Project 3 – Word Puzzle

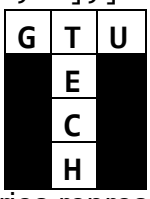
In this project, you are asked to implement functions that will enable you (and others) to play the game of word placement puzzle.

The objective of this game is to form the board and fill the empty cells with the letters of the given words. Usually puzzle is completed when all words are used.

Rules:

- You will implement each of the following functions **with exact names, parameters and default values if given** in your python file.
- Each of your functions should return the expected result in the given form.
- You will be graded for each function.
- Each board structure will be a 2D list (list of lists) in the form of: `[ ['G', 'T', 'U'], [0, 'E', 0], ... ]` describing a `row x column` matrix.
  - For example, a 4x3 board is passed as `[['G', 'T', 'U'], [0, 'E', 0], [0, 'C', 0], [0, 'H', 0],]`. This represents the actual board as:

```
G  T  U
0  E  0
0  C  0
0  H  0
```



  - Note that the `0` entries represent the forbidden boxes.
  - An empty board should consist of 0s and 1s:

Initial Board

```
1  1  1
0  1  0
0  1  0
0  1  0
```

Use GTU

1st letter

```
G  T  U
0  1  0
0  1  0
0  1  0
```

Use TECH

2nd letter

```
G  T  U
0  E  0
0  C  0
0  H  0
```

- The structure of the moves or commands should include 'W' followed by the word number, 'R' followed by the row number or first coordinate, 'C' followed by the column number or second coordinate, 'D' followed by the direction of the placement either vertical ('V') or horizontal ('H').
  - For example, 'W1R2C4DV' means take the first word, start to place it from [2,4] coordinates vertically.
  - The commands should not be case sensitive
    'W1R2C4DV' is same as 'w1r2c4dv' or 'W1r2C4dV' or 'W1R2C4Dv', etc.

  - The order can be in different forms:
    'W1R2C4DV' is same as ' R2W1C4DV' or 'dVr2C4W1' or ' C4W1DvR2', etc.
  - Do not forget the word number, column and row numbers can be two digits
    'W12R18C10DH'

- You should **create** and **mutate** the puzzle board. Your grade will depend on this. Do **NOT** copy or recreate the board unless it is specifically asked.
- The main objective of this game is to try to solve the puzzle by using a predefined set of moves, specifically mutating it to its final form. The final form is the board state after applying the given set of moves.
- For a given board, **the solution** (the solved form) is achieved if all the words are placed correctly to the puzzle board.
- Board size can go as high as `20x20` and as low as `2x2`, but this should not limit you or change anything in your implementation.
- Words and board are given with a `txt` file. The lines of the file starts with words, then an empty line followed by the board structure. The board structure consists of `0` and `1`. For example:

```
GTU
TECH

111
010
010
010
```

  - Note that there is no any empty line at the end of the file.
  - Following the file structure above, you can create your own puzzles for testing.

Function Name	**Explanation**
`read_file(filename="test_puzzle.txt")`	This function will take one parameter as `string` for filename. Default value of the input is `'test_puzzle.txt'`. This function opens the file, reads the words and board entries, and assign them different lists. Input: ● `filename : string` (default = `'test_puzzle.txt'`) Return word list as a list, and board as a list of strings. Return: ● `output1 :` list of strings for words ● `output2 :` list of strings for board

`check_consistency(board)`	This function will take one parameter as `list` for board as list of strings. Checks the all rows and columns are in same size. Input: ● `board :` list of strings Returns `boolean`: `True` if the board is consistent, `False` if the board is not consistent. Return: ● `output : boolean`
`create_board(board)`	This function will take one parameter as `list` for board as list of strings. Mutates the board to list of lists, as rows and columns of the board. Input: ● `board :` list of strings Return `None`.
`identifier(words)`	This function will take one parameter as `list` for words as list of strings. Creates a list with the same size of input, initialize it with `False`. Input: ● `words :` list of strings Return `list` with `boolean` entries. Return: ● `output :` a list with same size of input list
`print_board(board)` Create_board degiskeninin parametresini degistirmek zorunda kaldik.	This function will take one parameter as `list` for board as list of lists. Prints the board with spaces to see the current state of the board. Prints `'+'` character for the forbidden spaces. Input: ● `board :` list of lists Return `None`.
`print_board_w_c(board)`	This function will take one parameter as `list` for board as list of lists. Prints the board with spaces and coordinates to see the current state of the board. Rows should be specified with `R` and columns should be specified with `C`. Number of rows and columns should start from 1, not 0. Prints `'+'` character for the forbidden

	spaces. Input: ● `board` : list of lists Return `None`.
`print_wordlist(words,wstatus)`	This function will take two parameters as `list` for words as list of strings and `list` for states of the words as used or not. Prints the words with word number and currents states as 'USED' or 'NOT USED'. Words should be specified with `W`. Number of words should start from 1, not 0. Input: ● `words` : list of strings ● `wstatus` : list of boolean Return `None`.

`check_entries(coordinates,wordno,board,words)`	This function take four parameters, coordinates as a list, i.e. `[row number, column number]`, word number as integer, board as list of lists, and words as list of strings. Checks the coordinates that points a position in the board, and word number does not exceed the word list.
	
	Inputs:
	● `coordinates` : list `[rowno,columnno]`
	`len(board)≥rowno>0`
	`len(board[0])≥columnno>0`
	
	● `wordno` : integer
	`len(words)≥wordno>0`
	
	● `board` : list of lists
	
	● `words` : list of strings
	
	Return
	● `output1` : boolean
	If coordinates are not valid returns `False`, else `True`.
	● `output2` : boolean
	If word number is not valid returns `False`, else `True`.

`check_location(board,words,coordinates,wordno,direction='H')`	This function takes five parameters. Check the location pointed by the `coordinates` and word pointed by the `wordno`, and checks the cells with the following order, returns boolean and integer for the problem flag:

	• Check if the starting cell is a forbidden cell, then return `False,1`
	• Check if direction is vertical and the upper cell (if any) is not a forbidden cell, then return `False,2`
	• Check if direction is horizontal and the cell at the left (if any) is not a forbidden cell, then return `False,3`
	• Check, from starting coordinates, if the word exceeds the board, then return
	`False,4` for horizontal direction
	`False,7` for vertical direction
	• Check, from starting coordinates, if the word length does not fit to the space, then return
	`False,5` for horizontal direction
	`False,8` for vertical direction
	• Check, if the next cell (if any) of the words last cell in given direction is a not a forbidden cell, then return
	`False,6` for horizontal direction
	`False,9` for vertical direction
	
	Inputs:
	● `board :` list of lists
	● `words :` list of strings
	● `coordinates :` list `[rowno,columnno]`
	`len(board)≥rowno>0`
	`len(board[0])≥columnno>0`
	● `wordno :` integer
	`len(words)≥wordno>0`
	● `direction :` `'H'` for vertical (default)
	`'V'` for vertical
	
	Return
	● `output1 :` boolean
	If there is a problem, returns `False`, else `True`.
	● `output2 :` integer
	If `output1` is `True`, `0`,
	Else `output2` is the problem number.

`check_word_fits(board,words,coordinates,wordno,direction='H')`	This function takes five parameters. Check the location pointed by the `coordinates` and word pointed by the `wordno`, and checks the associated cells if they are empty or letters of the word are same as entries. Returns boolean

	and integer for problem flag:
	
	Inputs:
	● `board :` list of lists
	● `words :` list of strings
	● `coordinates :` list `[rowno,columnno]`
	`len(board)≥rowno>0`
	`len(board[0])≥columnno>0`
	● `wordno :` integer
	`len(words)≥wordno>0`
	● `direction :` `'H'` for vertical (default)
	`'V'` for vertical
	
	Return
	● `output1 :` boolean
	If one of the pointed cells is not empty or letter in a cell is different form the word's letters in order, returns `False`, else `True`.
	● `output2 :` integer
	If `output1` is `True`, `0`,
	Else `output2` is `1` for horizontal direction, `2` for vertical direction.

`clear_board(board,wstatus)`	This function will take two parameters as `list` for board as list of lists and `list` for word status as a list. Clear all entries by mutating the `board`, i.e. set all letter entries to `1`, and set all word status to `False` by mutating the `wstatus`. Inputs: ● `board :` list of lists ● `wstatus :` list of boolean Return `None`.
`decompose_command(str1)`	This function will take one parameter as string that represent the move or command. A proper command is explained above. The function checks if all word, row, column and direction information is specified in the input string. Then decomposes these information and returns them. If direction is not `'V'` or `'H'` then direction should be set to `'H'`. Input: ● `str1 :` string Return

	• `output1 :` integer
	`0`, if all information is available
	`-1`, if not.
	
	• `output2 :` integer or NoneType
	word number, if `output1` is `0`
	`None`, if `output1` is `-1`
	
	• `output3 :` list or NoneType
	If `output1` is `0`,`[rowno,columnno]`
	If `output1` is `-1, None`
	
	• `output3 :` string or NoneType
	If `output1` is `0`,
	`'V' for vertical,`
	`'H' for horizontal (default).`
	`(All capital letters.)`
	
	If `output1` is `-1, None`

`word_it(board,words,wstatus,coordinates,wordno,direction)`	This function will take six parameters. Checks the entries, checks the location, checks the word fits, check the word is used or not and place the word to the pointed cells by the `coordinates` by mutating the board. Returns `True` if the word placed successfully, `False` if there is a problem on checks. Inputs: • `board :` list of lists • `words :` list of strings • `wstatus :` list of boolean • `coordinates :` list `[rowno,columnno]` • `wordno :` integer • `direction :` `'H'` for vertical, `'V'` for vertical Return • `output :` boolean `True`, if word is placed successufully. `False`, if not.
`copy_board(board)`	This function will take one parameter as `list` for board as list of lists and returns the copy of the board with all entries as a list of lists.

	Inputs: ● `board` : list of lists Return ● `output` : list of lists
`track_move(mvn,trackboard,coordinates, wordno,direction,board,wstatus)`	This function will take seven parameters. This function is intended to track the move history. It appends the `coordinates`, `wordno`, `direction`, copy of the `board`, and copy of the `wstatus` to `trackboard` as a `Tuple` by mutating `trackboard`, and increases the `mvn` by `1` and returns the updated `mvn`. Inputs: ● `mvn` : integer ● `trackboard` : list ● `coordinates` : list `[rowno,columnno]` ● `wordno` : integer ● `direction` : `'H'` for vertical, `'V'` for vertical ● `board` : list of lists ● `wstatus` : list of boolean Return ● `output` : integer
`check_solved(board)`	This function will take one parameter as `list` for board as list of lists and returns `True` if the current state of the board is solved, `False` if it is not solved yet. Inputs: ● `board` : list of lists Return ● `output` : boolean `True`, if the board is solved `False`, if the board is not solved.
`solve_board(board,words)`	This function will take two parameters as `list` for board as list of lists and `list` for words as a list of strings. Solves the board by mutating the board, using the words in the `words` list. Returns `True` if it obtains the solution, else returns `False`. Inputs: ● `board` : list of lists

	● `words :` list of strings Return ● `output :` boolean `True`, if the solution is obtained, `False`, if not.
`word_puzzle()`	This function does not take any parameters. This function is the main function to play the word placement puzzle game. This function can be customized to yourself, but should ● ask for filename for the puzzle, ● perform proper checks on board, ● prints the words and board, ● ask for a move, ● perform proper checks on move, ● if there is problem print out the associated problem and ask to re-enter the move with quit and solve the board options, ● finalize the code with proper message. Optional : You can add move back option.

An example game

```
>>> words,board=read_file("test_puzzle1.txt")
>>> print(words)
['GTU', 'TECH']
>>> print(board)
['111', '010', '01', '010']
>>> print(check_consistency(board))
False
>>> words,board=read_file("test_puzzle.txt")
>>> print(words)
['GTU', 'TECH']
>>> print(board)
['111', '010', '010', '010']
>>> print(check_consistency(board))
True
>>> create_board(board)
>>> print_board(board)

+   +
+   +
+   +
```

```
>>> print_board_w_c(board)
  C1 C2 C3
R1
R2 +     +
R3 +     +
R4 +     +

>>> wstatus=identifier(words)
>>> print(wstatus)
[False, False]
>>> print_wordlist(words,wstatus)
Word List            Status
W1 GTU               NOT USED
W2 TECH              NOT USED

>>> print(check_solved(board))
False
>>> cmd1='w1r2c3dk'
>>> print(decompose_command(cmd1))
(0, 1, [2, 3], 'H')
>>> iflag,wordno,coordinates,direction=decompose_command(cmd1)
>>> print(iflag,wordno,coordinates,direction)
0 1 [2, 3] H
>>> print(check_entries(coordinates,wordno,board,words))
(True, True)
>>> print(check_location(board,words,coordinates,wordno,direction))
#Coordinate does not point a valid cell!
(False, 1)
>>> cmd2='W1r1c1'
>>> print(decompose_command(cmd2))
(-1, None, None, None)
>>> cmd3='W2DHR1C1'
>>> iflag,wordno,coordinates,direction=decompose_command(cmd3)
>>> print(iflag,wordno,coordinates,direction)
0 2 [1, 1] H
>>> print(check_entries(coordinates,wordno,board,words))
(True, True)
>>> print(check_location(board,words,coordinates,wordno,direction))
#Word exceeds the board!
(False, 4)
>>> cmd4='dVW2R1C2'
>>> iflag,wordno,coordinates,direction=decompose_command(cmd4)
>>> print(iflag,wordno,coordinates,direction)
0 2 [1, 2] V
>>> print(check_entries(coordinates,wordno,board,words))
(True, True)
```

```
>>> print(check_location(board,words,coordinates,wordno,direction))
(True, 0)
>>> print(check_word_fits(board,words,coordinates,wordno,direction))
(True, 0)
>>> print(word_it(board,words,wstatus,coordinates,wordno,direction))
True
>>> print_board_w_c(board)
  C1 C2 C3
R1    T
R2 +  E  +
R3 +  C  +
R4 +  H  +

>>> print_wordlist(words,wstatus)
Word List            Status
W1 GTU               NOT USED
W2 TECH              USED

>>> moveno=0
>>> trackmoves=[]
>>>
moveno=track_move(moveno,trackmoves,coordinates,wordno,direction,board,wstatus)
>>> print(moveno)
1
>>> print(trackmoves)
[([1, 2], 2, 'V', [[1, 'T', 1], [0, 'E', 0], [0, 'C', 0], [0, 'H', 0]], [False,
True])]
>>> print(check_solved(board))
False
>>> copyofboard=copy_board(board)
>>> cmd5='W1R1C1Dh'
>>> iflag,wordno,coordinates,direction=decompose_command(cmd5)
>>> print(iflag,wordno,coordinates,direction)
0 1 [1, 1] H
>>> print(word_it(board,words,wstatus,coordinates,wordno,direction))
True
>>> print_board_w_c(board)
  C1 C2 C3
R1 G  T  U
R2 +  E  +
R3 +  C  +
R4 +  H  +

>>> print_wordlist(words,wstatus)
Word List            Status
W1 GTU               USED
```

```
W2 TECH              USED

>>>
moveno=track_move(moveno,trackmoves,coordinates,wordno,direction,board,wstatus)
>>> print(moveno)
2
>>> print(trackmoves)
[([1, 2], 2, 'V', [[1, 'T', 1], [0, 'E', 0], [0, 'C', 0], [0, 'H', 0]], [False,
True]), ([1, 1], 1, 'H', [['G', 'T', 'U'], [0, 'E', 0], [0, 'C', 0], [0, 'H',
0]], [True, True])]
>>> print(check_solved(board))
True
>>> print('Board is solved!')
Board is solved!
>>> print_board_w_c(board)
  C1 C2 C3
R1 G  T  U
R2 +  E  +
R3 +  C  +
R4 +  H  +

>>> print_board_w_c(copyofboard)
  C1 C2 C3
R1    T
R2 +  E  +
R3 +  C  +
R4 +  H  +

>>> clear_board(board,wstatus)
>>> print_board_w_c(board)
  C1 C2 C3
R1
R2 +     +
R3 +     +
R4 +     +

>>> print_wordlist(words,wstatus)
Word List            Status
W1 GTU               NOT USED
W2 TECH              NOT USED

>>> print(solve_board(board, words))

Found the solution 1

True
```

```
>>> print_board_w_c(board)
  C1 C2 C3
R1 G  T  U
R2 +  E  +
R3 +  C  +
R4 +  H  +


>>> word_puzzle()
Game starts
Enter the file name (default=sample_puzzle.txt):
test_puzzle1.txt
The puzzle board of test_puzzle1.txt is not consistent!
Do you want to try a new file?
y:yes, n:no:y
Enter the file name (default=sample_puzzle.txt):
test_puzzle.txt

Word List            Status
W1 GTU               NOT USED
W2 TECH              NOT USED

  C1 C2 C3
R1
R2 +     +
R3 +     +
R4 +     +

Please choose a word from word list,
       choose a row and a column, and
       direction of the word, V: Vertical H:Horizontal
       enter your move as in the format: (WXRYCZDT)
Other options:
 cb - clear board
 q - quit game
 s - solve puzzle
W1C2r1dv
Word does not fit to the space vertically!
Do you want to try a new move?
y:yes, n:no:y

Word List            Status
W1 GTU               NOT USED
W2 TECH              NOT USED

  C1 C2 C3
```

```
R1
R2 +     +
R3 +     +
R4 +     +

Please choose a word from word list,
       choose a row and a column, and
       direction of the word, V: Vertical H:Horizontal
       enter your move as in the format: (WXRYCZDT)
Other options:
 cb - clear board
 q - quit game
 s - solve puzzle
w1r2c3dv
Coordinate does not point a valid cell!
Do you want to try a new move?
y:yes, n:no:y

Word List            Status
W1 GTU               NOT USED
W2 TECH              NOT USED

  C1 C2 C3
R1
R2 +     +
R3 +     +
R4 +     +

Please choose a word from word list,
       choose a row and a column, and
       direction of the word, V: Vertical H:Horizontal
       enter your move as in the format: (WXRYCZDT)
Other options:
 cb - clear board
 q - quit game
 s - solve puzzle
w1r1c1dh

Word List            Status
W1 GTU               USED
W2 TECH              NOT USED

  C1 C2 C3
R1 G  T  U
R2 +     +
R3 +     +
```

```
R4 +     +

Please choose a word from word list,
       choose a row and a column, and
       direction of the word, V: Vertical H:Horizontal
       enter your move as in the format: (WXRYCZDT)
Other options:
 cb - clear board
 q - quit game
 s - solve puzzle
cb
Restarting the game!

Word List            Status
W1 GTU               NOT USED
W2 TECH              NOT USED

  C1 C2 C3
R1
R2 +     +
R3 +     +
R4 +     +

Please choose a word from word list,
       choose a row and a column, and
       direction of the word, V: Vertical H:Horizontal
       enter your move as in the format: (WXRYCZDT)
Other options:
 cb - clear board
 q - quit game
 s - solve puzzle
w1r1c1dh

Word List            Status
W1 GTU               USED
W2 TECH              NOT USED

  C1 C2 C3
R1 G  T  U
R2 +     +
R3 +     +
R4 +     +

Please choose a word from word list,
       choose a row and a column, and
       direction of the word, V: Vertical H:Horizontal
```

```
       enter your move as in the format: (WXRYCZDT)
Other options:
 cb - clear board
 q - quit game
 s - solve puzzle
w2c2r2dv
Coordinate does not start form a valid cell!
Do you want to try a new move?
y:yes, n:no:y

Word List            Status
W1 GTU               USED
W2 TECH              NOT USED

  C1 C2 C3
R1 G  T  U
R2 +     +
R3 +     +
R4 +     +

Please choose a word from word list,
       choose a row and a column, and
       direction of the word, V: Vertical H:Horizontal
       enter your move as in the format: (WXRYCZDT)
Other options:
 cb - clear board
 q - quit game
 s - solve puzzle
w2c2r1dv

Congratulations!!!
PUZZLE SOLVED

Word List            Status
W1 GTU               USED
W2 TECH              USED

  C1 C2 C3
R1 G  T  U
R2 +  E  +
R3 +  C  +
R4 +  H  +

Would you like to play again?
y:yes, n:no:y
Enter the file name (default=sample_puzzle.txt):
```

```
Word List            Status
W1  ANSWER           NOT USED
W2  DESK             NOT USED
W3  DONE             NOT USED
W4  GLUE             NOT USED
W5  LEARN            NOT USED
W6  STUDY            NOT USED
W7  TAPE             NOT USED
W8  TEACH            NOT USED
W9  TEST             NOT USED
W10 WORK             NOT USED

   C1 C2 C3 C4 C5 C6 C7 C8 C9
R1  +  +                 +  +
R2  +  +  +  +     +  +  +  +
R3  +  +  +  +
R4  +  +  +  +     +  +     +
R5  +     +  +     +  +     +
R6  +                +     +
R7  +     +  +  +  +  +  +  +
R8             +     +  +  +
R9  +  +  +     +     +  +  +
R10 +  +  +     +     +  +  +
R11 +  +              +  +  +

Please choose a word from word list,
       choose a row and a column, and
       direction of the word, V: Vertical H:Horizontal
       enter your move as in the format: (WXRYCZDT)
Other options:
 cb - clear board
 q - quit game
 s - solve puzzle
w1r1c5dv

Word List            Status
W1  ANSWER           USED
W2  DESK             NOT USED
W3  DONE             NOT USED
W4  GLUE             NOT USED
W5  LEARN            NOT USED
W6  STUDY            NOT USED
W7  TAPE             NOT USED
W8  TEACH            NOT USED
W9  TEST             NOT USED
```

```
W10 WORK             NOT USED

   C1 C2 C3 C4 C5 C6 C7 C8 C9
R1  +  +        A        +  +
R2  +  +  +  +  N  +  +  +  +
R3  +  +  +  +  S
R4  +  +  +  +  W  +  +     +
R5  +     +  +  E  +  +     +
R6  +           R     +     +
R7  +     +  +  +  +  +  +  +
R8             +     +  +  +
R9  +  +  +     +     +  +  +
R10 +  +  +     +     +  +  +
R11 +  +              +  +  +

Please choose a word from word list,
       choose a row and a column, and
       direction of the word, V: Vertical H:Horizontal
       enter your move as in the format: (WXRYCZDT)
Other options:
 cb - clear board
 q - quit game
 s - solve puzzle
w6c3r1dh
Word does not fit to the space!
Word does not fit to the space!
Do you want to try a new move?
y:yes, n:no:y

Word List            Status
W1  ANSWER           USED
W2  DESK             NOT USED
W3  DONE             NOT USED
W4  GLUE             NOT USED
W5  LEARN            NOT USED
W6  STUDY            NOT USED
W7  TAPE             NOT USED
W8  TEACH            NOT USED
W9  TEST             NOT USED
W10 WORK             NOT USED

   C1 C2 C3 C4 C5 C6 C7 C8 C9
R1  +  +        A        +  +
R2  +  +  +  +  N  +  +  +  +
R3  +  +  +  +  S
R4  +  +  +  +  W  +  +     +
```

```
R5  +     +  +  E  +  +     +
R6  +           R     +     +
R7  +     +  +  +  +  +  +  +
R8              +     +  +  +
R9  +  +  +     +     +  +  +
R10 +  +  +     +     +  +  +
R11 +  +              +  +  +

Please choose a word from word list,
       choose a row and a column, and
       direction of the word, V: Vertical H:Horizontal
       enter your move as in the format: (WXRYCZDT)
Other options:
 cb - clear board
 q - quit game
 s - solve puzzle
w5c3r1dh

Word List            Status
W1  ANSWER           USED
W2  DESK             NOT USED
W3  DONE             NOT USED
W4  GLUE             NOT USED
W5  LEARN            USED
W6  STUDY            NOT USED
W7  TAPE             NOT USED
W8  TEACH            NOT USED
W9  TEST             NOT USED
W10 WORK             NOT USED

   C1 C2 C3 C4 C5 C6 C7 C8 C9
R1  +  +  L  E  A  R  N  +  +
R2  +  +  +  +  N  +  +  +  +
R3  +  +  +  +  S
R4  +  +  +  +  W  +  +     +
R5  +     +  +  E  +  +     +
R6  +           R     +     +
R7  +     +  +  +  +  +  +  +
R8              +     +  +  +
R9  +  +  +     +     +  +  +
R10 +  +  +     +     +  +  +
R11 +  +              +  +  +

Please choose a word from word list,
       choose a row and a column, and
       direction of the word, V: Vertical H:Horizontal
```

```
       enter your move as in the format: (WXRYCZDT)
Other options:
 cb - clear board
 q - quit game
 s - solve puzzle
s
Trying to solve the puzzle!!!
This will take some time if number of words is larger than 10!!!!
Found the solution 86283
   C1 C2 C3 C4 C5 C6 C7 C8 C9
R1  +  +  T  E  A  C  H  +  +
R2  +  +  +  +  N  +  +  +  +
R3  +  +  +  +  S  T  U  D  Y
R4  +  +  +  +  W  +  +  O  +
R5  +  G  +  +  E  +  +  N  +
R6  +  L  E  A  R  N  +  E  +
R7  +  U  +  +  +  +  +  +  +
R8  T  E  S  T  +  W  +  +  +
R9  +  +  +  A  +  O  +  +  +
R10 +  +  +  P  +  R  +  +  +
R11 +  +  D  E  S  K  +  +  +

Quitting the game!
Done!
>>>
```