (0) .  .  . {-1 .  . (2) 12
13  .  .  . {2} .  .  . (2) . (9) .  .  . {2} .  .  13
14  .  . {2} .  .  . (3) .  .  . (3) .  .  . {2} .  14
15  . {9} .  . (2) .  .  . {3} .  .  . (2) .  . {16 15
16  .  .  .  .  .  .  .  . (3) .  .  .  .  .  .  .  16

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

OPEN GAME: The computer's tiles:
OPEN GAME: [L1], [G2], [_3], [N1], [R1], [R1], [U1]
It's your turn! Your tiles:
[C3], [N1], [U1], [D2], [I1], [E1], [A1]
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles
and lower-case letters are wildcards.
Entering "," passes the turn.
ANCE,i6,,,,,,,
Illegal move format
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles
and lower-case letters are wildcards.
Entering "," passes the turn.
ANCE,i6
```

```
The board does not permit word ANCE at position i6. Please try again.
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles
and lower-case letters are wildcards.
Entering "," passes the turn.
ANCE,6i
The move is:    Word: ANCE at position 6i
Human player score:    21
Computer player score: 91

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  . {12 .  . (2) .  .  . {3} .  .  . (2) .  . {8}  1
 2  .  . {2} .  .  . (3) .  .  . (3) .  .  . {2} .   2
 3  .  .  . {2} .  .  . (2) . (2) .  .  . {2} .  .   3
 4  . (2) .  . {2} .  .  . (2) .  .  . {2} .  . (2)  4
 5  .  .  .  .  . {2} .  .  .  .  . {0} .  .  .  .   5
 6  .  . (3) .  .  . (3)D2 A1 N1 C3 E1  .  . (3) .   6
 7  .  .  . (2) .  .  . I1  . (2) .  .  . (2) .  .   7
 8 (3){-3 .  . (2) . U1 N1 P3 I1 t3 I1 E1 D2  . {-2  8
 9  .  .  . (2) .  .  . T1  . (2) .  .  . (2) .  .   9
10  .  . (3) .  .  . (3) .  .  . (3) .  .  . (3) .  10
11  .  .  .  .  . {2} .  .  .  .  . {2} .  .  .  .  11
12  . (2) .  . {-1 .  .  . (0) .  .  . {-1 .  . (2) 12
13  .  .  . {2} .  .  . (2) . (9) .  .  . {2} .  .  13
14  .  . {2} .  .  . (3) .  .  . (3) .  .  . {2} .  14
15  . {9} .  . (2) .  .  . {3} .  .  . (2) .  . {16 15
16  .  .  .  .  .  .  .  . (3) .  .  .  .  .  .  .  16

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

OPEN GAME: The computer's tiles:
OPEN GAME: [L1], [G2], [_3], [N1], [R1], [R1], [U1]
It's the computer's turn!
The move is:    Word: GURNaR at position n2
Human player score:    21
Computer player score: 115

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  . {12 .  . (2) .  .  . {3} .  .  . (2) .  . {8}  1
 2  .  . {2} .  .  . (3) .  .  . (3) .  . G2 {2} .   2
 3  .  .  . {2} .  .  . (2) . (2) .  .  . U1  .  .   3
 4  . (2) .  . {2} .  .  . (2) .  .  . {2}R1  . (2)  4
 5  .  .  .  .  . {2} .  .  .  .  . {0} . N1  .  .   5
 6  .  . (3) .  .  . (3)D2 A1 N1 C3 E1  . a3 (3) .   6
 7  .  .  . (2) .  .  . I1  . (2) .  .  . R1  .  .   7
 8 (3){-3 .  . (2) . U1 N1 P3 I1 t3 I1 E1 D2  . {-2  8
 9  .  .  . (2) .  .  . T1  . (2) .  .  . (2) .  .   9
10  .  . (3) .  .  . (3) .  .  . (3) .  .  . (3) .  10
11  .  .  .  .  . {2} .  .  .  .  . {2} .  .  .  .  11
12  . (2) .  . {-1 .  .  . (0) .  .  . {-1 .  . (2) 12
13  .  .  . {2} .  .  . (2) . (9) .  .  . {2} .  .  13
14  .  . {2} .  .  . (3) .  .  . (3) .  .  . {2} .  14
15  . {9} .  . (2) .  .  . {3} .  .  . (2) .  . {16 15
16  .  .  .  .  .  .  .  . (3) .  .  .  .  .  .  .  16

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

OPEN GAME: The computer's tiles:
OPEN GAME: [L1], [R1], [E1], [L1], [S1], [G2], [T1]
It's your turn! Your tiles:
[U1], [D2], [I1], [E1], [A1], [S1], [I1]
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles
and lower-case letters are wildcards.
Entering "," passes the turn.
DAE,4l
The move is:    Word: DAE at position 4l
Human player score:    31
Computer player score: 115

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  . {12 .  . (2) .  .  . {3} .  .  . (2) .  . {8}  1
 2  .  . {2} .  .  . (3) .  .  . (3) .  . G2 {2} .   2
 3  .  .  . {2} .  .  . (2) . (2) .  .  . U1  .  .   3
 4  . (2) .  . {2} .  .  . (2) .  . D2 A1 R1 E1 (2)  4
 5  .  .  .  .  . {2} .  .  .  .  . {0} . N1  .  .   5
 6  .  . (3) .  .  . (3)D2 A1 N1 C3 E1  . a3 (3) .   6
 7  .  .  . (2) .  .  . I1  . (2) .  .  . R1  .  .   7
 8 (3){-3 .  . (2) . U1 N1 P3 I1 t3 I1 E1 D2  . {-2  8
 9  .  .  . (2) .  .  . T1  . (2) .  .  . (2) .  .   9
10  .  . (3) .  .  . (3) .  .  . (3) .  .  . (3) .  10
11  .  .  .  .  . {2} .  .  .  .  . {2} .  .  .  .  11
12  . (2) .  . {-1 .  .  . (0) .  .  . {-1 .  . (2) 12
13  .  .  . {2} .  .  . (2) . (9) .  .  . {2} .  .  13
14  .  . {2} .  .  . (3) .  .  . (3) .  .  . {2} .  14
15  . {9} .  . (2) .  .  . {3} .  .  . (2) .  . {16 15
16  .  .  .  .  .  .  .  . (3) .  .  .  .  .  .  .  16

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

OPEN GAME: The computer's tiles:
```

```
OPEN GAME: [L1], [R1], [E1], [L1], [S1], [G2], [T1]
It's the computer's turn!
The move is:    Word: STELL at position i1
Human player score:    31
Computer player score: 136

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  . {12 .  . (2) .  .  . S1  .  .  . (2) .  . {8}  1
 2  .  . {2} .  .  . (3) . T1  . (3) .  . G2 {2} .   2
 3  .  .  . {2} .  .  . (2)E1 (2) .  .  . U1  .  .   3
 4  . (2) .  . {2} .  .  . L1  .  . D2 A1 R1 E1 (2)  4
 5  .  .  .  .  . {2} .  . L1  .  . {0} . N1  .  .   5
 6  .  . (3) .  .  . (3)D2 A1 N1 C3 E1  . a3 (3) .   6
 7  .  .  . (2) .  .  . I1  . (2) .  .  . R1  .  .   7
 8 (3){-3 .  . (2) . U1 N1 P3 I1 t3 I1 E1 D2  . {-2  8
 9  .  .  . (2) .  .  . T1  . (2) .  .  . (2) .  .   9
10  .  . (3) .  .  . (3) .  .  . (3) .  .  . (3) .  10
11  .  .  .  .  . {2} .  .  .  .  . {2} .  .  .  .  11
12  . (2) .  . {-1 .  .  . (0) .  .  . {-1 .  . (2) 12
13  .  .  . {2} .  .  . (2) . (9) .  .  . {2} .  .  13
14  .  . {2} .  .  . (3) .  .  . (3) .  .  . {2} .  14
15  . {9} .  . (2) .  .  . {3} .  .  . (2) .  . {16 15
16  .  .  .  .  .  .  .  . (3) .  .  .  .  .  .  .  16

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

OPEN GAME: The computer's tiles:
OPEN GAME: [R1], [G2], [P3], [E1], [O1], [O1], [A1]
It's your turn! Your tiles:
[U1], [I1], [S1], [I1], [O1], [T1], [Y4]
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles
and lower-case letters are wildcards.
Entering "," passes the turn.
O,j5
The board does not permit word O at position j5. Please try again.
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles
and lower-case letters are wildcards.
Entering "," passes the turn.
O,5j
The board does not permit word O at position 5j. Please try again.
Please enter your move in the format: "word,square" (without the quotes)
```

```
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles
and lower-case letters are wildcards.
Entering "," passes the turn.
SC,k5
With tiles [U1], [I1], [S1], [I1], [O1], [T1], [Y4] you cannot play word SC,k5
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles
and lower-case letters are wildcards.
Entering "," passes the turn.
SO,k5
The move is:    Word: SO at position k5
Human player score:    39
Computer player score: 136

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

 1  . {12 .  . (2) .  .  . S1  .  .  . (2) .  . {8}  1
 2  .  . {2} .  .  . (3) . T1  . (3) .  . G2 {2} .   2
 3  .  .  . {2} .  .  . (2)E1 (2) .  .  . U1  .  .   3
 4  . (2) .  . {2} .  .  . L1  .  . D2 A1 R1 E1 (2)  4
 5  .  .  .  .  . {2} .  . L1  . S1 {0} . N1  .  .   5
 6  .  . (3) .  .  . (3)D2 A1 N1 C3 E1  . a3 (3) .   6
 7  .  .  . (2) .  .  . I1  . (2)O1  .  . R1  .  .   7
 8 (3){-3 .  . (2) . U1 N1 P3 I1 t3 I1 E1 D2  . {-2  8
 9  .  .  . (2) .  .  . T1  . (2) .  .  . (2) .  .   9
10  .  . (3) .  .  . (3) .  .  . (3) .  .  . (3) .  10
11  .  .  .  .  . {2} .  .  .  .  . {2} .  .  .  .  11
12  . (2) .  . {-1 .  .  . (0) .  .  . {-1 .  . (2) 12
13  .  .  . {2} .  .  . (2) . (9) .  .  . {2} .  .  13
14  .  . {2} .  .  . (3) .  .  . (3) .  .  . {2} .  14
15  . {9} .  . (2) .  .  . {3} .  .  . (2) .  . {16 15
16  .  .  .  .  .  .  .  . (3) .  .  .  .  .  .  .  16

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

OPEN GAME: The computer's tiles:
OPEN GAME: [R1], [G2], [P3], [E1], [O1], [O1], [A1]
It's the computer's turn!
The move is:    Word: POGOER at position 1c
Human player score:    39
Computer player score: 148

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p
 1  . {12P3 O1 G2 O1 E1 R1 S1  .  .  . (2) .  . {8}  1
 2  .  . {2} .  .  . (3) . T1  . (3) .  . G2 {2} .   2
 3  .  .  . {2} .  .  . (2)E1 (2) .  .  . U1  .  .   3
 4  . (2) .  . {2} .  .  . L1  .  . D2 A1 R1 E1 (2)  4
 5  .  .  .  .  . {2} .  . L1  . S1 {0} . N1  .  .   5
 6  .  . (3) .  .  . (3)D2 A1 N1 C3 E1  . a3 (3) .   6
 7  .  .  . (2) .  .  . I1  . (2)O1  .  . R1  .  .   7
 8 (3){-3 .  . (2) . U1 N1 P3 I1 t3 I1 E1 D2  . {-2  8
 9  .  .  . (2) .  .  . T1  . (2) .  .  . (2) .  .   9
10  .  . (3) .  .  . (3) .  .  . (3) .  .  . (3) .  10
11  .  .  .  .  . {2} .  .  .  .  . {2} .  .  .  .  11
12  . (2) .  . {-1 .  .  . (0) .  .  . {-1 .  . (2) 12
13  .  .  . {2} .  .  . (2) . (9) .  .  . {2} .  .  13
14  .  . {2} .  .  . (3) .  .  . (3) .  .  . {2} .  14
15  . {9} .  . (2) .  .  . {3} .  .  . (2) .  . {16 15
16  .  .  .  .  .  .  .  . (3) .  .  .  .  .  .  .  16

    a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p

OPEN GAME: The computer's tiles:
OPEN GAME: [A1], [R1], [A1], [H4], [X8], [V4], [A1]
It's your turn! Your tiles:
[U1], [I1], [I1], [T1], [Y4], [U1], [W4]
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles
and lower-case letters are wildcards.
Entering "," passes the turn.
```

The example play still goes on for a while until one of the players has emptied their tile rack. Look at the play carefully, it can help disambiguate special cases of the specification!

Note specifically that in SkraBBKle, a move may never lead to more than one word of 2 or more letters being created on the board. (This is different from Scrabble.) The created word must always be in the same direction as the player's move and use all tiles of the move. This is why the moves `O,j5` and `O,5j` were both not allowed.

Board files

We need to define a file format for SkraBBKle boards that a user may write with an external text editor. We use a plain-text format.

The **first line** of a file for a SkraBBKle board stores a integer number S between 12 and 26. This number indicates the size of our `S x S` SkraBBKle board.

After the first line, there are S further lines. Each of these lines consists of exactly S of the following "tokens", each of which represents a square:

- `.` represents a standard square without premium.

- `(n)` represents a letter premium square, where `n` is an integer between `-9` and `99`.
- `{n}` represents a word premium square, where `n` is an integer between `-9` and `99`.

Note that in our file format, a premium square may need either 3 or 4 characters. Our file format does not allow for spaces.

See the file `resources/defaultBoard.txt` for an example. A file is *valid* if it is syntactically correct as specified above.

Moves

As indicated above, you will need to read *moves* for the human player.

The *pass move* is indicated by writing a line that consists of just one comma: `,`

A move to *play tiles* is indicated by writing the word spelled by the tiles that are played (excluding the tiles already on the board), then a comma, then the position with the direction in format `xy`:

- If `xy` has `x` as a column label and `y` as a row number, it means that the tiles are played downward, with `xy` as the "origin" of the move. An example is `d7` (where `x=d` and `y=7`) to indicate that the first of the played tiles should go to `d7` and the tiles should be played downward.
- If `xy` has `x` as a row number and `y` as a column label, it means that the tiles are played from left to right, with `yx` as the "origin" of the move. An example is `3g` (where `x=3` and `y=g`) to indicate that the first of the played tiles should go to `g3` and the tiles should be played from left to right.

For example, `OND,f8` says that from the position in column `f` and row `8` on the board, going down, a tile sequence corresponding to the word `OND` should be played, where occupied squares are skipped. Another example is `VLuE,10b`, which says that from position `b10`, going right, a tile sequence corresponding to the word `VLUE` should be played, where a wildcard should be used for the `U` on the board (indicated by the use of the lower-case letter `u`).

User requirements

> **Note: the requirements below are mandatory to follow. You will lose marks if your implementation does not meet these requirements**

In this coursework, you shall implement a Java program in which a human user shall play SkraBBKle described above against the computer. **The human always makes the first move.**

Initiation

When the program is executed, it first prompts the user to provide a file name that stores a plain board configuration:

```
Would you like to _l_oad a board or use the _d_efault board?
Please enter your choice (l/d):
```

The user inputs `l` or `d`. Otherwise, the program states this and asks the user again for input. If the user inputs `d`, the default board is loaded from the file `resources/defaultBoard.txt`. If the user inputs `l`, the program asks for the file name:

```
Please enter the file name of the board:
```

Then the user enters the file name. If this file is not valid, the program states that and prompts to provide a file name again:

```
This is not a valid file. Please enter the file name of the board:
```

This continues until the user provides a valid file. If a valid file is provided, the board that it contains is used for the game.

Finally, the user is asked whether to play an open game or a closed game:

```
Would you like to play an _o_pen or a _c_losed game?
Please enter your choice (o/c):
```

The user inputs o or c. Otherwise, the program states this and asks the user again for input. If the user inputs o, an open game will be played, and if the user inputs c, a closed game will be played. The only difference between an open game and a closed game is that before every move, the tiles on the computer player's rack are displayed (the lines starting with `OPEN GAME:` in the example play).

After these initial questions to the user, the game begins.

Play moves

As we observe in the example play, the human player and the computer player take turns with their moves. The human player makes the first move.

Before each move, the program prints the current score and the current board.

If it is the human player's turn, the program asks the human player for their move:

```
Illegal move format
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles
and lower-case letters are wildcards.
Entering "," passes the turn.
```

If the entered move does not follow the requested format, the program states this and asks again for input of a move:

```
Illegal move format
Please enter your move in the format: "word,square" (without the quotes)
For example, for suitable tile rack and board configuration, a downward move
could be "HI,f4" and a rightward move could be "HI,4f".

In the word, upper-case letters are standard tiles
and lower-case letters are wildcards.
Entering "," passes the turn.
```

This continues until the user inputs a valid move. Then the updated board and score are displayed.

When it is the computer player's turn, the move is computed by the program based on the tiles in the computer player's tile rack and the current state of the board. The move is then printed, and the updated board and score are displayed.

If the condition for the end of the game is reached (see Rules above), this is stated, the final scores of both players are calculated and shown, and the winner is announced.

```
Game Over!
The human player scored 216 points.
The computer player scored 203 points.
The human player wins!
```

(scores may vary). Instead of

```
The human player wins!
```

it may also read

```
The computer player wins!
```

or

```
It's a draw!
```

Then the program terminates.

Computer player

Implement the computer player in such a way that it will always make a move with at least one tile whenever such a move is possible with the given tile rack, board state, and word list.

Hint: A possible strategy to achieve this goal could be the following:

1. For each number n of tiles that could be played from the current tile rack (between at most the number of tiles on the rack and at least 1), for each free position on the board and for both directions, check whether it is possible to play n tiles there. (Often enough, the answer will be `false`, e.g., because no connection to the existing crossword on the board would be made. There is no need to check all the possible combinations of n tiles for a position and direction where they cannot be played anyway!)
2. For those cases of part (1) where it is indeed possible to play n tiles, go through all combinations of n tiles on your rack. If one of them leads to a valid move for the board and the word list, make the move.

Hint: Consider implementing first the other parts of the coursework project. While you are getting up and running, a computer player that always passes their move does the job.

Implementation requirements

- The requirements below are mandatory to follow. **You will lose marks** if your implementation does not adhere to these requirements, **even in case your program runs with no errors**.
- Your implementation shall be in **Java 17**.
- The class from which the program is run shall be called `Main.java` and be in the package `pij.main`.
- Your program design shall employ object-oriented principles with suitable definitions of classes/interfaces and using packages, encapsulation, polymorphism, and inheritance as appropriate for this case.

Validation

Your submission shall contain 20 (or more) JUnit 5 test cases that test the functionality of your implementation. The test cases should cover a broad range of scenarios.

Development and Submission

Your code shall be well-structured in terms of visual readability (use indentation, spacing, etc.) and non-redundancy (instead of duplicating long pieces of code, introduce your own methods). Follow sensible naming conventions. Your code shall be well documented.

You shall accept this coursework assignment via its invitation link, which is available behind the *Assessment* tile of the Moodle website for the *Programming in Java* module. When you accept the coursework assignment, your personal *GitHub* repository with the initial files for the coursework assignment will be set up for you so that you can use it with your GitHub account. It is up to you whether you create a new GitHub account for this assignment or use an existing GitHub account.

You are expected to work on the coursework in your assigned *GitHub* repository. You must create commits, with messages, that describe the history of the development of your project. We expect **a rich commit history**. *You will lose marks if you do not meet this requirement*. The use of other features of git, e.g., branches, is optional and does not contribute to your mark.

Information about the release date and the submission deadline for this coursework assignment is available behind the *Assessment* tile of the Moodle website for the *Programming in Java* module.

Academic declaration

The file `README.md` shall contain your name in the first section. In the second section, you shall provide the following academic declaration:

“I have read and understood the sections of plagiarism in the College Policy on academic integrity and misconduct and confirm that the work is my own, with the work of others clearly acknowledged. I give my permission to submit my report to the plagiarism testing database that the College is using and test it using plagiarism detection software, search engines or meta-searching software.”

This refers to the following document: College Policy on academic integrity and misconduct

A submission without this declaration shall receive 0 marks.

The other sections (from “Description of the coursework: SkraBBKle” onward) shall not be modified.

Hint: Cloning your repository from GitHub to your computer, then entering your name and the academic declaration into `README.md`, and then committing and pushing your changes to GitHub could be a wonderful way to start your coursework assignment and check whether your git set-up works as intended.

Submission

The files of the final version of your project must be submitted via GitHub ***and*** via Moodle. We require at least the following in your submission:

- `README.md` with your name and your academic declaration added in the first two sections
- `src/pij/main/Main.java`
- `resources/defaultBoard.txt`
- `resources/wordlist.txt`

and any other files or directories that are needed for the program started from class `Main` in package `pij.main` to run, as well as the JUnit tests.

> Note: by the deadline of coursework submission, you must have both:
>
> - the GitHub repo with the final version of your coursework (and history)
> - the code of the project on Moodle
>
> Both entries contribute to your mark.

Additional Libraries

You can use *any standard Java libraries that are part of Java 17* in your implementation. Additionally, you will need the libraries for JUnit 5. Otherwise, no libraries are allowed.

Marking

We aim to determine your mark according to the following rubric:

Category	Weight	full	¾	½	¼	0
User requirements (*completeness* = system responds to all use scenarios; *correctness* = all responses are as expected)	40%	Correct and complete	Correct and mostly complete	Mostly correct and mostly complete	Either mostly incorrect and mostly complete, or mostly incomplete and mostly correct	Either fully incorrect, or fully incomplete, or mostly incorrect and mostly incomplete
Implementation requirements (*completeness* = a sensible design has been chosen; *correctness* = its implementation is correct	30%	Correct and complete	Correct and mostly complete	Mostly correct and mostly complete	Either mostly incorrect and mostly complete, or mostly incomplete and mostly correct	Either fully incorrect, or fully incomplete, or mostly incorrect and mostly incomplete

Category	Weight	full	¾	½	¼	0
Validation (*completeness* = appropriate variety of scenarios are verified; *correctness* = tests for them are correctly written)	10%	Correct and complete	Correct and mostly complete	Mostly correct and mostly complete	Either mostly incorrect and mostly complete, or mostly incomplete and mostly correct	Either fully incorrect, or fully incomplete, or mostly incorrect and mostly incomplete
Development style (*well-structured code* = visual readability and non-redundancy (see Software Specification); *commit history* = rich commit history (see Development and Submission)	20%	Well-structured code and a rich commit history	Well-structured code and a moderately rich commit history	Nearly well-structured code and a moderately rich commit history	Not well-structured code or poor commit history	Not well-structured code and poor commit history

Credits

The file `resources/wordlist.txt` is a slightly abridged version of the SOWPODS word list that is available for download here.