Contents

1	Basic Test Results	2
2	AUTHORS	5
3	apple.py	6
4	board.py	7
5	common.py	10
6	results.py	13
7	snake.py	28
8	snake game.py	30
9	snake main.py	33
10	wall.py	35

1 Basic Test Results

```
Tue 17 Jan 2023 23:41:14 IST
               Tue 17 Jan 2023 23:41:14 IST
               Archive: /tmp/bodek.mydalp9k/intro2cs1/ex10r1/mallis/final/submission
  4
                     inflating: src/apple.py
                  extracting: src/AUTHORS
                     inflating: src/board.py
                     inflating: src/common.py
  8
                     inflating: src/results.py
                     inflating: src/snake.py
  9
                     inflating: src/snake_game.py
                     inflating: src/snake_main.py
11
12
                     inflating: src/wall.py
               12 passed tests out of 12 in test set named 'presubmit'.
               result_code presubmit
                                                                                                      12
14
15
               Round 149: Error:
               Game state differs from expected.
16
               Last round:
                                                                           (24, {(34, 5): 'black', (34, 6): 'black', (34, 7): 'black', (34, 8): 'black', (33, 8):
17
                'black', (32, 8): 'black', (31, 8): 'black', (30, 8): 'black', (29, 8): 'black', (28, 8): 'black', (27, 8):
               'black', (26, 8): 'black', (25, 8): 'black', (24, 8): 'black', (23, 8): 'black', (22, 8): 'black', (21, 8):
19
               'black', (20, 8): 'black', (19, 8): 'black', (18, 8): 'black', (17, 8): 'black', (16, 8): 'black', (15, 8): 'black', (14, 8): 'black', (34, 4): 'green', (9, 0): 'green', (36, 14): 'green', (23, 7): 'green', (36, 27):
20
21
                 'green', (36, 17): 'green', (31, 2): 'green', (13, 18): 'blue', (12, 18): 'blue', (11, 18): 'blue', (34, 22):
22
               'blue', (33, 22): 'blue', (32, 22): 'blue', (23, 24): 'blue', (23, 25): 'blue', (23, 26): 'blue', (6, 0):
23
                'blue', (6, 1): 'blue', (6, 2): 'blue', (0, 27): 'blue', (0, 26): 'blue', (0, 25): 'blue'})
24
               Current expected: (28, {(34, 4): 'black', (34, 5): 'black', (34, 6): 'black', (34, 7): 'black', (34, 8):
25
               'black', (33, 8): 'black', (32, 8): 'black', (31, 8): 'black', (30, 8): 'black', (29, 8): 'black', (28, 8): 'black', (27, 8): 'black', (26, 8): 'black', (25, 8): 'black', (24, 8): 'black', (23, 8): 'black', (22, 8): 'black', (21, 8): 'black', (20, 8): 'black', (19, 8): 'black', (18, 8): 'black', (17, 8): 'black', (16, 8):
27
28
               'black', (15, 8): 'black', (14, 8): 'black', (9, 0): 'green', (36, 14): 'green', (23, 7): 'green', (36, 27):
               'green', (36, 17): 'green', (36, 10): 'green', (31, 2): 'green', (13, 18): 'blue', (12, 18): 'blue', (11,
30
               18): 'blue', (34, 22): 'blue', (33, 22): 'blue', (32, 22): 'blue', (23, 24): 'blue', (23, 25): 'blue',
31
               26): 'blue', (6, 0): 'blue', (6, 1): 'blue', (6, 2): 'blue', (0, 27): 'blue', (0, 26): 'blue', (0, 25):
               'blue'})
33
                                                                         (29, {(34, 4): 'black', (34, 5): 'black', (34, 6): 'black', (34, 7): 'black', (34, 8):
34
               Current actual:
                 'black', (33, 8): 'black', (32, 8): 'black', (31, 8): 'black', (30, 8): 'black', (29, 8): 'black', (28, 8):
35
               'black', (27, 8): 'black', (26, 8): 'black', (25, 8): 'black', (24, 8): 'black', (23, 8): 'black', (22, 8):
36
                'black', (21, 8): 'black', (20, 8): 'black', (19, 8): 'black', (18, 8): 'black', (17, 8): 'black', (16, 8):
37
               'black', (15, 8): 'black', (14, 8): 'black', (31, 2): 'green', (9, 0): 'green', (36, 17): 'green', (34, 22):
38
               'blue', (33, 22): 'blue', (32, 22): 'blue', (36, 14): 'green', (6, 0): 'blue', (6, 1): 'blue', (6, 2):
39
                 'blue', (0, 27): 'blue', (0, 26): 'blue', (0, 25): 'blue', (13, 18): 'blue', (12, 18): 'blue', (11, 18):
40
               'blue', (23, 7): 'green', (23, 24): 'blue', (23, 25): 'blue', (23, 26): 'blue', (36, 27): 'green', (36, 10):
41
               'green'})
42
                   -> BEGIN TEST INFORMATION
43
               Test name: moretests_10
44
               Module tested: game_display
               Function call: rungame('jjj',Namespace(width=40, height=30, apples=7, debug=False, walls=5,
46
               rounds=-1),['Right', None, 'Up',
47
               None, None, None, 'Right', 'Up', None, None, None, None, None, 'Left', None, N
               None, None, 'Up', 'Left', None, None,
49
               None, 'Down', None, None, None, None, None, None, None,
               'Right', None, 'Down', None
51
               None, 'Left',
52
               None, 'Up', None, None, None, None, None, 'Right',
               None, 
54
               None, None, None, None, None, None, None, 'Down', None, None, None, None, 'Left', None, None,
               None, None, None, None, None, None, None, None, 'Up', None, None, None, 'Left', None, None, None, None, None,
               None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, 
              None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, None, 
               None, None, None, None, None, None, None, None, None, None, None, None, None, None, 'Left', None, None,
```

```
None, None, 'Down', None, 'Right', 'Down', None, None,
             None, None, None, None, None, None, None, None, 'Left', None, None, None, None, None, None, None, None, Ellipsis)
  61
  62
             Expected return value: None
             More test options: {}
                -> END TEST INFORMATION
  64
             *************************
  65
             *******
  66
                                                                          There is a problem:
             ******
                                                                        The test named 'moretests 10' failed.
  67
  68
             ************************************
             Game state was wrong
  69
                                             moretests 10
  70
             result code
                                                                                        error 1
             Round 30: Error:
  71
             Game state differs from expected.
  72
                                                       (1, {(29, 13): 'black', (30, 13): 'black', (31, 13): 'black', (23, 4): 'green', (23, 1):
             Last round:
  73
              'green', (6, 24): 'green', (5, 25): 'green', (5, 22): 'green', (13, 14): 'green', (11, 20): 'green', (25,
  74
             25): 'green', (7, 22): 'green', (26, 29): 'green', (36, 12): 'green', (24, 4): 'green', (32, 5): 'green', (0,
  75
             10): 'green', (1, 5): 'green', (23, 2): 'green', (3, 11): 'green', (30, 7): 'green', (28, 13): 'green', (32,
             28): 'green', (2, 21): 'blue', (3, 21): 'blue', (4, 21): 'blue', (2, 5): 'blue', (3, 5): 'blue', (4, 5): 'blue', (22, 8): 'blue', (22, 9): 'blue', (22, 10): 'blue', (34, 8): 'blue', (33, 8): 'blue', (32, 8):
  77
  78
             'blue', (10, 0): 'blue', (9, 0): 'blue', (8, 0): 'blue', (13, 13): 'blue', (12, 13): 'blue', (11, 13):
             'blue', (21, 29): 'blue', (21, 28): 'blue', (26, 17): 'blue', (25, 17): 'blue', (24, 17): 'blue', (10, 22): 'blue', (10, 21): 'blue', (10, 20): 'blue', (26, 10): 'blue', (26, 9): 'blue', (26, 8): 'blue', (16, 26):
  80
  81
             'blue', (17, 26): 'blue', (18, 26): 'blue', (14, 20): 'blue', (15, 20): 'blue', (16, 20): 'blue', (12, 10):
  82
             'blue', (12, 11): 'blue', (12, 12): 'blue', (30, 26): 'blue', (29, 26): 'blue', (28, 26): 'blue', (29, 11): 'blue', (30, 11): 'blue', (31, 11): 'blue', (24, 19): 'blue', (24, 20): 'blue', (24, 21): 'blue'})
  83
  84
             Current expected: (2, {(28, 13): 'black', (29, 13): 'black', (30, 13): 'black', (31, 13): 'black', (23, 4):
  85
              'green', (23, 1): 'green', (6, 24): 'green', (5, 25): 'green', (5, 22): 'green', (13, 14): 'green', (11, 20):
  86
   87
               green', (25, 25): 'green', (7, 22): 'green', (26, 29): 'green', (36, 12): 'green', (24, 4): 'green', (32,
             5): 'green', (0, 10): 'green', (23, 2): 'green', (3, 11): 'green', (30, 7): 'green', (24, 12): 'green', (32,
  88
             28): 'green', (1, 21): 'blue', (2, 21): 'blue', (3, 21): 'blue', (1, 5): 'blue', (2, 5): 'blue', (3, 5):
  89
  90
              'blue', (22, 7): 'blue', (22, 8): 'blue', (22, 9): 'blue', (35, 8): 'blue', (34, 8): 'blue', (33, 8): 'blue',
             (11, 0): 'blue', (10, 0): 'blue', (9, 0): 'blue', (14, 13): 'blue', (13, 13): 'blue', (12, 13): 'blue', (21,
  91
             29): 'blue', (27, 17): 'blue', (26, 17): 'blue', (25, 17): 'blue', (10, 23): 'blue', (10, 22): 'blue', (10,
             21): 'blue', (26, 11): 'blue', (26, 10): 'blue', (26, 9): 'blue', (15, 26): 'blue', (16, 26): 'blue', (17, 26): 'blue', (13, 20): 'blue', (14, 20): 'blue', (15, 20): 'blue', (12, 9): 'blue', (12, 10): 'blue', (
  93
  94
             11): 'blue', (31, 26): 'blue', (30, 26): 'blue', (29, 26): 'blue', (28, 11): 'blue', (29, 11): 'blue', (30,
             11): 'blue', (24, 18): 'blue', (24, 19): 'blue', (24, 20): 'blue'})
  96
                                                       (3, {(28, 13): 'black', (29, 13): 'black', (30, 13): 'black', (31, 13): 'black', (35, 8):
  97
             Current actual:
             'blue', (34, 8): 'blue', (33, 8): 'blue', (30, 7): 'green', (22, 7): 'blue', (22, 8): 'blue', (22, 9):
             'blue', (32, 28): 'green', (1, 21): 'blue', (2, 21): 'blue', (3, 21): 'blue', (21, 29): 'blue', (11, 20): 'green', (10, 23): 'blue', (10, 22): 'blue', (10, 21): 'blue', (5, 22): 'green', (13, 20): 'blue', (14, 20):
  99
100
             'blue', (15, 20): 'blue', (26, 29): 'green', (12, 9): 'blue', (12, 10): 'blue', (12, 11): 'blue', (0, 10):
101
             'green', (31, 26): 'blue', (30, 26): 'blue', (29, 26): 'blue', (25, 25): 'green', (5, 25): 'green', (27, 17):
102
              'blue', (26, 17): 'blue', (25, 17): 'blue', (6, 24): 'green', (11, 0): 'blue', (10, 0): 'blue', (9, 0):
103
             'blue', (23, 2): 'green', (1, 5): 'blue', (2, 5): 'blue', (3, 5): 'blue', (13, 14): 'green', (36, 12):
104
             'green', (14, 13): 'blue', (13, 13): 'blue', (12, 13): 'blue', (24, 4): 'green', (26, 11): 'blue', (26, 10):
105
             'blue', (26, 9): 'blue', (7, 22): 'green', (3, 11): 'green', (23, 4): 'green', (28, 11): 'blue', (29, 11): 'blue', (30, 11): 'blue', (23, 1): 'green', (24, 18): 'blue', (24, 19): 'blue', (24, 20): 'blue', (32, 5):
106
107
             'green', (15, 26): 'blue', (16, 26): 'blue', (17, 26): 'blue', (24, 12): 'green'})
108
                -> BEGIN TEST INFORMATION
109
             Test name: moretests 18
110
             Module tested: game_display
             Function call: rungame('rrr', Namespace(width=40, height=30, apples=20, debug=False, walls=16,
112
             rounds=-1),[None, None, 
113
             None, None, None, 'Down', None, None, None, None, None, None, 'Left', None, None, None, None, None,
114
             None, 'Up', None, 'Right', None, None, None, None, None, None, None, None, None, 'Down', None, None,
115
             None, None, None, None, None, None, Veft', None, None, None, Vup', None, None, 'Right', None,
116
             None, None, None, 'Up', 'Left', None, None, None, None, None, None, 'Down', None, None, None, None, None, None,
117
             'Right', None, 'Up', None, None, None, 'Left', None, None, None, 'Up', None, None, None, None, None,
118
             None, 'Left', None, 'Up', None, 'Right', None,
119
             None, None, None, None, 'Up', None, None, None, None, None, None, None, None, 'Right', None, None, None,
120
             None, None, None, None, None, 'Down', None, None, None, None, None, None, 'Left', None, None, None,
121
             None, None, 'Down', None, None,
122
             'Right', None, None, None, 'Up', 'Left', None, None, 'Up', None, None, None, None, None, None, None, None,
123
             None, None, None, None, None, Veft', None, None, None, 'Down', None, None, None, 'Right', None, None,
124
125
              'Down', None, None, None, None, None, 'Right', None, None, None, None, None, 'Up', None, None, None,
             None, None, None, None, None, None, None, None, None, None, 'Right', None, None, None, 'Up', None, None,
126
             None, None, None, Vone, 'Left', None, None, None, None, None, None, 'Up', 'Right', None, N
```

```
128 None, 'Down', None, None, None, None, None, None, None, 'Right', None, None, 'None, 'Up', None, 'Right', None,
    None], Ellipsis)
129
130
    Expected return value: None
131 More test options: {}
    --> END TEST INFORMATION
132
134 *********************** There is a problem:
135 ************************* The test named 'moretests_18' failed.
136
    ***********************
    Game state was wrong
137
    result_code moretests_18 error
                                      1
138
    24 passed tests out of 26 in test set named 'moretests'.
139
result_code moretests 24 1
141 Listing AUTHORS...
142 dor_kaiser,mallis:AUTHORSFULL
143 TESTING COMPLETED
```

2 AUTHORS

dor_kaiser,mallis

3 apple.py

```
1
    # FILE : apple.py
   # WRITERS : Nimrod Mallis, mallis ; Dor Kaiser, dor_kaiser
   # EXERCISE : intro2cs1 ex10 2023
4
   # DESCRIPTION: Implements the Apple for the Snake Game
   # STUDENTS I DISCUSSED THE EXERCISE WITH: N/A
   # WEB PAGES I USED: N/A
    # NOTES: N/A
   9
10
   from common import BaseGameObject, Coordinate
11
12
13
    class Apple(BaseGameObject):
14
       Represents an Apple object in the Snake Game.
15
       This class is used only for matters of interaction
16
          and it implements no special functionality.
17
18
19
       def __init__(self, coordinate: Coordinate) -> None:
    """
20
21
           Creates a new apple in the given coordinate.
22
23
           Yes, it's a single coordinate, no oversized apples in this realm.
24
           super().__init__([coordinate], "green")
25
26
27
       def movement_requirements(self):
28
29
           Filler function because the interface must have this implemented.
30
           return None
31
       def move(self) -> bool:
33
34
           Filler function because the interface must have this implemented.
35
36
37
           return True
```

4 board.py

```
1
    # FILE : board.py
   # WRITERS : Nimrod Mallis, mallis ; Dor Kaiser, dor_kaiser
   # EXERCISE : intro2cs1 ex10 2023
4
    # DESCRIPTION: Implements the game board for the Snake Game
   # STUDENTS I DISCUSSED THE EXERCISE WITH: N/A
   # WEB PAGES I USED: N/A
8
    # NOTES: N/A
    9
10
    from typing import List, Callable, Optional
11
    {\tt from} \ {\tt game\_display} \ {\tt import} \ {\tt GameDisplay}
12
    from common import Coordinate, BaseGameObject, is_in_boundries, draw_coordinates
14
    # Signature for the interaction callback, first game object is the source and
15
    # the second game object is the destination
16
    InteractionCallback = Callable[[BaseGameObject, BaseGameObject], None]
17
    # Signature for the out of bounds callback, called upon interaction of an object
18
    # with the Board's boundries. The boolean parameter is a flag for whether the whole
19
20
       object went outside the boundries
21
    OutOfBoundsCallback = Callable[[BaseGameObject, bool], bool]
22
23
    class Board(object):
24
        Represents the Game Board for Snake.
25
26
        The Board houses generic game objects and is responsible for their state
27
        and interaction with the board.
28
29
30
        def __init__(self, dimensions: Coordinate) -> None:
31
            Creates a new board with the given rows {\it X} columns dimensions.
33
34
            self._dimensions: Coordinate = dimensions
            self._game_objects: List[BaseGameObject] = []
35
36
37
        def draw_board(self, gui: GameDisplay) -> None:
38
            Draws the board to the given Game Display.
39
40
            The board assumes the Game Display is of the same dimensions
            as the Board itself.
41
42
43
            priority_draw = []
            priority_color = "blue"
44
            for game_object in self._game_objects:
45
                if priority_color == game_object.get_object_color:
46
47
                   priority_draw.append(game_object.get_coordinates())
48
                    draw_coordinates(gui, self._dimensions, game_object.get_coordinates(), game_object.get_object_color())
49
50
            for coords in priority_draw:
51
                draw_coordinates(gui, self._dimensions, coords, priority_color)
52
53
        def add_game_object(self, game_object: BaseGameObject) -> bool:
54
55
            Adds the game obejct to the board
            If the object shares coordinates with another object already placed
57
58
            on the board, the method will fail and return False, True otherwise.
```

```
60
              # Making sure the object doesn't overlap with any other object first
              for coordinate in game_object.get_coordinates():
61
62
                  if self._get_object_at_coordinate(coordinate) is not None:
                      return False
 63
64
65
              self._game_objects.append(game_object)
66
              return True
67
68
         def remove_game_object(self, game_object: BaseGameObject) -> None:
69
              Removing the game object from the board, if it exists
70
71
              # If the object is not on the board, don't even try to remove it
72
73
              if game_object in self._game_objects:
 74
                  self._game_objects.remove(game_object)
75
76
         def move_game_objects(self,
                                interaction_callback: InteractionCallback,
77
                                out_of_bounds_callback: OutOfBoundsCallback) -> bool:
78
79
              Moving all game objects one after the other, then checks if any
80
              object\ interacts\ with\ another,\ if\ so,\ interaction\_callback\ is\ called.
81
              If an object interacts with the boundries of the border,
82
83
              out\_of\_bounds\_callback\ is\ called.
84
85
              interactions = {}
              out_of_bounds = []
86
87
              # Iterating on all objects and moving them
88
89
              for game_object in self._game_objects:
90
                  requirement = game_object.movement_requirements()
                  # If the game object can move and the requirements are satisfied
91
92
                  if requirement is not None and self._is_in_boundries(requirement):
93
                      destination_obj = self._get_object_at_coordinate(requirement)
94
                      # If there is a game object at the destination,
95
96
                          add to possible interactions, it will be checked later if the
                         interaction is still relevant after all objects moved
97
                      if game_object.move() and destination_obj is not None:
98
                          \#interactions.append((game\_object, destination\_obj))
99
100
                          interactions[(requirement.column, requirement.row)] = (game_object, destination_obj)
101
                  # If the game object is moving out of bounds, let the caller know
102
103
                  elif requirement is not None and not self._is_in_boundries(requirement):
                      game_object.move()
104
105
                      out_of_bounds.append(game_object)
106
              # Executing interaction between the objects
107
108
              for coordinate, objs in interactions.items():
109
                  src_obj = objs[0]
                  dst_obj = objs[1]
110
                  # Making sure after the movement that the objects still interact with eachother
111
112
                  if Coordinate.from_legacy_coordinate(coordinate) in dst_obj.get_coordinates():
113
                      # If the destination and source objects are the same,
                         then we check that it actually hit itself (since it
114
                          may have just moved in and out of the cell this round)
115
116
                      # If it is not the same object, then interact normally
117
                      if (src_obj is dst_obj and self._is_hitting_itself(src_obj)) or \
                              (src_obj is not dst_obj):
118
119
                          interaction_callback(src_obj, dst_obj)
120
              # Executing interaction between objects to the board boundries
121
              for object1 in out_of_bounds:
122
                  out_of_bounds_callback(object1, self._is_off_board(object1.get_coordinates()))
123
124
125
         def _is_hitting_itself(self, object1: BaseGameObject):
126
127
              Checking if an object is hitting itself
```

```
128
              Receives the object which should be checked and returns True if so.
129
             coordinates = object1.get_coordinates()
130
              # If the object has no coordinates, we shouldn't check anything
131
             if 0 == len(coordinates):
132
133
                 return False
              # If the object has a coordinate twice, then it hit itself
134
             if 1 != coordinates.count(coordinates[0]):
135
136
                  return True
137
             return False
138
139
         def _is_interacting(self, coordinate, object1: BaseGameObject, object2: BaseGameObject) -> bool:
140
141
142
              Returns whether the two objects interact with each other
              (e.g. share some coordinate(s) at the given moment)
143
144
             for coordinate in object1.get_coordinates():
145
                  if coordinate in object2.get_coordinates():
146
147
                      return True
             return False
148
149
          def _get_object_at_coordinate(self, coordinate: Coordinate) -> Optional[BaseGameObject]:
150
151
152
              Returns the object residing in the requested coordinate.
153
             If no object exists at the specified coordinate None is returned.
154
155
              for game_object in self._game_objects:
                  if coordinate in game_object.get_coordinates():
156
157
                      return game_object
158
             return None # No object at the specified coordinate
159
160
         def _is_off_board(self, coordinates: List[Coordinate]) -> bool:
    """
161
162
163
              Checks if the given coordinates are completely off the game board
164
              (e.g. none of the coordinates are within the board's boundries)
             Returns True if all are off board, False if at least one is on board.
165
166
             for coordinate in coordinates:
167
                  # Checking if at least one coordinate is on the board
168
                  if self._is_in_boundries(coordinate):
169
                      return False
170
171
              return True
172
         def _is_in_boundries(self, coordinate: Coordinate) -> bool:
173
174
              Checks if the given coordinate is within the board's boundries.
175
176
              Returns True if exists, False otherwise.
177
             return is_in_boundries(self._dimensions.row, self._dimensions.column, coordinate)
178
```

5 common.py

```
1
    # FILE : common.py
   # WRITERS : Nimrod Mallis, mallis ; Dor Kaiser, dor_kaiser
   # EXERCISE : intro2cs1 ex10 2023
4
    # DESCRIPTION: Common functionality for use around the codebase
   # STUDENTS I DISCUSSED THE EXERCISE WITH: N/A
   # WEB PAGES I USED: N/A
    # NOTES: N/A
   9
10
    from typing import List, Tuple
11
   from game_display import GameDisplay
12
13
    class Direction(object):
14
15
        Used as an Enum for Directions
16
17
18
       LEFT = "Left"
       RIGHT = "Right"
19
       UP = "Up"
20
21
       DOWN = "Down"
22
       Directions = [LEFT, RIGHT, UP, DOWN]
23
24
    class Coordinate(object):
25
26
27
        Represents a coordinate in a Two-Dimensional Coordinate System
28
29
       def __init__(self, row: int, column: int) -> None:
30
31
           Initializes the coordinate with the given Y (row) and X (column) values
33
           # Attributes are intentionally public, to refrain from getter/setter functions
34
           self.row = row
35
           self.column = column
36
37
       Ostaticmethod
38
       def from_legacy_coordinate(legacy_coordinate: Tuple[int, int]):
39
40
           Creating a Coordinate from a legacy (X, Y) Tuple Coordinate
41
42
           return Coordinate(legacy_coordinate[1], legacy_coordinate[0])
43
44
45
        def __str__(self) -> str:
46
           Prints the coordinates in (row, column) format
47
           return "({row}, {column})".format(row=self.row, column=self.column)
49
50
51
        def __repr__(self) -> str:
52
           Returns the coordinates in a string format (similar to \_\_str\_\_)
53
54
55
           return self.__str__()
       def __eq__(self, other: object) -> bool:
57
58
           Compares one coordinate to another.
```

```
60
              A coordinate is equal if both the row and column coordinates are exact.
 61
              if (other.row == self.row) and (other.column == self.column):
 62
                  return True
 63
 64
              return False
 65
     class BaseGameObject(object):
 66
 67
 68
          Represents a basic game object for the Snake Game.
          All game objects set on the board should inherit from this class
 69
          and \ implement \ its \ functions
 70
 71
 72
          def __init__(self, coordinates: List[Coordinate], color: str) -> None:
 73
 74
              Initializes the object.
 75
 76
 77
              :param coordinates: List of coordinates occupied
                                  by the object on initialization.
 78
 79
              :param color: The color of the object when shown graphically.
 80
              self._coordinates = coordinates
 81
              self._color = color
 82
 83
 84
          def __str__(self) -> str:
 85
              Prints the game object's Coordinates
 86
 87
              Primarily used for debugging matters.
 88
 89
              return str(self._coordinates)
 90
         def __len__(self) -> int:
 91
 92
 93
              Returns the length of the object.
              The length is how many cells the object occupies
 94
 95
              in a two dimensional coordinate system.
 96
 97
              Builtin len() function should be used.
 98
              return len(self._coordinates)
 99
100
101
          def get_object_color(self):
102
103
              Returns the color of the object.
104
              return self._color
105
106
          def get_coordinates(self) -> List[Coordinate]:
107
108
109
              Returns the coordinates the object occupies.
110
111
              return self._coordinates
112
113
          def movement_requirements(self) -> Coordinate:
114
              Relevant mostly for moving game objects.
115
116
              Should specify the coordinate the game object
              requires to be empty before moving.
117
118
119
              Calling this directly will throw
              NotImplementedError\ Exception.
120
121
122
              raise NotImplementedError
123
          def move(self) -> bool:
124
125
              Relevant mostly for moving game objects.
126
127
              Should move the object in any way seem fit.
```

```
128
              Calling this directly will throw
129
130
              NotImplementedError Exception.
131
132
              raise NotImplementedError
133
     class BaseDynamicGameObject(BaseGameObject):
134
135
136
          Represents a dynamic game object, which is on the move during the game.
          This is an addition over the regular BaseGameObject in a manner which
137
          allows easier interface for more complicated moving objects.
138
139
140
          def __init__(self, starting_direction, coordinates: List[Coordinate], color: str) -> None:
141
142
              Initializes the object with the given starting direction,
143
144
              coordinates occupied by the object, and the color of the object on display.
145
              super().__init__(coordinates, color)
146
147
              self._current_direction = starting_direction
148
149
          def movement_requirements(self) -> Coordinate:
150
151
152
              Returns the requirements for the given direction regardless of
153
              whether it is valid or invalid (e.g. if the current direction is
              left and this function is given right)
154
155
              head = self._coordinates[0] # The head element
156
157
              if Direction.LEFT == self._current_direction:
158
                  return Coordinate(head.row, head.column-1)
              elif Direction.RIGHT == self._current_direction:
159
160
                  return Coordinate(head.row, head.column+1)
161
              elif Direction.UP == self._current_direction:
                 return Coordinate(head.row+1, head.column)
162
163
              else: # Direction is Down
164
                  return Coordinate(head.row-1, head.column)
165
     def is_in_boundries(height: int, width: int, coordinate: Coordinate) -> bool:
166
167
168
          Checks if a coordinate is within the given height X width dimensions.
169
          # Checking the row coordinate is within the boundries
170
171
          if coordinate.row >= height or coordinate.row < 0:</pre>
             return False
172
173
174
          # Checking the column coordinate is within the boundries
          if coordinate.column >= width or coordinate.column < 0:</pre>
175
176
              return False
177
         return True
178
179
180
     def draw_coordinates(
          gui: GameDisplay, dimensions: Coordinate, coordinates: List[Coordinate], color) -> None:
181
182
          Draws a list of coordinates on a game display, with the given color.
183
184
185
          for coordinate in coordinates:
              # Only if the coordinate is within the display board draw it
186
187
              if is_in_boundries(dimensions.row, dimensions.column, coordinate):
188
                  gui.draw_cell(coordinate.column, coordinate.row, color)
```

6 results.py

```
a = [(0, {(20, 15): 'black', (20, 14): 'black', (20, 13): 'black', (28, 13):
           'green', (16, 25): 'blue', (16, 24): 'blue', (16, 23): 'blue'}), (0, {(20, 16): 'black', (20, 15): 'black',
           (20, 14): 'black', (21, 14): 'green', (28, 13): 'green', (16, 25): 'blue', (16, 24): 'blue', (16, 23):
           'blue', (7, 27): 'blue', (6, 27): 'blue', (5, 27): 'blue'}), (0, {(20, 17): 'black', (20, 16): 'black', (20,
           15): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (16, 26): 'blue', (16, 25): 'blue',
           (16, 24): 'blue', (8, 27): 'blue', (7, 27): 'blue', (6, 27): 'blue'}), (0, {(20, 18): 'black', (20, 17):
            'black', (20, 16): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (16, 26): 'blue', (16,
           25): 'blue', (16, 24): 'blue', (8, 27): 'blue', (7, 27): 'blue', (6, 27): 'blue'}), (0, {(20, 19): 'black',
            (20, 18): 'black', (20, 17): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (16, 27):
          'blue', (16, 26): 'blue', (16, 25): 'blue', (9, 27): 'blue', (8, 27): 'blue', (7, 27): 'blue'}), (0, {(20, 20): 'black', (20, 19): 'black', (20, 18): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green',
10
11
           (16, 27): 'blue', (16, 26): 'blue', (16, 25): 'blue', (9, 27): 'blue', (8, 27): 'blue', (7, 27): 'blue'}), (0, {(20, 21): 'black', (20, 20): 'black', (20, 19): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3):
12
13
            'green', (16, 28): 'blue', (16, 27): 'blue', (16, 26): 'blue', (10, 27): 'blue', (9, 27): 'blue', (8, 27):
14
15
           'blue'}), (0, {(20, 22): 'black', (20, 21): 'black', (20, 20): 'black', (21, 14): 'green', (28, 13): 'green',
16
          (12, 3): 'green', (16, 28): 'blue', (16, 27): 'blue', (16, 26): 'blue', (10, 27): 'blue', (9, 27): 'blue', (8, 27): 'blue'}), (0, {(20, 23): 'black', (20, 22): 'black', (20, 21): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (16, 29): 'blue', (16, 28): 'blue', (16, 27): 'blue', (11, 27): 'blue', (10, 27):
17
18
19
           'blue', (9, 27): 'blue'}), (0, {(21, 23): 'black', (20, 23): 'black', (20, 22): 'black', (21, 14): 'green',
20
           (28, 13): 'green', (12, 3): 'green', (16, 29): 'blue', (16, 28): 'blue', (16, 27): 'blue', (11, 27): 'blue', (10, 27): 'blue', (9, 27): 'blue'}), (0, {(22, 23): 'black', (21, 23): 'black', (20, 23): 'black', (21, 14):
21
22
            'green', (28, 13): 'green', (12, 3): 'green', (16, 29): 'blue', (16, 28): 'blue', (12, 27): 'blue', (11, 27):
           'blue', (10, 27): 'blue'}), (0, {(23, 23): 'black', (22, 23): 'black', (21, 23): 'black', (21, 14): 'green',
24
           (28, 13): 'green', (12, 3): 'green', (16, 29): 'blue', (16, 28): 'blue', (12, 27): 'blue', (11, 27): 'blue',
           (10, 27): 'blue'}), (0, {(24, 23): 'black', (23, 23): 'black', (22, 23): 'black', (21, 14): 'green', (28,
26
          13): 'green', (12, 3): 'green', (16, 29): 'blue', (13, 27): 'blue', (12, 27): 'blue', (11, 27): 'blue'}), (0, {(25, 23): 'black', (24, 23): 'black', (23, 23): 'black', (24, 23): 'black', (23, 23): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (16, 29): 'blue', (13, 27): 'blue', (12, 27): 'blue', (11, 27): 'blue'}), (0, {(26, 23): 'black', (26, 23): 'black', (27, 27): 'blue', (11, 27): 'blue', (28, 23): 'black', (28, 28): 'black
27
28
29
           (25, 23): 'black', (24, 23): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (25, 29): 'blue', (25, 28): 'blue', (14, 27): 'blue', (13, 27): 'blue', (12, 27): 'blue'}), (0, {(27, 23): 'black', (25, 28): 'blue', (27, 28): 'blue', (28, 28): 'blue', 
31
           (26, 23): 'black', (25, 23): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (25, 29):
32
           'blue', (25, 28): 'blue', (14, 27): 'blue', (13, 27): 'blue', (12, 27): 'blue'}), (0, {(27, 22): 'black',
           (27, 23): 'black', (26, 23): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (25, 29): 'blue', (15, 27): 'blue', (14, 27): 'blue', (13, 27): 'blue'}), (0, {(27, 21): 'black', (27, 22): 'black',
34
35
           (27, 23): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (25, 29): 'blue', (15, 27):
36
            'blue', (14, 27): 'blue', (13, 27): 'blue'}), (0, {(27, 20): 'black', (27, 21): 'black', (27, 22): 'black',
37
           (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (16, 27): 'blue', (15, 27): 'blue', (14, 27): 'blue',
38
           (6, 10): 'blue', (7, 10): 'blue', (8, 10): 'blue'}), (0, {(27, 19): 'black', (27, 20): 'black', (27, 21):
39
           'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (16, 27): 'blue', (15, 27): 'blue', (14, 27): 'blue', (6, 10): 'blue', (7, 10): 'blue', (8, 10): 'blue'}), (0, {(27, 18): 'black', (27, 19): 'black',
40
41
           (27, 20): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (17, 27): 'blue', (16, 27):
42
           'blue', (15, 27): 'blue', (5, 10): 'blue', (6, 10): 'blue', (7, 10): 'blue'}), (0, {(27, 17): 'black', (27,
43
           18): 'black', (27, 19): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (17, 27): 'blue',
44
           (16, 27): 'blue', (15, 27): 'blue', (5, 10): 'blue', (6, 10): 'blue', (7, 10): 'blue'}), (0, {(27, 16):
45
           'black', (27, 17): 'black', (27, 18): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (18,
46
           27): 'blue', (17, 27): 'blue', (16, 27): 'blue', (4, 10): 'blue', (5, 10): 'blue', (6, 10): 'blue'}), (0,
47
           {(27, 15): 'black', (27, 16): 'black', (27, 17): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (12, 3
48
           'green', (18, 27): 'blue', (17, 27): 'blue', (16, 27): 'blue', (4, 10): 'blue', (5, 10): 'blue', (6, 10): 'blue'}), (0, {(27, 14): 'black', (27, 15): 'black', (27, 16): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (19, 27): 'blue', (18, 27): 'blue', (17, 27): 'blue', (3, 10): 'blue', (4, 10): 'blue', (5,
50
51
           10): 'blue'}), (0, {(28, 14): 'black', (27, 14): 'black', (27, 15): 'black', (21, 14): 'green', (28, 13):
            'green', (12, 3): 'green', (19, 27): 'blue', (18, 27): 'blue', (17, 27): 'blue', (3, 10): 'blue', (4, 10): 'blue', (5, 10): 'blue'}), (0, {(29, 14): 'black', (28, 14): 'black', (27, 14): 'black', (21, 14): 'green',
53
54
           (28, 13): 'green', (12, 3): 'green', (20, 27): 'blue', (19, 27): 'blue', (18, 27): 'blue', (2, 10): 'blue',
55
           (3, 10): 'blue', (4, 10): 'blue'}), (0, {(29, 13): 'black', (29, 14): 'black', (28, 14): 'black', (21, 14):
56
             green', (28, 13): 'green', (12, 3): 'green', (20, 27): 'blue', (19, 27): 'blue', (18, 27): 'blue', (2, 10):
           'blue', (3, 10): 'blue', (4, 10): 'blue'}), (0, {(29, 12): 'black', (29, 13): 'black', (29, 14): 'black',
58
           (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (21, 27): 'blue', (20, 27): 'blue', (19, 27): 'blue',
```

```
(1, 10): 'blue', (2, 10): 'blue', (3, 10): 'blue'}), (0, {(29, 11): 'black', (29, 12): 'black', (29, 13):
                'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (21, 27): 'blue', (20, 27): 'blue', (19, 27): 'blue', (1, 10): 'blue', (2, 10): 'blue', (3, 10): 'blue'}), (0, {(29, 10): 'black', (29, 11): 'black', (20, 11
  61
  62
                (29, 12): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (22, 27): 'blue', (21, 27):
                 'blue', (20, 27): 'blue', (0, 10): 'blue', (1, 10): 'blue', (2, 10): 'blue'}), (0, {(29, 9): 'black', (29,
  64
                10): 'black', (29, 11): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (22, 27): 'blue',
  65
                (21, 27): 'blue', (20, 27): 'blue', (0, 10): 'blue', (1, 10): 'blue', (2, 10): 'blue'}), (0, {(29, 8):
   66
                'black', (29, 9): 'black', (29, 10): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (23,
  67
                27): 'blue', (22, 27): 'blue', (21, 27): 'blue', (0, 10): 'blue', (1, 10): 'blue'}), (0, {(29, 7): 'black',
   68
                (29, 8): 'black', (29, 9): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (23, 27): 'blue',
  69
                (22, 27): 'blue', (21, 27): 'blue', (0, 10): 'blue', (1, 10): 'blue'}), (0, {(28, 7): 'black', (29, 7): 'black', (29, 8): 'black', (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (24, 27): 'blue', (23, 14): 'green', (12, 14): 'green'
   70
   71
                27): 'blue', (22, 27): 'blue', (0, 10): 'blue'}), (0, {(28, 8): 'black', (28, 7): 'black', (29, 7): 'black',
  72
                (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (24, 27): 'blue', (23, 27): 'blue', (22, 27): 'blue',
  73
   74
                 (0, 10): 'blue'}), (0, {(28, 9): 'black', (28, 8): 'black', (28, 7): 'black', (21, 14): 'green', (28, 13):
                 'green', (12, 3): 'green', (25, 27): 'blue', (24, 27): 'blue', (23, 27): 'blue', (5, 12): 'blue', (4, 12):
  75
                'blue', (3, 12): 'blue'}), (0, {(28, 10): 'black', (28, 9): 'black', (28, 8): 'black', (21, 14): 'green',
   76
                (28, 13): 'green', (12, 3): 'green', (25, 27): 'blue', (24, 27): 'blue', (23, 27): 'blue', (5, 12): 'blue', (4, 12): 'blue', (3, 12): 'blue'}), (0, {(28, 11): 'black', (28, 10): 'black', (28, 9): 'black', (21, 14):
   77
  78
                  'green', (28, 13): 'green', (12, 3): 'green', (26, 27): 'blue', (25, 27): 'blue', (24, 27): 'blue', (6, 12):
                 'blue', (5, 12): 'blue', (4, 12): 'blue'}), (0, {(28, 12): 'black', (28, 11): 'black', (28, 10): 'black',
   80
                (21, 14): 'green', (28, 13): 'green', (12, 3): 'green', (26, 27): 'blue', (25, 27): 'blue', (24, 27): 'blue',
   81
                (6, 12): 'blue', (5, 12): 'blue', (4, 12): 'blue'}), (1, {(28, 13): 'black', (28, 12): 'black', (28, 11):
  82
                'black', (21, 14): 'green', (16, 28): 'green', (12, 3): 'green', (27, 27): 'blue', (26, 27): 'blue', (25,
  83
   84
                27): 'blue', (7, 12): 'blue', (6, 12): 'blue', (5, 12): 'blue'}), (1, {(27, 13): 'black', (28, 13): 'black',
  85
                (28, 12): 'black', (28, 11): 'black', (21, 14): 'green', (16, 28): 'green', (12, 3): 'green', (27, 27):
   86
                 'blue', (26, 27): 'blue', (25, 27): 'blue', (7, 12): 'blue', (6, 12): 'blue', (5, 12): 'blue'}), (1, {(26,
   87
                13): 'black', (27, 13): 'black', (28, 13): 'black', (28, 12): 'black', (28, 11): 'black', (21, 14): 'green',
   88
                (16, 28): 'green', (12, 3): 'green', (28, 27): 'blue', (27, 27): 'blue', (26, 27): 'blue', (8, 12): 'blue',
   89
   90
                 (7, 12): 'blue', (6, 12): 'blue'}), (1, {(25, 13): 'black', (26, 13): 'black', (27, 13): 'black', (28, 13):
                 'black', (28, 12): 'black', (28, 11): 'black', (21, 14): 'green', (16, 28): 'green', (12, 3): 'green', (28,
  91
                27): 'blue', (27, 27): 'blue', (26, 27): 'blue', (8, 12): 'blue', (7, 12): 'blue', (6, 12): 'blue'}), (1,
  92
                {(24, 13): 'black', (25, 13): 'black', (26, 13): 'black', (27, 13): 'black', (28, 13): 'black', (28, 12): 'black', (21, 14): 'green', (16, 28): 'green', (12, 3): 'green', (29, 27): 'blue', (28, 27): 'blue', (27, 12): 'blue', (28, 27): 'blue', (28, 28): 'blue', (28
  93
  94
                27): 'blue', (9, 12): 'blue', (8, 12): 'blue', (7, 12): 'blue'}), (1, {(23, 13): 'black', (24, 13): 'black',
                (25, 13): 'black', (26, 13): 'black', (27, 13): 'black', (28, 13): 'black', (21, 14): 'green', (16, 28):
  96
                 'green', (12, 3): 'green', (29, 27): 'blue', (28, 27): 'blue', (27, 27): 'blue', (9, 12): 'blue', (8, 12):
  97
                'blue', (7, 12): 'blue'}), (1, {(22, 13): 'black', (23, 13): 'black', (24, 13): 'black', (25, 13): 'black',
  98
                (26, 13): 'black', (27, 13): 'black', (21, 14): 'green', (16, 28): 'green', (12, 3): 'green', (30, 27): 'blue', (29, 27): 'blue', (28, 27): 'blue', (10, 12): 'blue', (9, 12): 'blue', (8, 12): 'blue'}), (1, {(21,
  99
 100
                13): 'black', (22, 13): 'black', (23, 13): 'black', (24, 13): 'black', (25, 13): 'black', (26, 13): 'black',
101
                (21, 14): 'green', (16, 28): 'green', (12, 3): 'green', (30, 27): 'blue', (29, 27): 'blue', (28, 27): 'blue',
102
                 (10, 12): 'blue', (9, 12): 'blue', (8, 12): 'blue'}), (3, {(21, 14): 'black', (21, 13): 'black', (22, 13):
103
                 'black', (23, 13): 'black', (24, 13): 'black', (25, 13): 'black', (12, 3): 'green', (1, 20): 'green', (16,
104
               28): 'green', (31, 27): 'blue', (30, 27): 'blue', (29, 27): 'blue', (11, 12): 'blue', (10, 12): 'blue', (9, 12): 'blue'), (3, {(21, 15): 'black', (21, 14): 'black', (21, 13): 'black', (22, 13): 'black', (23, 13): 'black', (24, 13): 'black', (25, 13): 'black', (12, 3): 'green', (1, 20): 'green', (16, 28): 'green', (31, 12): 'green', (15, 28): 'green', (16, 28): 'green', (16, 28): 'green', (16, 28): 'green', (17, 20): 'green', (18, 28): 'green', (1
105
106
107
                27): 'blue', (30, 27): 'blue', (29, 27): 'blue', (11, 12): 'blue', (10, 12): 'blue', (9, 12): 'blue'}), (3,
108
                {(21, 16): 'black', (21, 15): 'black', (21, 14): 'black', (21, 13): 'black', (22, 13): 'black', (23, 13): 'black', (24, 13): 'black', (25, 13): 'black', (12, 3): 'green', (1, 20): 'green', (16, 28): 'green', (32, 13): 'black', (10, 10): 'green', (10, 10): 'gre
109
110
                27): 'blue', (31, 27): 'blue', (30, 27): 'blue', (12, 12): 'blue', (11, 12): 'blue', (10, 12): 'blue'}), (3,
111
                {(21, 17): 'black', (21, 16): 'black', (21, 15): 'black', (21, 14): 'black', (21, 13): 'black', (22, 13):
112
                'black', (23, 13): 'black', (24, 13): 'black', (25, 13): 'black', (12, 3): 'green', (1, 20): 'green', (16,
113
               28): 'green', (32, 27): 'blue', (31, 27): 'blue', (30, 27): 'blue', (12, 12): 'blue', (11, 12): 'blue', (10, 12): 'blue'), (3, {(20, 17): 'black', (21, 17): 'black', (21, 16): 'black', (21, 15): 'black', (21, 14): 'black', (21, 13): 'black', (22, 13): 'black', (23, 13): 'black', (24, 13): 'black', (12, 3): 'green', (1, 14): 'black', (12, 14): 'black', (12, 14): 'black', (12, 15): 'black', (
114
115
116
                20): 'green', (16, 28): 'green', (33, 27): 'blue', (32, 27): 'blue', (31, 27): 'blue', (13, 12): 'blue', (12,
117
                12): 'blue', (11, 12): 'blue'}), (3, {(19, 17): 'black', (20, 17): 'black', (21, 17): 'black', (21, 16):
118
                 'black', (21, 15): 'black', (21, 14): 'black', (21, 13): 'black', (22, 13): 'black', (23, 13): 'black', (12,
119
                3): 'green', (1, 20): 'green', (16, 28): 'green', (33, 27): 'blue', (32, 27): 'blue', (31, 27): 'blue', (13,
120
                12): 'blue', (12, 12): 'blue', (11, 12): 'blue'}), (3, {(18, 17): 'black', (19, 17): 'black', (20, 17):
121
                 'black', (21, 17): 'black', (21, 16): 'black', (21, 15): 'black', (21, 14): 'black', (21, 13): 'black', (22,
122
                13): 'black', (12, 3): 'green', (1, 20): 'green', (16, 28): 'green', (34, 27): 'blue', (33, 27): 'blue', (32,
123
                27): 'blue', (14, 12): 'blue', (13, 12): 'blue', (12, 12): 'blue')), (3, {(17, 17): 'black', (18, 17):
124
125
                'black', (19, 17): 'black', (20, 17): 'black', (21, 17): 'black', (21, 16): 'black', (21, 15): 'black', (21,
                14): 'black', (21, 13): 'black', (12, 3): 'green', (1, 20): 'green', (16, 28): 'green', (34, 27): 'blue',
126
                (33, 27): 'blue', (32, 27): 'blue', (14, 12): 'blue', (13, 12): 'blue', (12, 12): 'blue'), (3, {(16, 17):
```

```
'black', (17, 17): 'black', (18, 17): 'black', (19, 17): 'black', (20, 17): 'black', (21, 17): 'black', (21,
128
       16): 'black', (21, 15): 'black', (21, 14): 'black', (12, 3): 'green', (1, 20): 'green', (16, 28): 'green',
129
       (35, 27): 'blue', (34, 27): 'blue', (33, 27): 'blue', (15, 12): 'blue', (14, 12): 'blue', (13, 12): 'blue'})
130
       (3, {(16, 18): 'black', (16, 17): 'black', (17, 17): 'black', (18, 17): 'black', (19, 17): 'black', (20, 17):
131
       'black', (21, 17): 'black', (21, 16): 'black', (21, 15): 'black', (12, 3): 'green', (1, 20): 'green', (16,
132
       28): 'green', (35, 27): 'blue', (34, 27): 'blue', (33, 27): 'blue', (15, 12): 'blue', (14, 12): 'blue', (13,
133
       12): 'blue'}), (3, {(16, 19): 'black', (16, 18): 'black', (16, 17): 'black', (17, 17): 'black', (18, 17):
134
       'black', (19, 17): 'black', (20, 17): 'black', (21, 17): 'black', (21, 16): 'black', (12, 3): 'green', (1,
135
136
       20): 'green', (16, 28): 'green', (36, 27): 'blue', (35, 27): 'blue', (34, 27): 'blue', (16, 12): 'blue', (15,
       12): 'blue', (14, 12): 'blue'}), (3, {(16, 20): 'black', (16, 19): 'black', (16, 18): 'black', (16, 17):
137
       'black', (17, 17): 'black', (18, 17): 'black', (19, 17): 'black', (20, 17): 'black', (21, 17): 'black', (12,
138
      3): 'green', (1, 20): 'green', (16, 28): 'green', (36, 27): 'blue', (35, 27): 'blue', (34, 27): 'blue', (16, 12): 'blue', (15, 12): 'blue', (14, 12): 'blue'}), (3, {(16, 21): 'black', (16, 20): 'black', (16, 19):
139
140
       'black', (16, 18): 'black', (16, 17): 'black', (17, 17): 'black', (18, 17): 'black', (19, 17): 'black', (20,
141
142
       17): 'black', (12, 3): 'green', (1, 20): 'green', (16, 28): 'green', (37, 27): 'blue', (36, 27): 'blue', (35,
       27): 'blue', (17, 12): 'blue', (16, 12): 'blue', (15, 12): 'blue'}), (3, {(16, 22): 'black', (16, 21):
143
       'black', (16, 20): 'black', (16, 19): 'black', (16, 18): 'black', (16, 17): 'black', (17, 17): 'black', (18,
144
       17): 'black', (19, 17): 'black', (12, 3): 'green', (1, 20): 'green', (16, 28): 'green', (37, 27): 'blue',
145
       (36, 27): 'blue', (35, 27): 'blue', (17, 12): 'blue', (16, 12): 'blue', (15, 12): 'blue')), (3, {(16, 23):
146
       'black', (16, 22): 'black', (16, 21): 'black', (16, 20): 'black', (16, 19): 'black', (16, 18): 'black', (16,
       17): 'black', (17, 17): 'black', (18, 17): 'black', (12, 3): 'green', (1, 20): 'green', (16, 28): 'green',
148
       (38, 27): 'blue', (37, 27): 'blue', (36, 27): 'blue', (18, 12): 'blue', (17, 12): 'blue', (16, 12): 'blue'})
149
       (3, {(16, 24): 'black', (16, 23): 'black', (16, 22): 'black', (16, 21): 'black', (16, 20): 'black', (16, 19):
150
       'black', (16, 18): 'black', (16, 17): 'black', (17, 17): 'black', (12, 3): 'green', (1, 20): 'green', (16,
151
       28): 'green', (38, 27): 'blue', (37, 27): 'blue', (36, 27): 'blue', (18, 12): 'blue', (17, 12): 'blue', (16,
152
153
      12): 'blue'}), (3, {(16, 25): 'black', (16, 24): 'black', (16, 23): 'black', (16, 22): 'black', (16, 21): 'black', (16, 20): 'black', (16, 19): 'black', (16, 18): 'black', (16, 17): 'black', (12, 3): 'green', (1, 18): 'black', (16, 18): 'bla
154
155
       20): 'green', (16, 28): 'green', (39, 27): 'blue', (38, 27): 'blue', (37, 27): 'blue', (19, 12): 'blue', (18,
156
       12): 'blue', (17, 12): 'blue'}), (3, {(16, 26): 'black', (16, 25): 'black', (16, 24): 'black', (16, 23):
157
158
       'black', (16, 22): 'black', (16, 21): 'black', (16, 20): 'black', (16, 19): 'black', (16, 18): 'black', (12,
       3): 'green', (1, 20): 'green', (16, 28): 'green', (39, 27): 'blue', (38, 27): 'blue', (37, 27): 'blue', (19,
159
       12): 'blue', (18, 12): 'blue', (17, 12): 'blue'), (3, {(16, 27): 'black', (16, 26): 'black', (16, 25):
160
       'black', (16, 24): 'black', (16, 23): 'black', (16, 22): 'black', (16, 21): 'black', (16, 20): 'black', (16,
161
       19): 'black', (12, 3): 'green', (1, 20): 'green', (16, 28): 'green', (39, 27): 'blue', (38, 27): 'blue', (20,
162
       12): 'blue', (19, 12): 'blue', (18, 12): 'blue'}), (6, {(16, 28): 'black', (16, 27): 'black', (16, 26):
163
       'black', (16, 25): 'black', (16, 24): 'black', (16, 23): 'black', (16, 22): 'black', (16, 21): 'black', (16,
164
       20): 'black', (12, 3): 'green', (5, 15): 'green', (1, 20): 'green', (39, 27): 'blue', (38, 27): 'blue', (20,
165
       12): 'blue', (19, 12): 'blue', (18, 12): 'blue'}), (6, {(15, 28): 'black', (16, 28): 'black', (16, 27):
166
       'black', (16, 26): 'black', (16, 25): 'black', (16, 24): 'black', (16, 23): 'black', (16, 22): 'black', (16,
167
       21): 'black', (16, 20): 'black', (12, 3): 'green', (5, 15): 'green', (1, 20): 'green', (39, 27): 'blue', (21,
168
       12): 'blue', (20, 12): 'blue', (19, 12): 'blue'}), (6, {(14, 28): 'black', (15, 28): 'black', (16, 28):
169
       'black', (16, 27): 'black', (16, 26): 'black', (16, 25): 'black', (16, 24): 'black', (16, 23): 'black', (16,
170
       22): 'black', (16, 21): 'black', (16, 20): 'black', (12, 3): 'green', (5, 15): 'green', (1, 20): 'green',
171
       (39, 27): 'blue', (21, 12): 'blue', (20, 12): 'blue', (19, 12): 'blue'}), (6, {(13, 28): 'black', (14, 28):
172
       'black', (15, 28): 'black', (16, 28): 'black', (16, 27): 'black', (16, 26): 'black', (16, 25): 'black', (16,
173
174
       24): 'black', (16, 23): 'black', (16, 22): 'black', (16, 21): 'black', (16, 20): 'black', (12, 3): 'green',
       (5, 15): 'green', (1, 20): 'green', (12, 7): 'blue', (12, 8): 'blue', (12, 9): 'blue', (22, 12): 'blue', (21,
175
       12): 'blue, (20, 12): 'blue, (6, {(12, 28): 'black', (13, 28): 'black', (14, 28): 'black', (15, 28):
176
       'black', (16, 28): 'black', (16, 27): 'black', (16, 26): 'black', (16, 25): 'black', (16, 24): 'black', (16,
177
       23): 'black', (16, 22): 'black', (16, 21): 'black', (12, 3): 'green', (5, 15): 'green', (1, 20): 'green',
178
       (12, 7): 'blue', (12, 8): 'blue', (12, 9): 'blue', (22, 12): 'blue', (21, 12): 'blue', (20, 12): 'blue'}),
179
180
       (6, {(11, 28): 'black', (12, 28): 'black', (13, 28): 'black', (14, 28): 'black', (15, 28): 'black', (16, 28):
       'black', (16, 27): 'black', (16, 26): 'black', (16, 25): 'black', (16, 24): 'black', (16, 23): 'black', (16,
181
       22): 'black', (12, 3): 'green', (5, 15): 'green', (1, 20): 'green', (12, 6): 'blue', (12, 7): 'blue', (12,
182
       8): 'blue', (23, 12): 'blue', (22, 12): 'blue', (21, 12): 'blue'}), (6, {(10, 28): 'black', (11, 28):
183
       'black', (12, 28): 'black', (13, 28): 'black', (14, 28): 'black', (15, 28): 'black', (16, 28): 'black', (16,
184
       27): 'black', (16, 26): 'black', (16, 25): 'black', (16, 24): 'black', (16, 23): 'black', (12, 3): 'green',
185
       (5, 15): 'green', (1, 20): 'green', (12, 6): 'blue', (12, 7): 'blue', (12, 8): 'blue', (23, 12): 'blue', (22,
186
       12): 'blue', (21, 12): 'blue'}), (6, {(9, 28): 'black', (10, 28): 'black', (11, 28): 'black', (12, 28):
187
       'black', (13, 28): 'black', (14, 28): 'black', (15, 28): 'black', (16, 28): 'black', (16, 27): 'black', (16,
188
       26): 'black', (16, 25): 'black', (16, 24): 'black', (12, 3): 'green', (5, 15): 'green', (1, 20): 'green',
189
       (12, 5): 'blue', (12, 6): 'blue', (12, 7): 'blue', (24, 12): 'blue', (23, 12): 'blue', (22, 12): 'blue'})
190
       (6, {(8, 28): 'black', (9, 28): 'black', (10, 28): 'black', (11, 28): 'black', (12, 28): 'black', (13, 28):
191
       'black', (14, 28): 'black', (15, 28): 'black', (16, 28): 'black', (16, 27): 'black', (16, 26): 'black', (16,
192
       25): 'black', (12, 3): 'green', (5, 15): 'green', (1, 20): 'green', (12, 5): 'blue', (12, 6): 'blue', (12,
193
       7): 'blue', (24, 12): 'blue', (23, 12): 'blue', (22, 12): 'blue'}), (6, {(7, 28): 'black', (8, 28): 'black',
194
       (9, 28): 'black', (10, 28): 'black', (11, 28): 'black', (12, 28): 'black', (13, 28): 'black', (14, 28):
```

```
'black', (15, 28): 'black', (16, 28): 'black', (16, 27): 'black', (16, 26): 'black', (12, 3): 'green', (5,
196
                    15): 'green', (1, 20): 'green', (12, 4): 'blue', (12, 5): 'blue', (12, 6): 'blue', (25, 12): 'blue', (24, 12): 'blue', (23, 12): 'blue'), (6, {(6, 28): 'black', (7, 28): 'black', (8, 28): 'black', (9, 28): 'bla
197
198
                    (10, 28): 'black', (11, 28): 'black', (12, 28): 'black', (13, 28): 'black', (14, 28): 'black', (15, 28): 'black', (16, 28): 'black', (16, 27): 'black', (12, 3): 'green', (5, 15): 'green', (1, 20): 'green', (12, 4): 'blue', (12, 5): 'blue', (12, 6): 'blue', (25, 12): 'blue', (24, 12): 'blue', (23, 12): 'blue'}), (6, {(5, 12): 'blue', (12, 12): 'blue', (12, 12): 'blue'})
199
200
201
                     28): 'black', (6, 28): 'black', (7, 28): 'black', (8, 28): 'black', (9, 28): 'black', (10, 28): 'black', (11,
202
                    28): 'black', (12, 28): 'black', (13, 28): 'black', (14, 28): 'black', (15, 28): 'black', (16, 28): 'black',
203
                     (5, 15): 'green', (33, 21): 'green', (1, 20): 'green', (12, 3): 'blue', (12, 4): 'blue', (12, 5): 'blue',
204
                     (26, 12): 'blue', (25, 12): 'blue', (24, 12): 'blue'}), (6, {(4, 28): 'black', (5, 28): 'black', (6, 2
205
                    'black', (7, 28): 'black', (8, 28): 'black', (9, 28): 'black', (10, 28): 'black', (11, 28): 'black', (12, 28): 'black', (13, 28): 'black', (14, 28): 'black', (15, 28): 'black', (5, 15): 'green', (33, 21): 'green'
206
207
                     (1, 20): 'green', (12, 3): 'blue', (12, 4): 'blue', (12, 5): 'blue', (26, 12): 'blue', (25, 12): 'blue', (24,
208
                    12): 'blue'}), (6, {(3, 28): 'black', (4, 28): 'black', (5, 28): 'black', (6, 28): 'black', (7, 28): 'black', (8, 28): 'black', (9, 28): 'black', (10, 28): 'black', (11, 28): 'black', (12, 28): 'black', (13, 28):
209
210
                      'black', (14, 28): 'black', (5, 15): 'green', (33, 21): 'green', (1, 20): 'green', (12, 2): 'blue', (12, 3):
211
                     'blue', (12, 4): 'blue', (27, 12): 'blue', (26, 12): 'blue', (25, 12): 'blue'}), (6, {(2, 28): 'black', (3,
212
                   28): 'black', (4, 28): 'black', (5, 28): 'black', (6, 28): 'black', (7, 28): 'black', (8, 28): 'black', (9, 28): 'black', (10, 28): 'black', (11, 28): 'black', (12, 28): 'black', (13, 28): 'black', (5, 15): 'green',
213
214
                     (33, 21): 'green', (1, 20): 'green', (12, 2): 'blue', (12, 3): 'blue', (12, 4): 'blue', (27, 12): 'blue', (26, 12): 'blue', (25, 12): 'blue'}), (6, {(2, 27): 'black', (2, 28): 'black', (3, 28): 'black', (4, 28):
216
                     'black', (5, 28): 'black', (6, 28): 'black', (7, 28): 'black', (8, 28): 'black', (9, 28): 'black', (10, 28):
217
                     'black', (11, 28): 'black', (12, 28): 'black', (5, 15): 'green', (33, 21): 'green', (1, 20): 'green', (12,
218
                   1): 'blue', (12, 2): 'blue', (12, 3): 'blue', (28, 12): 'blue', (27, 12): 'blue', (26, 12): 'blue'}), (6, {(2, 26): 'black', (2, 27): 'black', (2, 28): 'black', (3, 28): 'black', (4, 28): 'black', (5, 28): 'black',
219
220
                     (6, 28): 'black', (7, 28): 'black', (8, 28): 'black', (9, 28): 'black', (10, 28): 'black', (11, 28): 'black',
221
222
                     (5, 15): 'green', (33, 21): 'green', (1, 20): 'green', (12, 1): 'blue', (12, 2): 'blue', (12, 3): 'blue',
223
                     (28, 12): 'blue', (27, 12): 'blue', (26, 12): 'blue'}), (6, {(2, 25): 'black', (2, 26): 'black', (2, 27):
224
225
                     'black', (2, 28): 'black', (3, 28): 'black', (4, 28): 'black', (5, 28): 'black', (6, 28): 'black', (7, 28):
226
                     'black', (8, 28): 'black', (9, 28): 'black', (10, 28): 'black', (5, 15): 'green', (33, 21): 'green', (1, 20):
                    'green', (12, 0): 'blue', (12, 1): 'blue', (12, 2): 'blue', (29, 12): 'blue', (28, 12): 'blue', (27, 12): 'blue'}), (6, {(2, 24): 'black', (2, 25): 'black', (2, 26): 'black', (2, 27): 'black', (2, 28): 'black', (3, 28): 'black', (2, 28): 'black',
227
228
                   28): 'black', (4, 28): 'black', (5, 28): 'black', (6, 28): 'black', (7, 28): 'black', (8, 28): 'black', (9, 28): 'black', (5, 15): 'green', (33, 21): 'green', (1, 20): 'green', (12, 0): 'blue', (12, 1): 'blue',
229
230
                    2): 'blue', (29, 12): 'blue', (28, 12): 'blue', (27, 12): 'blue'}), (6, {(2, 23): 'black', (2, 24): 'black',
                     (2, 25): 'black', (2, 26): 'black', (2, 27): 'black', (2, 28): 'black', (3, 28): 'black', (4, 28): 'black', (5, 28): 'black', (6, 28): 'black', (7, 28): 'black', (8, 28): 'black', (5, 15): 'green', (33, 21): 'green',
232
233
                     (1, 20): 'green', (12, 0): 'blue', (12, 1): 'blue', (30, 12): 'blue', (29, 12): 'blue', (28, 12): 'blue'}), (6, {(2, 22): 'black', (2, 23): 'black', (2, 24): 'black', (2, 25): 'black', (2, 26): 'black', (2, 27):
234
235
                     'black', (2, 28): 'black', (3, 28): 'black', (4, 28): 'black', (5, 28): 'black', (6, 28): 'black', (7, 28):
236
                     'black', (5, 15): 'green', (33, 21): 'green', (1, 20): 'green', (12, 0): 'blue', (12, 1): 'blue', (30, 12):
237
                     'blue', (29, 12): 'blue', (28, 12): 'blue'}), (6, {(2, 21): 'black', (2, 22): 'black', (2, 23): 'black', (2,
238
                     24): 'black', (2, 25): 'black', (2, 26): 'black', (2, 27): 'black', (2, 28): 'black', (3, 28): 'black', (4,
239
                    28): 'black', (5, 28): 'black', (6, 28): 'black', (5, 15): 'green', (33, 21): 'green', (1, 20): 'green', (12,
240
                    0): 'blue', (31, 12): 'blue', (30, 12): 'blue', (29, 12): 'blue'}), (6, {(2, 20): 'black', (2, 21): 'black',
241
                     (2, 22): 'black', (2, 23): 'black', (2, 24): 'black', (2, 25): 'black', (2, 26): 'black', (2, 27): 'black', (2, 28): 'black', (3, 28): 'black', (4, 28): 'black', (5, 28): 'black', (5, 15): 'green', (33, 21): 'green',
242
243
                     (1, 20): 'green', (12, 0): 'blue', (31, 12): 'blue', (30, 12): 'blue', (29, 12): 'blue'}), (9, {(1, 20):
244
                     'black', (2, 20): 'black', (2, 21): 'black', (2, 22): 'black', (2, 23): 'black', (2, 24): 'black', (2, 25): 'black', (2, 26): 'black', (2, 27): 'black', (2, 28): 'black', (3, 28): 'black', (4, 28): 'black', (5, 15):
245
246
                     'green', (38, 15): 'green', (33, 21): 'green', (8, 29): 'blue', (8, 28): 'blue', (8, 27): 'blue', (32, 12):
247
248
                      'blue', (31, 12): 'blue', (30, 12): 'blue'}), (9, {(1, 19): 'black', (1, 20): 'black', (2, 20): 'black', (2,
                     21): 'black', (2, 22): 'black', (2, 23): 'black', (2, 24): 'black', (2, 25): 'black', (2, 26): 'black', (2,
249
                     27): 'black', (2, 28): 'black', (3, 28): 'black', (4, 28): 'black', (5, 15): 'green', (38, 15): 'green', (33,
250
                   21): 'green', (8, 29): 'blue', (8, 28): 'blue', (8, 27): 'blue', (32, 12): 'blue', (31, 12): 'blue', (30, 12): 'blue'}), (9, {(1, 18): 'black', (1, 19): 'black', (1, 20): 'black', (2, 20): 'black', (2, 21): 'bl
251
252
                     (2, 22): 'black', (2, 23): 'black', (2, 24): 'black', (2, 25): 'black', (2, 26): 'black', (2, 27): 'black',
253
                     (2, 28): 'black', (3, 28): 'black', (4, 28): 'black', (5, 15): 'green', (38, 15): 'green', (33, 21): 'green', (8, 29): 'blue', (8, 28): 'blue', (33, 12): 'blue', (32, 12): 'blue', (31, 12): 'blue'}), (9, {(1, 17):
254
255
                      'black', (1, 18): 'black', (1, 19): 'black', (1, 20): 'black', (2, 20): 'black', (2, 21): 'black', (2, 22):
256
                     'black', (2, 23): 'black', (2, 24): 'black', (2, 25): 'black', (2, 26): 'black', (2, 27): 'black', (2, 28): 'black', (3, 28): 'black', (4, 28): 'black', (5, 15): 'green', (38, 15): 'green', (33, 21): 'green', (8, 29):
257
258
                     'blue', (8, 28): 'blue', (33, 12): 'blue', (32, 12): 'blue', (31, 12): 'blue')), (9, {(1, 16): 'black', (1,
259
                    17): 'black', (1, 18): 'black', (1, 19): 'black', (1, 20): 'black', (2, 20): 'black', (2, 21): 'black'
260
                   22): 'black', (2, 23): 'black', (2, 24): 'black', (2, 25): 'black', (2, 26): 'black', (2, 27): 'black', (2, 28): 'black', (3, 28): 'black', (5, 15): 'green', (38, 15): 'green', (33, 21): 'green', (8, 29): 'blue', (34, 28): 'black', (3, 28): 'blac
261
262
                    12): 'blue', (33, 12): 'blue', (32, 12): 'blue'}), (9, {(2, 16): 'black', (1, 16): 'black', (1, 17): 'black',
```

```
(1, 18): 'black', (1, 19): 'black', (1, 20): 'black', (2, 20): 'black', (2, 21): 'black', (2, 22): 'black',
264
                (2, 23): 'black', (2, 24): 'black', (2, 25): 'black', (2, 26): 'black', (2, 27): 'black', (2, 28): 'black', (5, 15): 'green', (38, 15): 'green', (33, 21): 'green', (8, 29): 'blue', (34, 12): 'blue', (33, 12): 'blue',
265
266
                 (32, 12): 'blue'}), (9, {(3, 16): 'black', (2, 16): 'black', (1, 16): 'black', (1, 17): 'black', (1, 18):
267
                 'black', (1, 19): 'black', (1, 20): 'black', (2, 20): 'black', (2, 21): 'black', (2, 22): 'black', (2, 23):
268
                'black', (2, 24): 'black', (2, 25): 'black', (2, 26): 'black', (2, 27): 'black', (5, 15): 'green', (38, 15):
269
                'green', (33, 21): 'green', (1, 24): 'blue', (1, 25): 'blue', (1, 26): 'blue', (35, 12): 'blue', (34, 12):
270
               'blue', (33, 12): 'blue'}), (9, {(4, 16): 'black', (3, 16): 'black', (2, 16): 'black', (1, 16): 'black', (1, 17): 'black', (1, 18): 'black', (1, 19): 'black', (1, 20): 'black', (2, 20): 'black', (2, 21): 'black
271
272
               22): 'black', (2, 23): 'black', (2, 24): 'black', (2, 25): 'black', (2, 26): 'black', (5, 15): 'green', (38,
273
               15): 'green', (33, 21): 'green', (1, 24): 'blue', (1, 25): 'blue', (1, 26): 'blue', (35, 12): 'blue', (34, 12): 'blue', (33, 12): 'blue'), (9, {(5, 16): 'black', (4, 16): 'black', (3, 16): 'black', (2, 16): 'black', (3, 16): 'bl
274
275
                (1, 16): 'black', (1, 17): 'black', (1, 18): 'black', (1, 19): 'black', (1, 20): 'black', (2, 20): 'black',
276
                (2, 21): 'black', (2, 22): 'black', (2, 23): 'black', (2, 24): 'black', (2, 25): 'black', (5, 15): 'green', (38, 15): 'green', (33, 21): 'green', (1, 23): 'blue', (1, 24): 'blue', (1, 25): 'blue', (36, 12): 'blue', (35, 12): 'blue', (34, 12): 'blue'}), (12, {(5, 15): 'black', (5, 16): 'black', (4, 16): 'black', (3, 16): 'black', (4, 16): 'black', (4, 16): 'black', (4, 16): 'black', (5, 16): 'black', (6, 16): 'black', (7, 16): 'black', (7, 16): 'black', (8, 16): 'black', (1, 1
277
278
279
                'black', (2, 16): 'black', (1, 16): 'black', (1, 17): 'black', (1, 18): 'black', (1, 19): 'black', (1, 20):
280
                'black', (2, 20): 'black', (2, 21): 'black', (2, 22): 'black', (2, 23): 'black', (2, 24): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (1, 23): 'blue', (1, 24): 'blue', (1, 25): 'blue', (36, 12):
281
282
                'blue', (35, 12): 'blue', (34, 12): 'blue'}), (12, {(5, 14): 'black', (5, 15): 'black', (5, 16): 'black', (4,
               16): 'black', (3, 16): 'black', (2, 16): 'black', (1, 16): 'black', (1, 17): 'black', (1, 18): 'black', (1, 19): 'black', (1, 20): 'black', (2, 20): 'black', (2, 21): 'black', (2, 22): 'black', (2, 23): 'black'
284
285
                24): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (1, 22): 'blue', (1, 23): 'blue', (1,
286
               24): 'blue', (37, 12): 'blue', (36, 12): 'blue', (35, 12): 'blue'}), (12, {(5, 13): 'black', (5, 14):
287
                'black', (5, 15): 'black', (5, 16): 'black', (4, 16): 'black', (3, 16): 'black', (2, 16): 'black', (1, 16):
288
                'black', (1, 17): 'black', (1, 18): 'black', (1, 19): 'black', (1, 20): 'black', (2, 20): 'black', (2, 21):
289
                'black', (2, 22): 'black', (2, 23): 'black', (2, 24): 'black', (38, 15): 'green', (33, 21): 'green', (31,
290
291
               13): 'green', (1, 22): 'blue', (1, 23): 'blue', (1, 24): 'blue', (37, 12): 'blue', (36, 12): 'blue', (35,
292
293
               12): 'blue'}), (12, {(5, 12): 'black', (5, 13): 'black', (5, 14): 'black', (5, 15): 'black', (5, 16):
294
                 'black', (4, 16): 'black', (3, 16): 'black', (2, 16): 'black', (1, 16): 'black', (1, 17): 'black', (1, 18):
                'black', (1, 19): 'black', (1, 20): 'black', (2, 20): 'black', (2, 21): 'black', (2, 22): 'black', (2, 23):
295
                'black', (2, 24): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (1, 21): 'blue', (1, 22):
296
               'blue', (1, 23): 'blue', (38, 12): 'blue', (37, 12): 'blue', (36, 12): 'blue'}), (12, {(5, 11): 'black', (5, 12): 'black', (5, 13): 'black', (5, 14): 'black', (5, 15): 'black', (5, 16): 'black', (4, 16): 'black', (3,
297
298
                16): 'black', (2, 16): 'black', (1, 16): 'black', (1, 17): 'black', (1, 18): 'black', (1, 19): 'black', (1,
               20): 'black', (2, 20): 'black', (2, 21): 'black', (2, 22): 'black', (2, 23): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (1, 21): 'blue', (1, 22): 'blue', (1, 23): 'blue', (38, 12): 'blue', (37,
300
301
                12): 'blue', (36, 12): 'blue'}), (12, {(5, 10): 'black', (5, 11): 'black', (5, 12): 'black', (5, 13):
302
                'black', (5, 14): 'black', (5, 15): 'black', (5, 16): 'black', (4, 16): 'black', (3, 16): 'black', (2, 16):
303
                'black', (1, 16): 'black', (1, 17): 'black', (1, 18): 'black', (1, 19): 'black', (1, 20): 'blue', (2, 20):
304
                'black', (2, 21): 'black', (2, 22): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (1,
305
               21): 'blue', (1, 22): 'blue', (39, 12): 'blue', (38, 12): 'blue', (37, 12): 'blue'), (12, {(5, 9): 'black',
306
                (5, 10): 'black', (5, 11): 'black', (5, 12): 'black', (5, 13): 'black', (5, 14): 'black', (5, 15): 'black', (5, 16): 'black', (4, 16): 'black', (3, 16): 'black', (2, 16): 'black', (1, 16): 'black', (1, 17): 'black',
307
308
                (1, 18): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (1, 20): 'blue', (1, 21): 'blue',
309
310
                (1, 22): 'blue', (39, 12): 'blue', (38, 12): 'blue', (37, 12): 'blue'}), (12, {(5, 8): 'black', (5, 9):
                 'black', (5, 10): 'black', (5, 11): 'black', (5, 12): 'black', (5, 13): 'black', (5, 14): 'black', (5, 15):
311
                'black', (5, 16): 'black', (4, 16): 'black', (3, 16): 'black', (2, 16): 'black', (1, 16): 'black', (1, 17):
312
                'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (1, 19): 'blue', (1, 20): 'blue', (1, 21): 'blue', (39, 12): 'blue', (38, 12): 'blue'}), (12, {(6, 8): 'black', (5, 8): 'black', (5, 9): 'black', (5,
313
314
                10): 'black', (5, 11): 'black', (5, 12): 'black', (5, 13): 'black', (5, 14): 'black', (5, 15): 'black', (5,
315
               16): 'black', (4, 16): 'black', (3, 16): 'black', (2, 16): 'black', (1, 16): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (1, 19): 'blue', (1, 20): 'blue', (1, 21): 'blue', (39, 12): 'blue', (38,
316
317
                12): 'blue'}), (12, {(7, 8): 'black', (6, 8): 'black', (5, 8): 'black', (5, 9): 'black', (5, 10): 'black',
318
                (5, 11): 'black', (5, 12): 'black', (5, 13): 'black', (5, 14): 'black', (5, 15): 'black', (5, 16): 'black', (4, 16): 'black', (3, 16): 'black', (2, 16): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13):
319
320
                 'green', (1, 18): 'blue', (1, 19): 'blue', (1, 20): 'blue', (39, 12): 'blue'}), (12, {(8, 8): 'black', (7,
321
               8): 'black', (6, 8): 'black', (5, 8): 'black', (5, 9): 'black', (5, 10): 'black', (5, 11): 'black', (5, 12):
322
                'black', (5, 13): 'black', (5, 14): 'black', (5, 15): 'black', (5, 16): 'black', (4, 16): 'black', (3, 16):
323
                'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (1, 18): 'blue', (1, 19): 'blue', (1, 20):
324
               'blue', (39, 12): 'blue'}), (12, {(9, 8): 'black', (8, 8): 'black', (7, 8): 'black', (6, 8): 'black', (5, 8): 'black', (5, 9): 'black', (5, 10): 'black', (5, 11): 'black', (5, 12): 'black', (5, 13): 'black', (5, 14): 'black', (5, 15): 'black', (5, 16): 'black', (4, 16): 'black', (38, 15): 'green', (33, 21): 'green', (31, 15): 'green', (31, 15): 'black', (10, 15): 'black',
325
326
327
               13): 'green', (6, 25): 'blue', (5, 25): 'blue', (4, 25): 'blue', (1, 17): 'blue', (1, 18): 'blue', (1, 19):
328
                 'blue'}), (12, {(10, 8): 'black', (9, 8): 'black', (8, 8): 'black', (7, 8): 'black', (6, 8): 'black', (5, 8):
329
                'black', (5, 9): 'black', (5, 10): 'black', (5, 11): 'black', (5, 12): 'black', (5, 13): 'black', (5, 14):
330
                'black', (5, 15): 'black', (5, 16): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (6,
```

```
25): 'blue', (5, 25): 'blue', (4, 25): 'blue', (1, 17): 'blue', (1, 18): 'blue', (1, 19): 'blue'}), (12,
             {(11, 8): 'black', (10, 8): 'black', (9, 8): 'black', (8, 8): 'black', (7, 8): 'black', (6, 8): 'black', (5,
333
             8): 'black', (5, 9): 'black', (5, 10): 'black', (5, 11): 'black', (5, 12): 'black', (5, 13): 'black', (5,
334
            14): 'black', (5, 15): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (7, 25): 'blue', (6, 25): 'blue', (5, 25): 'blue', (1, 16): 'blue', (1, 17): 'blue', (1, 18): 'blue'}), (12, {(12, 8): 'black',
336
             (11, 8): 'black', (10, 8): 'black', (9, 8): 'black', (8, 8): 'black', (7, 8): 'black', (6, 8): 'black', (5,
337
             8): 'black', (5, 9): 'black', (5, 10): 'black', (5, 11): 'black', (5, 12): 'black', (5, 13): 'black', (5,
338
            14): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (7, 25): 'blue', (6, 25): 'blue', (5, 25): 'blue', (1, 16): 'blue', (1, 17): 'blue', (1, 18): 'blue'}), (12, {(13, 8): 'black', (12, 8): 'b
339
340
             (11, 8): 'black', (10, 8): 'black', (9, 8): 'black', (8, 8): 'black', (7, 8): 'black', (6, 8): 'black', (5,
341
             8): 'black', (5, 9): 'black', (5, 10): 'black', (5, 11): 'black', (5, 12): 'black', (5, 13): 'black', (38,
342
             15): 'green', (33, 21): 'green', (31, 13): 'green', (8, 25): 'blue', (7, 25): 'blue', (6, 25): 'blue', (1,
             15): 'blue', (1, 16): 'blue', (1, 17): 'blue'}), (12, {(14, 8): 'black', (13, 8): 'black', (12, 8): 'black',
344
             (11, 8): 'black', (10, 8): 'black', (9, 8): 'black', (8, 8): 'black', (7, 8): 'black', (6, 8): 'black', (5,
345
             8): 'black', (5, 9): 'black', (5, 10): 'black', (5, 11): 'black', (5, 12): 'black', (38, 15): 'green', (33,
346
             21): 'green', (31, 13): 'green', (8, 25): 'blue', (7, 25): 'blue', (6, 25): 'blue', (1, 15): 'blue', (1, 16):
347
             'blue', (1, 17): 'blue'}), (12, {(15, 8): 'black', (14, 8): 'black', (13, 8): 'black', (12, 8): 'black', (11,
348
             8): 'black', (10, 8): 'black', (9, 8): 'black', (8, 8): 'black', (7, 8): 'black', (6, 8): 'black', (5, 8):
349
             'black', (5, 9): 'black', (5, 10): 'black', (5, 11): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13):
350
             'green', (9, 25): 'blue', (8, 25): 'blue', (7, 25): 'blue', (1, 14): 'blue', (1, 15): 'blue', (1, 16): 'blue'}), (12, {(16, 8): 'black', (15, 8): 'black', (14, 8): 'black', (13, 8): 'black', (12, 8): 'black', (11, 8): 'black', (10, 8): 'black', (9, 8): 'black', (8, 8): 'black', (7, 8): 'black', (6, 8): 'black', (5, 8): 'black', (10, 8
352
353
             8): 'black', (5, 9): 'black', (5, 10): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (9,
354
            25): 'blue', (8, 25): 'blue', (7, 25): 'blue', (1, 14): 'blue', (1, 15): 'blue', (1, 16): 'blue'}), (12, {(17, 8): 'black', (16, 8): 'black', (15, 8): 'black', (14, 8): 'black', (13, 8): 'black', (12, 8): 'black', (13, 8): 'black', (14, 8): 'black', (15, 8): 'blac
355
356
             (11, 8): 'black', (10, 8): 'black', (9, 8): 'black', (8, 8): 'black', (7, 8): 'black', (6, 8): 'black', (5,
357
            8): 'black', (5, 9): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (10, 25): 'blue', (9, 25): 'blue', (8, 25): 'blue', (1, 13): 'blue', (1, 14): 'blue', (1, 15): 'blue'}), (12, {(18, 8): 'black',
358
359
360
361
             (17, 8): 'black', (16, 8): 'black', (15, 8): 'black', (14, 8): 'black', (13, 8): 'black', (12, 8): 'black',
            (11, 8): 'black', (10, 8): 'black', (9, 8): 'black', (8, 8): 'black', (7, 8): 'black', (6, 8): 'black', (5, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (10, 25): 'blue', (9, 25): 'blue', (8,
362
363
             25): 'blue', (1, 13): 'blue', (1, 14): 'blue', (1, 15): 'blue'}), (12, {(19, 8): 'black', (18, 8): 'black',
364
             (17, 8): 'black', (16, 8): 'black', (15, 8): 'black', (14, 8): 'black', (13, 8): 'black', (12, 8): 'black', (11, 8): 'black', (10, 8): 'black', (9, 8): 'black', (8, 8): 'black', (7, 8): 'black', (6, 8): 'black', (38,
365
366
             15): 'green', (33, 21): 'green', (31, 13): 'green', (11, 25): 'blue', (10, 25): 'blue', (9, 25): 'blue', (1,
             12): 'blue', (1, 13): 'blue', (1, 14): 'blue'}), (12, {(20, 8): 'black', (19, 8): 'black', (18, 8): 'black',
368
             (17, 8): 'black', (16, 8): 'black', (15, 8): 'black', (14, 8): 'black', (13, 8): 'black', (12, 8): 'black',
369
             (11, 8): 'black', (10, 8): 'black', (9, 8): 'black', (8, 8): 'black', (7, 8): 'black', (38, 15): 'green',
370
             (33, 21): 'green', (31, 13): 'green', (11, 25): 'blue', (10, 25): 'blue', (9, 25): 'blue', (1, 12): 'blue',
371
             (1, 13): 'blue', (1, 14): 'blue'}), (12, {(21, 8): 'black', (20, 8): 'black', (19, 8): 'black', (18, 8):
372
              'black', (17, 8): 'black', (16, 8): 'black', (15, 8): 'black', (14, 8): 'black', (13, 8): 'black', (12, 8):
373
             'black', (11, 8): 'black', (10, 8): 'black', (9, 8): 'black', (8, 8): 'black', (38, 15): 'green', (33, 21):
374
              'green', (31, 13): 'green', (12, 25): 'blue', (11, 25): 'blue', (10, 25): 'blue', (1, 11): 'blue', (1, 12):
375
             'blue', (1, 13): 'blue'}), (12, {(22, 8): 'black', (21, 8): 'black', (20, 8): 'black', (19, 8): 'black', (18,
376
            8): 'black', (17, 8): 'black', (16, 8): 'black', (15, 8): 'black', (14, 8): 'black', (13, 8): 'black', (12, 8): 'black', (11, 8): 'black', (10, 8): 'black', (9, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (12, 25): 'blue', (11, 25): 'blue', (10, 25): 'blue', (1, 11): 'blue', (1, 12): '
377
378
379
             13): 'blue'}), (12, {(23, 8): 'black', (22, 8): 'black', (21, 8): 'black', (20, 8): 'black', (19, 8):
380
             'black', (18, 8): 'black', (17, 8): 'black', (16, 8): 'black', (15, 8): 'black', (14, 8): 'black', (13, 8): 'black', (12, 8): 'black', (11, 8): 'black', (10, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31,
381
382
             13): 'green', (13, 25): 'blue', (12, 25): 'blue', (11, 25): 'blue', (1, 10): 'blue', (1, 11): 'blue', (1,
384
             12): 'blue'}), (12, {(24, 8): 'black', (23, 8): 'black', (22, 8): 'black', (21, 8): 'black', (20, 8):
             'black', (19, 8): 'black', (18, 8): 'black', (17, 8): 'black', (16, 8): 'black', (15, 8): 'black', (14, 8):
385
             'black', (13, 8): 'black', (12, 8): 'black', (11, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31,
386
             13): 'green', (13, 25): 'blue', (12, 25): 'blue', (11, 25): 'blue', (1, 10): 'blue', (1, 11): 'blue', (1, 12): 'blue'}), (12, {(25, 8): 'black', (24, 8): 'black', (23, 8): 'black', (22, 8): 'black', (21, 8):
387
388
             'black', (20, 8): 'black', (19, 8): 'black', (18, 8): 'black', (17, 8): 'black', (16, 8): 'black', (15, 8):
389
             'black', (14, 8): 'black', (13, 8): 'black', (12, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31,
390
             13): 'green', (14, 25): 'blue', (13, 25): 'blue', (12, 25): 'blue', (1, 9): 'blue', (1, 10): 'blue', (1, 11):
391
             'blue'}), (12, {(26, 8): 'black', (25, 8): 'black', (24, 8): 'black', (23, 8): 'black', (22, 8): 'black',
392
             (21, 8): 'black', (20, 8): 'black', (19, 8): 'black', (18, 8): 'black', (17, 8): 'black', (16, 8): 'black',
393
             (15, 8): 'black', (14, 8): 'black', (13, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (14, 25): 'blue', (13, 25): 'blue', (12, 25): 'blue', (1, 9): 'blue', (1, 10): 'blue', (1, 11):
394
395
             'blue'}), (12, {(27, 8): 'black', (26, 8): 'black', (25, 8): 'black', (24, 8): 'black', (23, 8): 'black',
396
              (22, 8): 'black', (21, 8): 'black', (20, 8): 'black', (19, 8): 'black', (18, 8): 'black', (17, 8): 'black',
397
             (16, 8): 'black', (15, 8): 'black', (14, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13):
398
              'green', (15, 25): 'blue', (14, 25): 'blue', (13, 25): 'blue', (1, 8): 'blue', (1, 9): 'blue', (1, 10):
```

```
'blue'}), (12, {(28, 8): 'black', (27, 8): 'black', (26, 8): 'black', (25, 8): 'black', (24, 8): 'black',
400
          (23, 8): 'black', (22, 8): 'black', (21, 8): 'black', (20, 8): 'black', (19, 8): 'black', (18, 8): 'black',
401
          (17, 8): 'black', (16, 8): 'black', (15, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13):
402
           'green', (15, 25): 'blue', (14, 25): 'blue', (13, 25): 'blue', (1, 8): 'blue', (1, 9): 'blue', (1, 10): 'blue'}), (12, {(29, 8): 'black', (28, 8): 'black', (27, 8): 'black', (26, 8): 'black', (25, 8): 'black',
404
          (24, 8): 'black', (23, 8): 'black', (22, 8): 'black', (21, 8): 'black', (20, 8): 'black', (19, 8): 'black',
405
          (18, 8): 'black', (17, 8): 'black', (16, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13):
406
           'green', (16, 25): 'blue', (15, 25): 'blue', (14, 25): 'blue', (1, 7): 'blue', (1, 8): 'blue', (1, 9): 'blue'}), (12, {(30, 8): 'black', (29, 8): 'black', (28, 8): 'black', (27, 8): 'black', (26, 8): 'black',
407
408
          (25, 8): 'black', (24, 8): 'black', (23, 8): 'black', (22, 8): 'black', (21, 8): 'black', (20, 8): 'black',
409
          (19, 8): 'black', (18, 8): 'black', (17, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (16, 25): 'blue', (15, 25): 'blue', (14, 25): 'blue', (1, 7): 'blue', (1, 8): 'blue', (1, 9):
410
411
           'blue'}), (12, {(31, 8): 'black', (30, 8): 'black', (29, 8): 'black', (28, 8): 'black', (27, 8): 'black',
412
413
          (26, 8): 'black', (25, 8): 'black', (24, 8): 'black', (23, 8): 'black', (22, 8): 'black', (21, 8): 'black',
414
           (20, 8): 'black', (19, 8): 'black', (18, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13):
           'green', (17, 25): 'blue', (16, 25): 'blue', (15, 25): 'blue', (1, 6): 'blue', (1, 7): 'blue', (1, 8):
415
          'blue'}), (12, {(32, 8): 'black', (31, 8): 'black', (30, 8): 'black', (29, 8): 'black', (28, 8): 'black',
416
          (27, 8): 'black', (26, 8): 'black', (25, 8): 'black', (24, 8): 'black', (23, 8): 'black', (22, 8): 'black', (21, 8): 'black', (20, 8): 'black', (19, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13):
417
418
           green', (17, 25): 'blue', (16, 25): 'blue', (15, 25): 'blue', (1, 6): 'blue', (1, 7): 'blue', (1, 8):
           'blue'}), (12, {(32, 9): 'black', (32, 8): 'black', (31, 8): 'black', (30, 8): 'black', (29, 8): 'black',
420
          (28, 8): 'black', (27, 8): 'black', (26, 8): 'black', (25, 8): 'black', (24, 8): 'black', (23, 8): 'black',
421
          (22, 8): 'black', (21, 8): 'black', (20, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13):
422
           'green', (18, 25): 'blue', (17, 25): 'blue', (16, 25): 'blue', (1, 5): 'blue', (1, 6): 'blue', (1, 7): 'blue'}), (12, {(32, 10): 'black', (32, 9): 'black', (32, 8): 'black', (31, 8): 'black', (30, 8): 'black',
423
424
          (29, 8): 'black', (28, 8): 'black', (27, 8): 'black', (26, 8): 'black', (25, 8): 'black', (24, 8): 'black',
425
          (23, 8): 'black', (22, 8): 'black', (21, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (18, 25): 'blue', (17, 25): 'blue', (16, 25): 'blue', (1, 5): 'blue', (1, 6): 'blue', (1, 7):
426
427
          'blue'}), (12, {(32, 11): 'black', (32, 10): 'black', (32, 9): 'black', (32, 8): 'black', (31, 8): 'black',
428
429
430
          (30, 8): 'black', (29, 8): 'black', (28, 8): 'black', (27, 8): 'black', (26, 8): 'black', (25, 8): 'black',
          (24, 8): 'black', (23, 8): 'black', (22, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13):
431
           'green', (19, 25): 'blue', (18, 25): 'blue', (17, 25): 'blue', (1, 4): 'blue', (1, 5): 'blue', (1, 6):
432
           blue'}), (12, {(32, 12): 'black', (32, 11): 'black', (32, 10): 'black', (32, 9): 'black', (32, 8): 'black',
433
          (31, 8): 'black', (30, 8): 'black', (29, 8): 'black', (28, 8): 'black', (27, 8): 'black', (26, 8): 'black',
434
          (25, 8): 'black', (24, 8): 'black', (23, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13):
435
           green', (19, 25): 'blue', (18, 25): 'blue', (17, 25): 'blue', (1, 4): 'blue', (1, 5): 'blue', (1, 6):
436
           'blue'}), (12, {(32, 13): 'black', (32, 12): 'black', (32, 11): 'black', (32, 10): 'black', (32, 9): 'black',
437
          (32, 8): 'black', (31, 8): 'black', (30, 8): 'black', (29, 8): 'black', (28, 8): 'black', (27, 8): 'black',
438
          (26, 8): 'black', (25, 8): 'black', (24, 8): 'black', (38, 15): 'green', (33, 21): 'green', (31, 13): 'green', (20, 25): 'blue', (19, 25): 'blue', (18, 25): 'blue', (1, 3): 'blue', (1, 4): 'blue', (1, 5):
439
440
          'blue'}), (15, {(31, 13): 'black', (32, 13): 'black', (32, 12): 'black', (32, 11): 'black', (32, 10):
441
          'black', (32, 9): 'black', (32, 8): 'black', (31, 8): 'black', (30, 8): 'black', (29, 8): 'black', (28, 8): 'black', (26, 8): 'black', (26, 8): 'black', (31, 5): 'green', (38, 15): 'green', (33, 21):
442
443
           'green', (20, 25): 'blue', (19, 25): 'blue', (18, 25): 'blue', (1, 3): 'blue', (1, 4): 'blue<sup>'</sup>, (1, 5):
444
          'blue'}), (15, {(30, 13): 'black', (31, 13): 'black', (32, 13): 'black', (32, 12): 'black', (32, 11):
445
446
           'black', (32, 10): 'black', (32, 9): 'black', (32, 8): 'black', (31, 8): 'black', (30, 8): 'black', (29, 8):
          'black', (28, 8): 'black', (27, 8): 'black', (26, 8): 'black', (25, 8): 'black', (31, 5): 'green', (38, 15):
447
           'green', (33, 21): 'green', (21, 25): 'blue', (20, 25): 'blue', (19, 25): 'blue', (1, 2): 'blue', (1, 3):
448
           'blue', (1, 4): 'blue'}), (15, {(30, 14): 'black', (30, 13): 'black', (31, 13): 'black', (32, 13): 'black',
449
          (32, 12): 'black', (32, 11): 'black', (32, 10): 'black', (32, 9): 'black', (32, 8): 'black', (31, 8):
450
           'black', (30, 8): 'black', (29, 8): 'black', (28, 8): 'black', (27, 8): 'black', (26, 8): 'black', (25, 8):
451
         'black', (31, 5): 'green', (38, 15): 'green', (33, 21): 'green', (21, 25): 'blue', (20, 25): 'blue', (19, 25): 'blue', (1, 2): 'blue', (1, 3): 'blue', (1, 4): 'blue'}), (15, {(31, 14): 'black', (30, 14):
452
453
          (30, 13): 'black', (31, 13): 'black', (32, 13): 'black', (32, 12): 'black', (32, 11): 'black', (32, 10):
454
          'black', (32, 9): 'black', (32, 8): 'black', (31, 8): 'black', (30, 8): 'black', (29, 8): 'black', (28, 8): 'black', (27, 8): 'black', (26, 8): 'black', (25, 8): 'black', (31, 5): 'green', (38, 15): 'green', (33, 21):
455
456
          'green', (22, 25): 'blue', (21, 25): 'blue', (20, 25): 'blue', (1, 1): 'blue', (1, 2): 'blue', (1, 3):
457
          'blue'}), (15, {(32, 14): 'black', (31, 14): 'black', (30, 14): 'black', (30, 13): 'black', (31, 13):
458
           'black', (32, 13): 'black', (32, 12): 'black', (32, 11): 'black', (32, 10): 'black', (32, 9): 'black', (32,
459
          8): 'black', (31, 8): 'black', (30, 8): 'black', (29, 8): 'black', (28, 8): 'black', (27, 8): 'black', (26,
460
         8): 'black', (31, 5): 'green', (38, 15): 'green', (33, 21): 'green', (22, 25): 'blue', (21, 25): 'blue', (20, 25): 'blue', (1, 1): 'blue', (1, 2): 'blue', (1, 3): 'blue'}), (15, {(33, 14): 'black', (32, 14): 'black', (33, 
461
462
          (31, 14): 'black', (30, 14): 'black', (30, 13): 'black', (31, 13): 'black', (32, 13): 'black', (32, 12):
463
          'black', (32, 11): 'black', (32, 10): 'black', (32, 9): 'black', (32, 8): 'black', (31, 8): 'black', (30, 8):
464
          'black', (29, 8): 'black', (28, 8): 'black', (27, 8): 'black', (31, 5): 'green', (38, 15): 'green', (33, 21):
465
           'green', (23, 25): 'blue', (22, 25): 'blue', (21, 25): 'blue', (1, 0): 'blue', (1, 1): 'blue', (1, 2):
466
          'blue'}), (15, {(34, 14): 'black', (33, 14): 'black', (32, 14): 'black', (31, 14): 'black', (30, 14):
```

```
'black', (30, 13): 'black', (31, 13): 'black', (32, 13): 'black', (32, 12): 'black', (32, 11): 'black', (32,
468
       10): 'black', (32, 9): 'black', (32, 8): 'black', (31, 8): 'black', (30, 8): 'black', (29, 8): 'black', (28, 8): 'black', (31, 5): 'green', (38, 15): 'green', (33, 21): 'green', (23, 25): 'blue', (22, 25): 'blue', (21, 25): 'blue', (21, 25): 'blue', (21, 25): 'blue', (22, 25): 'blue', (21, 25): 'blue', (22, 25): 'blue', (21, 25): 'blue', (21, 25): 'blue', (22, 25): 'blue', (21, 25): 'blue', (22, 25): 'blue', (21, 25): 'blu
469
470
       25): 'blue', (1, 0): 'blue', (1, 1): 'blue', (1, 2): 'blue'}), (15, {(35, 14): 'black', (34, 14): 'black',
471
       (33, 14): 'black', (32, 14): 'black', (31, 14): 'black', (30, 14): 'black', (30, 13): 'black', (31, 13):
472
       'black', (32, 13): 'black', (32, 12): 'black', (32, 11): 'black', (32, 10): 'black', (32, 9): 'black', (32,
473
       8): 'black', (31, 8): 'black', (30, 8): 'black', (29, 8): 'black', (31, 5): 'green', (38, 15): 'green', (33,
474
       21): 'green', (24, 25): 'blue', (23, 25): 'blue', (22, 25): 'blue', (1, 0): 'blue', (1, 1): 'blue'}), (15, {(36, 14): 'black', (35, 14): 'black', (34, 14): 'black', (33, 14): 'black', (31, 14):
475
476
        'black', (30, 14): 'black', (30, 13): 'black', (31, 13): 'black', (32, 13): 'black', (32, 12): 'black', (32,
477
       11): 'black', (32, 10): 'black', (32, 9): 'black', (32, 8): 'black', (31, 8): 'black', (30, 8): 'black', (31,
478
              green', (38, 15): 'green', (33, 21): 'green', (24, 25): 'blue', (23, 25): 'blue', (22, 25): 'blue', (1,
479
       0): 'blue', (1, 1): 'blue'}), (15, {(37, 14): 'black', (36, 14): 'black', (35, 14): 'black', (34, 14):
480
481
       'black', (33, 14): 'black', (32, 14): 'black', (31, 14): 'black', (30, 14): 'black', (30, 13): 'black', (31,
       13): 'black', (32, 13): 'black', (32, 12): 'black', (32, 11): 'black', (32, 10): 'black', (32, 9): 'black',
482
       (32, 8): 'black', (31, 8): 'black', (31, 5): 'green', (38, 15): 'green', (33, 21): 'green', (25, 25): 'blue',
483
        (24, 25): 'blue', (23, 25): 'blue', (1, 0): 'blue'}), (15, {(38, 14): 'black', (37, 14): 'black', (36, 14):
484
        'black', (35, 14): 'black', (34, 14): 'black', (33, 14): 'black', (32, 14): 'black', (31, 14): 'black', (30,
485
       14): 'black', (30, 13): 'black', (31, 13): 'black', (32, 13): 'black', (32, 12): 'black', (32, 11): 'black',
486
        (32, 10): 'black', (32, 9): 'black', (32, 8): 'black', (31, 5): 'green', (38, 15): 'green', (33, 21):
487
        green', (25, 25): 'blue', (24, 25): 'blue', (23, 25): 'blue', (1, 0): 'blue'}), (19, {(38, 15): 'black',
488
        (38, 14): 'black', (37, 14): 'black', (36, 14): 'black', (35, 14): 'black', (34, 14): 'black', (33, 14):
489
        'black', (32, 14): 'black', (31, 14): 'black', (30, 14): 'black', (30, 13): 'black', (31, 13): 'black', (32,
490
       13): 'black', (32, 12): 'black', (32, 11): 'black', (32, 10): 'black', (32, 9): 'black', (31, 5): 'green',
491
        (33, 21): 'green', (4, 19): 'green', (26, 25): 'blue', (25, 25): 'blue', (24, 25): 'blue'}), (19, {(38, 16):
492
        'black', (38, 15): 'black', (38, 14): 'black', (37, 14): 'black', (36, 14): 'black', (35, 14): 'black', (34,
493
       14): 'black', (33, 14): 'black', (32, 14): 'black', (31, 14): 'black', (30, 14): 'black', (30, 13): 'black',
494
495
        (31, 13): 'black', (32, 13): 'black', (32, 12): 'black', (32, 11): 'black', (32, 10): 'black', (32, 9):
        'black', (31, 5): 'green', (33, 21): 'green', (4, 19): 'green', (26, 25): 'blue', (25, 25): 'blue', (24, 25):
496
       'blue', (11, 29): 'blue', (11, 28): 'blue', (11, 27): 'blue'), (19, {(38, 17): 'black', (38, 16): 'black',
497
498
        (38, 15): 'black', (38, 14): 'black', (37, 14): 'black', (36, 14): 'black', (35, 14): 'black', (34, 14):
499
       'black', (33, 14): 'black', (32, 14): 'black', (31, 14): 'black', (30, 14): 'black', (30, 13): 'black', (31,
500
       13): 'black', (32, 13): 'black', (32, 12): 'black', (32, 11): 'black', (32, 10): 'black', (32, 9): 'black',
501
       (31, 5): 'green', (33, 21): 'green', (4, 19): 'green', (27, 25): 'blue', (26, 25): 'blue', (25, 25): 'blue',
502
       (11, 29): 'blue', (11, 28): 'blue'}), (19, {(38, 18): 'black', (38, 17): 'black', (38, 16): 'black', (38,
       15): 'black', (38, 14): 'black', (37, 14): 'black', (36, 14): 'black', (35, 14): 'black', (34, 14): 'black',
504
       (33, 14): 'black', (32, 14): 'black', (31, 14): 'black', (30, 14): 'black', (30, 13): 'black', (31, 13):
505
       'black', (32, 13): 'black', (32, 12): 'black', (32, 11): 'black', (32, 10): 'black', (32, 9): 'black', (31,
506
       5): 'green', (33, 21): 'green', (4, 19): 'green', (27, 25): 'blue', (26, 25): 'blue', (25, 25): 'blue', (11, 29): 'blue', (11, 28): 'blue'}), (19, {(38, 19): 'black', (38, 18): 'black', (38, 17): 'black', (38, 16):
507
508
       'black', (38, 15): 'black', (38, 14): 'black', (37, 14): 'black', (36, 14): 'black', (35, 14): 'black', (34,
509
       14): 'black', (33, 14): 'black', (32, 14): 'black', (31, 14): 'black', (30, 14): 'black', (30, 13): 'black',
510
        (31, 13): 'black', (32, 13): 'black', (32, 12): 'black', (32, 11): 'black', (32, 10): 'black', (31, 5):
511
        'green', (33, 21): 'green', (4, 19): 'green', (28, 25): 'blue', (27, 25): 'blue', (26, 25): 'blue', (11, 29):
512
       'blue'}), (19, {(38, 20): 'black', (38, 19): 'black', (38, 18): 'black', (38, 17): 'black', (38, 16):
513
514
        'black', (38, 15): 'black', (38, 14): 'black', (37, 14): 'black', (36, 14): 'black', (35, 14): 'black', (34,
       14): 'black', (33, 14): 'black', (32, 14): 'black', (31, 14): 'black', (30, 14): 'black', (30, 13): 'black',
515
       (31, 13): 'black', (32, 13): 'black', (32, 12): 'black', (32, 11): 'black', (31, 5): 'green', (33, 21): 'green', (4, 19): 'green', (28, 25): 'blue', (27, 25): 'blue', (26, 25): 'blue', (11, 29): 'blue')), (19, (38, 21): 'black', (38, 20): 'black', (38, 19): 'black', (38, 18): 'black', (38, 17): 'black', (38, 16):
516
517
518
        'black', (38, 15): 'black', (38, 14): 'black', (37, 14): 'black', (36, 14): 'black', (35, 14): 'black', (34,
519
       14): 'black', (33, 14): 'black', (32, 14): 'black', (31, 14): 'black', (30, 14): 'black', (30, 13): 'black',
520
       (31, 13): 'black', (32, 13): 'black', (32, 12): 'black', (31, 5): 'green', (33, 21): 'green', (4, 19):
521
        'green', (29, 25): 'blue', (28, 25): 'blue', (27, 25): 'blue', (5, 25): 'blue', (4, 25): 'blue', (3, 25): 'blue'}), (19, {(38, 22): 'black', (38, 21): 'black', (38, 20): 'black', (38, 19): 'black', (38, 18):
522
523
       'black', (38, 17): 'black', (38, 16): 'black', (38, 15): 'black', (38, 14): 'black', (37, 14): 'black', (36,
524
       14): 'black', (35, 14): 'black', (34, 14): 'black', (33, 14): 'black', (32, 14): 'black', (31, 14): 'black',
525
       (30, 14): 'black', (30, 13): 'black', (31, 13): 'black', (32, 13): 'black', (31, 5): 'green', (33, 21): 'green', (4, 19): 'green', (29, 25): 'blue', (28, 25): 'blue', (27, 25): 'blue', (5, 25): 'blue', (4, 25):
526
527
       'blue', (3, 25): 'blue'}), (19, {(37, 22): 'black', (38, 22): 'black', (38, 21): 'black', (38, 20): 'black',
528
        (38, 19): 'black', (38, 18): 'black', (38, 17): 'black', (38, 16): 'black', (38, 15): 'black', (38, 14):
529
        'black', (37, 14): 'black', (36, 14): 'black', (35, 14): 'black', (34, 14): 'black', (33, 14): 'black', (32,
530
       14): 'black', (31, 14): 'black', (30, 14): 'black', (30, 13): 'black', (31, 13): 'black', (31, 5): 'green',
531
       (33, 21): 'green', (4, 19): 'green', (30, 25): 'blue', (29, 25): 'blue', (28, 25): 'blue', (6, 25): 'blue',
532
        (5, 25): 'blue', (4, 25): 'blue'}), (19, {(36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (38, 21):
533
       'black', (38, 20): 'black', (38, 19): 'black', (38, 18): 'black', (38, 17): 'black', (38, 16): 'black', (38,
534
       15): 'black', (38, 14): 'black', (37, 14): 'black', (36, 14): 'black', (35, 14): 'black', (34, 14): 'black',
```

```
(33, 14): 'black', (32, 14): 'black', (31, 14): 'black', (30, 14): 'black', (30, 13): 'black', (31, 5): 'green', (33, 21): 'green', (4, 19): 'green', (30, 25): 'blue', (29, 25): 'blue', (28, 25): 'blue', (6, 25):
536
537
         blue', (5, 25): 'blue', (4, 25): 'blue'}), (19, {(35, 22): 'black', (36, 22): 'black', (37, 22): 'black',
538
        (38, 22): 'black', (38, 21): 'black', (38, 20): 'black', (38, 19): 'black', (38, 18): 'black', (38, 17):
539
        'black', (38, 16): 'black', (38, 15): 'black', (38, 14): 'black', (37, 14): 'black', (36, 14): 'black', (35,
540
        14): 'black', (34, 14): 'black', (33, 14): 'black', (32, 14): 'black', (31, 14): 'black', (30, 14): 'black',
541
        (31, 5): 'green', (33, 21): 'green', (4, 19): 'green', (31, 25): 'blue', (30, 25): 'blue', (29, 25): 'blue',
542
        (7, 25): 'blue', (6, 25): 'blue', (5, 25): 'blue'}), (19, {(34, 22): 'black', (35, 22): 'black', (36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (38, 21): 'black', (38, 20): 'black', (38, 19): 'black', (38, 21): '
543
544
        18): 'black', (38, 17): 'black', (38, 16): 'black', (38, 15): 'black', (38, 14): 'black', (37, 14): 'black',
545
        (36, 14): 'black', (35, 14): 'black', (34, 14): 'black', (33, 14): 'black', (32, 14): 'black', (31, 14): 'black', (31, 5): 'green', (33, 21): 'green', (4, 19): 'green', (31, 25): 'blue', (30, 25): 'blue', (29, 25):
546
547
        'blue', (7, 25): 'blue', (6, 25): 'blue', (5, 25): 'blue'}), (19, {(33, 22): 'black', (34, 22): 'black', (35,
548
       22): 'black', (36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (38, 21): 'black', (38, 20): 'black',
549
550
        (38, 19): 'black', (38, 18): 'black', (38, 17): 'black', (38, 16): 'black', (38, 15): 'black', (38, 14):
        'black', (37, 14): 'black', (36, 14): 'black', (35, 14): 'black', (34, 14): 'black', (33, 14): 'black', (32,
551
       14): 'black', (31, 5): 'green', (33, 21): 'green', (4, 19): 'green', (32, 25): 'blue', (31, 25): 'blue', (30,
552
        25): 'blue', (8, 25): 'blue', (7, 25): 'blue', (6, 25): 'blue'}), (23, {(33, 21): 'black', (33, 22): 'black',
553
        (34, 22): 'black', (35, 22): 'black', (36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (38, 21):
554
        'black', (38, 20): 'black', (38, 19): 'black', (38, 18): 'black', (38, 17): 'black', (38, 16): 'black', (38,
       15): 'black', (38, 14): 'black', (37, 14): 'black', (36, 14): 'black', (35, 14): 'black', (34, 14): 'black',
556
        (33, 14): 'black', (31, 5): 'green', (33, 20): 'green', (4, 19): 'green', (32, 25): 'blue', (31, 25): 'blue',
557
        (30, 25): 'blue', (8, 25): 'blue', (7, 25): 'blue', (6, 25): 'blue'}), (23, {(32, 21): 'black', (33, 21):
558
        'black', (33, 22): 'black', (34, 22): 'black', (35, 22): 'black', (36, 22): 'black', (37, 22): 'black', (38,
559
       22): 'black', (38, 21): 'black', (38, 20): 'black', (38, 19): 'black', (38, 18): 'black', (38, 17): 'black',
560
        (38, 16): 'black', (38, 15): 'black', (38, 14): 'black', (37, 14): 'black', (36, 14): 'black', (35, 14):
561
        'black', (34, 14): 'black', (33, 14): 'black', (31, 5): 'green', (33, 20): 'green', (4, 19): 'green', (33,
562
        25): 'blue', (32, 25): 'blue', (31, 25): 'blue', (9, 25): 'blue', (8, 25): 'blue', (7, 25): 'blue'}), (23,
563
        {(31, 21): 'black', (32, 21): 'black', (33, 21): 'black', (33, 22): 'black', (34, 22): 'black', (35, 22):
564
        'black', (36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (38, 21): 'black', (38, 20): 'black', (38,
565
566
        19): 'black', (38, 18): 'black', (38, 17): 'black', (38, 16): 'black', (38, 15): 'black', (38, 14): 'black',
567
        (37, 14): 'black', (36, 14): 'black', (35, 14): 'black', (34, 14): 'black', (33, 14): 'black', (31, 5): 'green', (33, 20): 'green', (4, 19): 'green', (33, 25): 'blue', (32, 25): 'blue', (31, 25): 'blue', (9, 25):
568
569
        'blue', (8, 25): 'blue', (7, 25): 'blue'}), (23, {(30, 21): 'black', (31, 21): 'black', (32, 21): 'black',
570
        (33, 21): 'black', (33, 22): 'black', (34, 22): 'black', (35, 22): 'black', (36, 22): 'black', (37, 22):
571
        'black', (38, 22): 'black', (38, 21): 'black', (38, 20): 'black', (38, 19): 'black', (38, 18): 'black', (38,
572
        17): 'black', (38, 16): 'black', (38, 15): 'black', (38, 14): 'black', (37, 14): 'black', (36, 14): 'black',
573
        (35, 14): 'black', (34, 14): 'black', (33, 14): 'black', (31, 5): 'green', (33, 20): 'green', (4, 19):
574
        'green', (34, 25): 'blue', (33, 25): 'blue', (32, 25): 'blue', (10, 25): 'blue', (9, 25): 'blue', (8, 25): 'blue'}), (23, {(29, 21): 'black', (30, 21): 'black', (31, 21): 'black', (32, 21): 'black', (33, 21):
575
576
        'black', (33, 22): 'black', (34, 22): 'black', (35, 22): 'black', (36, 22): 'black', (37, 22): 'black', (38,
577
       22): 'black', (38, 21): 'black', (38, 20): 'black', (38, 19): 'black', (38, 18): 'black', (38, 17): 'black',
578
        (38, 16): 'black', (38, 15): 'black', (38, 14): 'black', (37, 14): 'black', (36, 14): 'black', (35, 14):
579
        'black', (34, 14): 'black', (31, 5): 'green', (33, 20): 'green', (4, 19): 'green', (34, 25): 'blue', (33,
580
       25): 'blue', (32, 25): 'blue', (10, 25): 'blue', (9, 25): 'blue', (8, 25): 'blue'), (23, {(28, 21): 'black', (29, 21): 'black', (30, 21): 'black', (31, 21): 'black', (32, 21): 'black', (33, 21): 'black', (33, 22):
581
582
        'black', (34, 22): 'black', (35, 22): 'black', (36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (38,
583
584
       21): 'black', (38, 20): 'black', (38, 19): 'black', (38, 18): 'black', (38, 17): 'black', (38, 16): 'black',
        (38, 15): 'black', (38, 14): 'black', (37, 14): 'black', (36, 14): 'black', (35, 14): 'black', (31, 5): 'green', (33, 20): 'green', (4, 19): 'green', (35, 25): 'blue', (34, 25): 'blue', (33, 25): 'blue', (11, 25):
585
586
        'blue', (10, 25): 'blue', (9, 25): 'blue'}), (23, {(27, 21): 'black', (28, 21): 'black', (29, 21): 'black',
587
        (30, 21): 'black', (31, 21): 'black', (32, 21): 'black', (33, 21): 'black', (33, 22): 'black', (34, 22):
588
        'black', (35, 22): 'black', (36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (38, 21): 'black', (38,
589
       20): 'black', (38, 19): 'black', (38, 18): 'black', (38, 17): 'black', (38, 16): 'black', (38, 15): 'black',
590
        (38, 14): 'black', (37, 14): 'black', (36, 14): 'black', (31, 5): 'green', (33, 20): 'green', (4, 19): 'green', (35, 25): 'blue', (34, 25): 'blue', (33, 25): 'blue', (11, 25): 'blue', (10, 25): 'blue', (9, 25):
591
592
        'blue'}), (23, {(27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (29, 21): 'black', (30, 21):
593
        'black', (31, 21): 'black', (32, 21): 'black', (33, 21): 'black', (33, 22): 'black', (34, 22): 'black', (35,
594
        22): 'black', (36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (38, 21): 'black', (38, 20): 'black',
595
        (38, 19): 'black', (38, 18): 'black', (38, 17): 'black', (38, 16): 'black', (38, 15): 'black', (38, 14):
596
597
        'black', (37, 14): 'black', (31, 5): 'green', (33, 20): 'green', (4, 19): 'green', (36, 25): 'blue', (35,
        25): 'blue', (34, 25): 'blue', (12, 25): 'blue', (11, 25): 'blue', (10, 25): 'blue'}), (23, {(28, 20):
598
        'black', (27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (29, 21): 'black', (30, 21): 'black', (31,
599
       21): 'black', (32, 21): 'black', (33, 21): 'black', (33, 22): 'black', (34, 22): 'black', (35, 22): 'black',
600
        (36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (38, 21): 'black', (38, 20): 'black', (38, 19):
601
        'black', (38, 18): 'black', (38, 17): 'black', (38, 16): 'black', (38, 15): 'black', (38, 14): 'black', (31,
602
       5): 'green', (33, 20): 'green', (4, 19): 'green', (36, 25): 'blue', (35, 25): 'blue', (34, 25): 'blue', (12,
```

```
25): 'blue', (11, 25): 'blue', (10, 25): 'blue'}), (23, {(29, 20): 'black', (28, 20): 'black', (27, 20):
        'black', (27, 21): 'black', (28, 21): 'black', (29, 21): 'black', (30, 21): 'black', (31, 21): 'black', (32,
605
       21): 'black', (33, 21): 'black', (33, 22): 'black', (34, 22): 'black', (35, 22): 'black', (36, 22): 'black',
606
        (37, 22): 'black', (38, 22): 'black', (38, 21): 'black', (38, 20): 'black', (38, 19): 'black', (38, 18):
        'black', (38, 17): 'black', (38, 16): 'black', (38, 15): 'black', (31, 5): 'green', (33, 20): 'green', (4,
608
       19): 'green', (37, 25): 'blue', (36, 25): 'blue', (35, 25): 'blue', (13, 25): 'blue', (12, 25): 'blue', (11,
609
       25): 'blue'}), (23, {(30, 20): 'black', (29, 20): 'black', (28, 20): 'black', (27, 20): 'black', (27, 21):
610
       'black', (28, 21): 'black', (29, 21): 'black', (30, 21): 'black', (31, 21): 'black', (32, 21): 'black', (33,
611
612
       21): 'black', (33, 22): 'black', (34, 22): 'black', (35, 22): 'black', (36, 22): 'black', (37, 22): 'black',
       (38, 22): 'black', (38, 21): 'black', (38, 20): 'black', (38, 19): 'black', (38, 18): 'black', (38, 17):
613
        'black', (38, 16): 'black', (31, 5): 'green', (33, 20): 'green', (4, 19): 'green', (37, 25): 'blue', (36,
614
       25): 'blue', (35, 25): 'blue', (13, 25): 'blue', (12, 25): 'blue', (11, 25): 'blue'}), (23, {(31, 20):
615
       'black', (30, 20): 'black', (29, 20): 'black', (28, 20): 'black', (27, 20): 'black', (27, 21): 'black', (28,
616
       21): 'black', (29, 21): 'black', (30, 21): 'black', (31, 21): 'black', (32, 21): 'black', (33, 21): 'black',
617
        (33, 22): 'black', (34, 22): 'black', (35, 22): 'black', (36, 22): 'black', (37, 22): 'black', (38, 22):
618
        'black', (38, 21): 'black', (38, 20): 'black', (38, 19): 'black', (38, 18): 'black', (38, 17): 'black', (31,
619
       5): 'green', (33, 20): 'green', (4, 19): 'green', (38, 25): 'blue', (37, 25): 'blue', (36, 25): 'blue', (14,
620
       25): 'blue', (13, 25): 'blue', (12, 25): 'blue'}), (23, {(32, 20): 'black', (31, 20): 'black', (30, 20):
621
       'black', (29, 20): 'black', (28, 20): 'black', (27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (29,
622
       21): 'black', (30, 21): 'black', (31, 21): 'black', (32, 21): 'black', (33, 21): 'black', (33, 22): 'black',
       (34, 22): 'black', (35, 22): 'black', (36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (38, 21): 'black', (38, 20): 'black', (38, 19): 'black', (38, 18): 'black', (31, 5): 'green', (33, 20): 'green', (4,
624
625
       19): 'green', (38, 25): 'blue', (37, 25): 'blue', (36, 25): 'blue', (14, 25): 'blue', (13, 25): 'blue', (12, 25): 'blue'), (27, {(33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20): 'black', (29, 20): 'black', (28, 20): 'black', (27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (29, 21): 'black', (30, 20): 'black'
626
627
628
       21): 'black', (31, 21): 'black', (32, 21): 'black', (33, 21): 'black', (33, 22): 'black', (34, 22): 'black',
629
       (35, 22): 'black', (36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (38, 21): 'black', (38, 20): 'black', (38, 19): 'black', (31, 5): 'green', (29, 18): 'green', (4, 19): 'green', (39, 25): 'blue', (38,
630
631
       25): 'blue', (37, 25): 'blue', (15, 25): 'blue', (14, 25): 'blue', (13, 25): 'blue'}), (27, {(34, 20):
632
633
       'black', (33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20): 'black', (29, 20): 'black', (28,
634
       20): 'black', (27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (29, 21): 'black', (30, 21): 'black',
       (31, 21): 'black', (32, 21): 'black', (33, 21): 'black', (33, 22): 'black', (34, 22): 'black', (35, 22):
635
636
        'black', (36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (38, 21): 'black', (38, 20): 'black', (38,
637
       19): 'black', (31, 5): 'green', (29, 18): 'green', (4, 19): 'green', (39, 25): 'blue', (38, 25): 'blue', (37,
638
       25): 'blue', (15, 25): 'blue', (14, 25): 'blue', (13, 25): 'blue'}), (27, {(35, 20): 'black', (34, 20):
639
       'black', (33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20): 'black', (29, 20): 'black', (28,
640
       20): 'black', (27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (29, 21): 'black', (30, 21): 'black',
641
        (31, 21): 'black', (32, 21): 'black', (33, 21): 'black', (33, 22): 'black', (34, 22): 'black', (35, 22):
642
        'black', (36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (38, 21): 'black', (38, 20): 'black', (38,
643
       19): 'black', (31, 5): 'green', (29, 18): 'green', (4, 19): 'green', (39, 25): 'blue', (38, 25): 'blue', (16,
644
       25): 'blue', (15, 25): 'blue', (14, 25): 'blue'}), (27, {(36, 20): 'black', (35, 20): 'black', (34, 20):
645
        'black', (33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20): 'black', (29, 20): 'black', (28,
646
       20): 'black', (27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (29, 21): 'black', (30, 21): 'black',
647
       (31, 21): 'black', (32, 21): 'black', (33, 21): 'black', (33, 22): 'black', (34, 22): 'black', (35, 22):
648
       'black', (36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (38, 21): 'black', (38, 20): 'black', (38,
649
       19): 'black', (31, 5): 'green', (29, 18): 'green', (4, 19): 'green', (39, 25): 'blue', (38, 25): 'blue', (16, 25): 'blue', (15, 25): 'blue', (14, 25): 'blue'}), (27, {(37, 20): 'black', (36, 20): 'black', (35, 20):
650
651
652
       'black', (34, 20): 'black', (33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20): 'black', (29,
       20): 'black', (28, 20): 'black', (27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (29, 21): 'black',
653
       (30, 21): 'black', (31, 21): 'black', (32, 21): 'black', (33, 21): 'black', (33, 22): 'black', (34, 22):
654
       'black', (35, 22): 'black', (36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (38, 21): 'black', (38,
       20): 'black', (31, 5): 'green', (29, 18): 'green', (4, 19): 'green', (39, 25): 'blue', (17, 25): 'blue', (16, 25): 'blue', (15, 25): 'blue'}), (27, {(38, 20): 'black', (37, 20): 'black', (36, 20): 'black', (35, 20):
656
657
       'black', (34, 20): 'black', (33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20): 'black', (29,
658
       20): 'black', (28, 20): 'black', (27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (29, 21): 'black',
659
       (30, 21): 'black', (31, 21): 'black', (32, 21): 'black', (33, 21): 'black', (33, 22): 'black', (34, 22):
660
       'black', (35, 22): 'black', (36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (38, 21): 'black', (31,
661
       5): 'green', (29, 18): 'green', (4, 19): 'green', (39, 25): 'blue', (17, 25): 'blue', (16, 25): 'blue', (15, 25): 'blue'}), (27, {(38, 19): 'black', (38, 20): 'black', (37, 20): 'black', (36, 20): 'black', (35, 20):
662
663
       'black', (34, 20): 'black', (33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20): 'black', (29,
664
       20): 'black', (28, 20): 'black', (27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (29, 21): 'black',
665
        (30, 21): 'black', (31, 21): 'black', (32, 21): 'black', (33, 21): 'black', (33, 22): 'black', (34, 22):
666
        'black', (35, 22): 'black', (36, 22): 'black', (37, 22): 'black', (38, 22): 'black', (31, 5): 'green', (29,
667
       18): 'green', (4, 19): 'green', (18, 25): 'blue', (17, 25): 'blue', (16, 25): 'blue', (5, 28): 'blue', (4,
668
       28): 'blue', (3, 28): 'blue'}), (27, {(38, 18): 'black', (38, 19): 'black', (38, 20): 'black', (37, 20):
669
       'black', (36, 20): 'black', (35, 20): 'black', (34, 20): 'black', (33, 20): 'black', (32, 20): 'black', (31,
670
       20): 'black', (30, 20): 'black', (29, 20): 'black', (28, 20): 'black', (27, 20): 'black', (27, 21): 'black',
```

```
(28, 21): 'black', (29, 21): 'black', (30, 21): 'black', (31, 21): 'black', (32, 21): 'black', (33, 21):
672
           'black', (33, 22): 'black', (34, 22): 'black', (35, 22): 'black', (36, 22): 'black', (37, 22): 'black', (31,
673
          5): 'green', (29, 18): 'green', (4, 19): 'green', (18, 25): 'blue', (17, 25): 'blue', (16, 25): 'blue', (5,
674
          28): 'blue', (4, 28): 'blue', (3, 28): 'blue'}), (27, {(38, 17): 'black', (38, 18): 'black', (38, 19):
            'black', (38, 20): 'black', (37, 20): 'black', (36, 20): 'black', (35, 20): 'black', (34, 20): 'black', (33,
676
          20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20): 'black', (29, 20): 'black', (28, 20): 'black',
677
           (27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (29, 21): 'black', (30, 21): 'black', (31, 21):
678
           'black', (32, 21): 'black', (33, 21): 'black', (33, 22): 'black', (34, 22): 'black', (35, 22): 'black', (36,
679
680
           22): 'black', (31, 5): 'green', (29, 18): 'green', (4, 19): 'green', (19, 25): 'blue', (18, 25): 'blue', (17,
          25): 'blue', (6, 28): 'blue', (5, 28): 'blue', (4, 28): 'blue'}), (27, {(38, 16): 'black', (38, 17): 'black',
681
           (38, 18): 'black', (38, 19): 'black', (38, 20): 'black', (37, 20): 'black', (36, 20): 'black', (35, 20): 'black', (34, 20): 'black', (33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20): 'black', (29,
682
           20): 'black', (28, 20): 'black', (27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (29, 21): 'black',
684
           (30, 21): 'black', (31, 21): 'black', (32, 21): 'black', (33, 21): 'black', (33, 22): 'black', (34, 22): 'black', (35, 22): 'black', (31, 5): 'green', (29, 18): 'green', (4, 19): 'green', (19, 25): 'blue', (18,
685
686
           25): 'blue', (17, 25): 'blue', (6, 28): 'blue', (5, 28): 'blue', (4, 28): 'blue'}), (27, {(38, 15): 'black',
687
           (38, 16): 'black', (38, 17): 'black', (38, 18): 'black', (38, 19): 'black', (38, 20): 'black', (37, 20):
688
            'black', (36, 20): 'black', (35, 20): 'black', (34, 20): 'black', (33, 20): 'black', (32, 20): 'black', (31,
689
          20): 'black', (30, 20): 'black', (29, 20): 'black', (28, 20): 'black', (27, 20): 'black', (27, 21): 'black',
690
           (28, 21): 'black', (29, 21): 'black', (30, 21): 'black', (31, 21): 'black', (32, 21): 'black', (33, 21): 'black', (33, 22): 'black', (34, 22): 'black', (31, 5): 'green', (29, 18): 'green', (4, 19): 'green', (20,
692
          25): 'blue', (19, 25): 'blue', (18, 25): 'blue', (7, 28): 'blue', (6, 28): 'blue', (5, 28): 'blue'}), (27,
693
          {(38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38, 17): 'black', (38, 18): 'black', (38, 19): 'black', (38, 20): 'black', (37, 20): 'black', (36, 20): 'black', (35, 20): 'black', (34, 20): 'black', (34, 20): 'black', (35, 20): 'black', (36, 20): 'black', (37, 20): 'black', (38, 20): 'b
694
695
          20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20): 'black', (29, 20): 'black', (28, 20): 'black',
696
           (27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (29, 21): 'black', (30, 21): 'black', (31, 21):
697
           'black', (32, 21): 'black', (33, 21): 'black', (33, 22): 'black', (31, 5): 'green', (29, 18): 'green', (4,
698
           19): 'green', (20, 25): 'blue', (19, 25): 'blue', (18, 25): 'blue', (7, 28): 'blue', (6, 28): 'blue', (5,
699
          28): 'blue'}), (27, {(37, 14): 'black', (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38, 17):
700
701
           'black', (38, 18): 'black', (38, 19): 'black', (38, 20): 'black', (37, 20): 'black', (36, 20): 'black', (35,
702
           20): 'black', (34, 20): 'black', (33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20): 'black',
           (29, 20): 'black', (28, 20): 'black', (27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (29, 21):
703
704
           'black', (30, 21): 'black', (31, 21): 'black', (32, 21): 'black', (33, 21): 'black', (31, 5): 'green', (29,
705
          18): 'green', (4, 19): 'green', (21, 25): 'blue', (20, 25): 'blue', (19, 25): 'blue', (8, 28): 'blue', (7,
706
          28): 'blue', (6, 28): 'blue'}), (27, {(36, 14): 'black', (37, 14): 'black', (38, 14): 'black', (38, 15):
707
           'black', (38, 16): 'black', (38, 17): 'black', (38, 18): 'black', (38, 19): 'black', (38, 20): 'black', (37,
708
          20): 'black', (36, 20): 'black', (35, 20): 'black', (34, 20): 'black', (33, 20): 'black', (32, 20): 'black',
709
           (31, 20): 'black', (30, 20): 'black', (29, 20): 'black', (28, 20): 'black', (27, 20): 'black', (27, 21):
710
           'black', (28, 21): 'black', (29, 21): 'black', (30, 21): 'black', (31, 21): 'black', (32, 21): 'black', (31,
711
          5): 'green', (29, 18): 'green', (4, 19): 'green', (21, 25): 'blue', (20, 25): 'blue', (19, 25): 'blue', (8,
712
          28): 'blue', (7, 28): 'blue', (6, 28): 'blue'}), (27, {(35, 14): 'black', (36, 14): 'black', (37, 14):
713
           'black', (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38, 17): 'black', (38, 18): 'black', (38,
714
           19): 'black', (38, 20): 'black', (37, 20): 'black', (36, 20): 'black', (35, 20): 'black', (34, 20): 'black',
715
           (33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20): 'black', (29, 20): 'black', (28, 20):
716
           'black', (27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (29, 21): 'black', (30, 21): 'black', (31,
717
718
           21): 'black', (31, 5): 'green', (29, 18): 'green', (4, 19): 'green', (22, 25): 'blue', (21, 25): 'blue', (20,
          25): 'blue', (9, 28): 'blue', (8, 28): 'blue', (7, 28): 'blue'}), (27, {(34, 14): 'black', (35, 14): 'black',
719
720
           (36, 14): 'black', (37, 14): 'black', (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38, 17):
            'black', (38, 18): 'black', (38, 19): 'black', (38, 20): 'black', (37, 20): 'black', (36, 20): 'black', (35,
721
          20): 'black', (34, 20): 'black', (33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20): 'black',
722
           (29, 20): 'black', (28, 20): 'black', (27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (29, 21): 'black', (30, 21): 'black', (31, 5): 'green', (29, 18): 'green', (4, 19): 'green', (22, 25): 'blue', (21, 21): 'black', (21, 22): 'black', (22, 23): 'black', (23, 21): 'black', (21, 22): 'black', (22, 23): 'black', (23, 21): 'black
723
724
           25): 'blue', (20, 25): 'blue', (9, 28): 'blue', (8, 28): 'blue', (7, 28): 'blue'}), (27, {(33, 14): 'black',
725
           (34, 14): 'black', (35, 14): 'black', (36, 14): 'black', (37, 14): 'black', (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38, 17): 'black', (38, 18): 'black', (38, 19): 'black', (38, 20): 'black', (37,
726
727
          20): 'black', (36, 20): 'black', (35, 20): 'black', (34, 20): 'black', (33, 20): 'black', (32, 20): 'black',
728
           (31, 20): 'black', (30, 20): 'black', (29, 20): 'black', (28, 20): 'black', (27, 20): 'black', (27, 21):
729
           'black', (28, 21): 'black', (29, 21): 'black', (31, 5): 'green', (29, 18): 'green', (4, 19): 'green', (23,
730
           25): 'blue', (22, 25): 'blue', (21, 25): 'blue', (10, 28): 'blue', (9, 28): 'blue', (8, 28): 'blue'}), (27,
731
          {(32, 14): 'black', (33, 14): 'black', (34, 14): 'black', (35, 14): 'black', (36, 14): 'black', (37, 14): 'black', (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38, 17): 'black', (38, 18): 'b
732
733
           19): 'black', (38, 20): 'black', (37, 20): 'black', (36, 20): 'black', (35, 20): 'black', (34, 20): 'black',
734
           (33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20): 'black', (29, 20): 'black', (28, 20):
735
           'black', (27, 20): 'black', (27, 21): 'black', (28, 21): 'black', (31, 5): 'green', (29, 18): 'green', (4,
736
          19): 'green', (23, 25): 'blue', (22, 25): 'blue', (21, 25): 'blue', (10, 28): 'blue', (9, 28): 'blue', (8, 28): 'blue'}), (27, {(31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (34, 14): 'black', (35, 14):
737
738
           'black', (36, 14): 'black', (37, 14): 'black', (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38,
739
```

```
17): 'black', (38, 18): 'black', (38, 19): 'black', (38, 20): 'black', (37, 20): 'black', (36, 20): 'black',
740
             (35, 20): 'black', (34, 20): 'black', (33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20):
741
             'black', (29, 20): 'black', (28, 20): 'black', (27, 20): 'black', (27, 21): 'black', (31, 5): 'green', (29,
742
            18): 'green', (4, 19): 'green', (24, 25): 'blue', (23, 25): 'blue', (22, 25): 'blue', (11, 28): 'blue', (10, 28): 'blue', (9, 28): 'blue'}), (27, {(30, 14): 'black', (31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (31, 14): 'black', (32, 14): 'black', (33, 14): 'black',
743
744
            'black', (34, 14): 'black', (35, 14): 'black', (36, 14): 'black', (37, 14): 'black', (38, 14): 'black', (38,
745
            15): 'black', (38, 16): 'black', (38, 17): 'black', (38, 18): 'black', (38, 19): 'black', (38, 20): 'black',
746
            (37, 20): 'black', (36, 20): 'black', (35, 20): 'black', (34, 20): 'black', (33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20): 'black', (29, 20): 'black', (28, 20): 'black', (27, 20): 'black', (31, 20): 'bl
747
748
            5): 'green', (29, 18): 'green', (4, 19): 'green', (24, 25): 'blue', (23, 25): 'blue', (22, 25): 'blue', (11,
749
            28): 'blue', (10, 28): 'blue', (9, 28): 'blue'}), (27, {(30, 15): 'black', (30, 14): 'black', (31, 14):
750
             'black', (32, 14): 'black', (33, 14): 'black', (34, 14): 'black', (35, 14): 'black', (36, 14): 'black', (37,
751
            14): 'black', (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38, 17): 'black', (38, 18): 'black',
752
            (38, 19): 'black', (38, 20): 'black', (37, 20): 'black', (36, 20): 'black', (35, 20): 'black', (34, 20): 'black', (33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (30, 20): 'black', (29, 20): 'black', (28,
753
754
            20): 'black', (31, 5): 'green', (29, 18): 'green', (4, 19): 'green', (25, 25): 'blue', (24, 25): 'blue', (23,
755
            25): 'blue', (12, 28): 'blue', (11, 28): 'blue', (10, 28): 'blue'), (27, {(30, 16): 'black', (30, 15):
756
             'black', (30, 14): 'black', (31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (34, 14): 'black', (35,
757
            14): 'black', (36, 14): 'black', (37, 14): 'black', (38, 14): 'black', (38, 15): 'black', (38, 16): 'black',
758
             (38, 17): 'black', (38, 18): 'black', (38, 19): 'black', (38, 20): 'black', (37, 20): 'black', (36, 20):
             'black', (35, 20): 'black', (34, 20): 'black', (33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (30,
760
            20): 'black', (29, 20): 'black', (31, 5): 'green', (29, 18): 'green', (4, 19): 'green', (25, 25): 'blue',
761
            (24, 25): 'blue', (23, 25): 'blue', (12, 28): 'blue', (11, 28): 'blue', (10, 28): 'blue'}), (27, {(30, 17):
762
             'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black', (31, 14): 'black', (32, 14): 'black', (33,
763
            14): 'black', (34, 14): 'black', (35, 14): 'black', (36, 14): 'black', (37, 14): 'black', (38, 14): 'black',
764
             (38, 15): 'black', (38, 16): 'black', (38, 17): 'black', (38, 18): 'black', (38, 19): 'black', (38, 20):
765
            'black', (37, 20): 'black', (36, 20): 'black', (35, 20): 'black', (34, 20): 'black', (33, 20): 'black', (32,
766
767
            20): 'black', (31, 20): 'black', (30, 20): 'black', (31, 5): 'green', (29, 18): 'green', (4, 19): 'green',
            (26, 25): 'blue', (25, 25): 'blue', (24, 25): 'blue', (13, 28): 'blue', (12, 28): 'blue', (11, 28): 'blue'})
768
            (27, {(30, 18): 'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black', (31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (34, 14): 'black', (35, 14): 'black', (36, 14): 'black',
769
770
            (37, 14): 'black', (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38, 17): 'black', (38, 18):
771
            'black', (38, 19): 'black', (38, 20): 'black', (37, 20): 'black', (36, 20): 'black', (35, 20): 'black', (34,
772
            20): 'black', (33, 20): 'black', (32, 20): 'black', (31, 20): 'black', (31, 5): 'green', (29, 18): 'green',
773
774
            (4, 19): 'green', (26, 25): 'blue', (25, 25): 'blue', (24, 25): 'blue', (13, 28): 'blue', (12, 28): 'blue',
775
            (11, 28): 'blue'}), (32, {(29, 18): 'black', (30, 18): 'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black', (31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (34, 14): 'black',
776
777
             (35, 14): 'black', (36, 14): 'black', (37, 14): 'black', (38, 14): 'black', (38, 15): 'black', (38, 16):
778
             'black', (38, 17): 'black', (38, 18): 'black', (38, 19): 'black', (38, 20): 'black', (37, 20): 'black', (36,
779
            20): 'black', (35, 20): 'black', (34, 20): 'black', (33, 20): 'black', (32, 20): 'black', (31, 5): 'green',
780
            (4, 19): 'green', (20, 21): 'green', (27, 25): 'blue', (26, 25): 'blue', (25, 25): 'blue', (14, 28): 'blue', (13, 28): 'blue', (12, 28): 'blue'}), (32, {(29, 19): 'black', (29, 18): 'black', (30, 18): 'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black', (31, 14): 'black', (32, 14): 'black', (31, 14): 'black', (32, 14): 'b
781
782
783
            (33, 14): 'black', (34, 14): 'black', (35, 14): 'black', (36, 14): 'black', (37, 14): 'black', (38, 14):
784
            'black', (38, 15): 'black', (38, 16): 'black', (38, 17): 'black', (38, 18): 'black', (38, 19): 'black', (38,
785
786
            20): 'black', (37, 20): 'black', (36, 20): 'black', (35, 20): 'black', (34, 20): 'black', (33, 20): 'black',
            (32, 20): 'black', (31, 5): 'green', (4, 19): 'green', (20, 21): 'green', (27, 25): 'blue', (26, 25): 'blue',
787
            (25, 25): 'blue', (14, 28): 'blue', (13, 28): 'blue', (12, 28): 'blue')), (32, {(29, 20): 'black', (29, 19):
788
             'black', (29, 18): 'black', (30, 18): 'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30,
789
            14): 'black', (31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (34, 14): 'black', (35, 14): 'black',
790
             (36, 14): 'black', (37, 14): 'black', (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38, 17):
791
792
             'black', (38, 18): 'black', (38, 19): 'black', (38, 20): 'black', (37, 20): 'black', (36, 20): 'black', (35,
            20): 'black', (34, 20): 'black', (33, 20): 'black', (32, 20): 'black', (31, 5): 'green', (4, 19): 'green',
793
            (20, 21): 'green', (28, 25): 'blue', (27, 25): 'blue', (26, 25): 'blue', (15, 28): 'blue', (14, 28): 'blue', (13, 28): 'blue'), (32, {(29, 21): 'black', (29, 20): 'black', (29, 19): 'black', (29, 18): 'black', (30, 18): 'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black', (31, 14): 'b
794
795
796
             (32, 14): 'black', (33, 14): 'black', (34, 14): 'black', (35, 14): 'black', (36, 14): 'black', (37, 14):
797
            'black', (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38, 17): 'black', (38, 18): 'black', (38,
798
            19): 'black', (38, 20): 'black', (37, 20): 'black', (36, 20): 'black', (35, 20): 'black', (34, 20): 'black',
799
            (33, 20): 'black', (32, 20): 'black', (31, 5): 'green', (4, 19): 'green', (20, 21): 'green', (28, 25):
800
            'blue', (27, 25): 'blue', (26, 25): 'blue', (15, 28): 'blue', (14, 28): 'blue', (13, 28): 'blue'}), (32,
801
            {(29, 22): 'black', (29, 21): 'black', (29, 20): 'black', (29, 19): 'black', (29, 18): 'black', (30, 18):
802
             'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black', (31, 14): 'black', (32,
803
            14): 'black', (33, 14): 'black', (34, 14): 'black', (35, 14): 'black', (36, 14): 'black', (37, 14): 'black',
804
             (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38, 17): 'black', (38, 18): 'black', (38, 19):
805
            'black', (38, 20): 'black', (37, 20): 'black', (36, 20): 'black', (35, 20): 'black', (34, 20): 'black', (33,
806
            20): 'black', (31, 5): 'green', (4, 19): 'green', (20, 21): 'green', (29, 25): 'blue', (28, 25): 'blue', (27,
```

```
25): 'blue', (16, 28): 'blue', (15, 28): 'blue', (14, 28): 'blue'), (32, {(29, 23): 'black', (29, 22):
808
             'black', (29, 21): 'black', (29, 20): 'black', (29, 19): 'black', (29, 18): 'black', (30, 18): 'black', (30,
809
             17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black', (31, 14): 'black', (32, 14): 'black',
810
             (33, 14): 'black', (34, 14): 'black', (35, 14): 'black', (36, 14): 'black', (37, 14): 'black', (38, 14):
811
             'black', (38, 15): 'black', (38, 16): 'black', (38, 17): 'black', (38, 18): 'black', (38, 19): 'black', (38,
812
            20): 'black', (37, 20): 'black', (36, 20): 'black', (35, 20): 'black', (34, 20): 'black', (31, 5): 'green',
813
             (4, 19): 'green', (20, 21): 'green', (29, 25): 'blue', (28, 25): 'blue', (27, 25): 'blue', (16, 28): 'blue',
814
            (15, 28): 'blue', (14, 28): 'blue'}), (32, {(28, 23): 'black', (29, 23): 'black', (29, 22): 'black', (29, 21): 'black', (29, 20): 'black', (29, 19): 'black', (29, 18): 'black', (30, 18): 'black', (30, 17): 'black',
815
816
             (30, 16): 'black', (30, 15): 'black', (30, 14): 'black', (31, 14): 'black', (32, 14): 'black', (33, 14):
817
             'black', (34, 14): 'black', (35, 14): 'black', (36, 14): 'black', (37, 14): 'black', (38, 14): 'black', (38,
818
             15): 'black', (38, 16): 'black', (38, 17): 'black', (38, 18): 'black', (38, 19): 'black', (38, 20): 'black',
819
             (37, 20): 'black', (36, 20): 'black', (35, 20): 'black', (31, 5): 'green', (4, 19): 'green', (20, 21):
820
             'green', (30, 25): 'blue', (29, 25): 'blue', (28, 25): 'blue', (17, 28): 'blue', (16, 28): 'blue', (15, 28): 'blue'}), (32, {(27, 23): 'black', (28, 23): 'black', (29, 23): 'black', (29, 22): 'black', (29, 21):
821
822
             'black', (29, 20): 'black', (29, 19): 'black', (29, 18): 'black', (30, 18): 'black', (30, 17): 'black', (30,
823
             16): 'black', (30, 15): 'black', (30, 14): 'black', (31, 14): 'black', (32, 14): 'black', (33, 14): 'black',
824
             (34, 14): 'black', (35, 14): 'black', (36, 14): 'black', (37, 14): 'black', (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38, 17): 'black', (38, 18): 'black', (38, 19): 'black', (38, 20): 'black', (37,
825
826
            20): 'black', (36, 20): 'black', (31, 5): 'green', (4, 19): 'green', (20, 21): 'green', (30, 25): 'blue',
827
             (29, 25): 'blue', (28, 25): 'blue', (17, 28): 'blue', (16, 28): 'blue', (15, 28): 'blue'}), (32, {(26, 23):
828
             'black', (27, 23): 'black', (28, 23): 'black', (29, 23): 'black', (29, 22): 'black', (29, 21): 'black', (29,
829
             20): 'black', (29, 19): 'black', (29, 18): 'black', (30, 18): 'black', (30, 17): 'black', (30, 16): 'black',
830
             (30, 15): 'black', (30, 14): 'black', (31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (34, 14): 'black', (35, 14): 'black', (36, 14): 'black', (37, 14): 'black', (38, 14): 'black', (38, 15): 'bl
831
832
             16): 'black', (38, 17): 'black', (38, 18): 'black', (38, 19): 'black', (38, 20): 'black', (37, 20): 'black',
833
             (31, 5): 'green', (4, 19): 'green', (20, 21): 'green', (31, 25): 'blue', (30, 25): 'blue', (29, 25): 'blue', (18, 28): 'blue', (17, 28): 'blue', (16, 28): 'blue'}), (32, {(25, 23): 'black', (26, 23): 'black', (27, 23): 'black', (28, 28): 'bl
834
835
              'black', (28, 23): 'black', (29, 23): 'black', (29, 22): 'black', (29, 21): 'black', (29, 20): 'black', (29,
836
837
            19): 'black', (29, 18): 'black', (30, 18): 'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black',
838
             (30, 14): 'black', (31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (34, 14): 'black', (35, 14):
              'black', (36, 14): 'black', (37, 14): 'black', (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38,
839
             17): 'black', (38, 18): 'black', (38, 19): 'black', (38, 20): 'black', (31, 5): 'green', (4, 19): 'green',
840
            (20, 21): 'green', (31, 25): 'blue', (30, 25): 'blue', (29, 25): 'blue', (18, 28): 'blue', (17, 28): 'blue', (16, 28): 'blue')), (32, {(24, 23): 'black', (25, 23): 'black', (26, 23): 'black', (27, 23): 'black', (28,
841
842
843
            23): 'black', (29, 23): 'black', (29, 22): 'black', (29, 21): 'black', (29, 20): 'black', (29, 19): 'black',
844
             (29, 18): 'black', (30, 18): 'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14):
845
             'black', (31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (34, 14): 'black', (35, 14): 'black', (36,
846
             14): 'black', (37, 14): 'black', (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38, 17): 'black',
847
             (38, 18): 'black', (38, 19): 'black', (31, 5): 'green', (4, 19): 'green', (20, 21): 'green', (32, 25):
848
             'blue', (31, 25): 'blue', (30, 25): 'blue', (19, 28): 'blue', (18, 28): 'blue', (17, 28): 'blue'}), (32,
849
            {(23, 23): 'black', (24, 23): 'black', (25, 23): 'black', (26, 23): 'black', (27, 23): 'black', (28, 23): 'black', (29, 23): 'black', (29, 21): 'black', (29, 20): 'black', (29, 19): 'black', (29, 21): 'b
850
851
             18): 'black', (30, 18): 'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black',
852
             (31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (34, 14): 'black', (35, 14): 'black', (36, 14):
853
854
              'black', (37, 14): 'black', (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38, 17): 'black', (38,
             18): 'black', (31, 5): 'green', (4, 19): 'green', (20, 21): 'green', (32, 25): 'blue', (31, 25): 'blue', (30,
855
            25): 'blue', (19, 28): 'blue', (18, 28): 'blue', (17, 28): 'blue')), (32, {(22, 23): 'black', (23, 23):
856
             'black', (24, 23): 'black', (25, 23): 'black', (26, 23): 'black', (27, 23): 'black', (28, 23): 'black', (29,
857
            23): 'black', (29, 22): 'black', (29, 21): 'black', (29, 20): 'black', (29, 19): 'black', (29, 18): 'black',
858
             (30, 18): 'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black', (31, 14):
859
              'black', (32, 14): 'black', (33, 14): 'black', (34, 14): 'black', (35, 14): 'black', (36, 14): 'black', (37,
860
             14): 'black', (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (38, 17): 'black', (31, 5): 'green',
861
            (4, 19): 'green', (20, 21): 'green', (33, 25): 'blue', (32, 25): 'blue', (31, 25): 'blue', (20, 28): 'blue', (19, 28): 'blue', (18, 28): 'blue'}), (32, {(21, 23): 'black', (22, 23): 'black', (23, 23): 'black', (24, 23): 'black', (25, 23): 'black', (26, 23): 'black', (27, 23): 'black', (28, 23): 'black', (29, 23): 'black', (29, 23): 'black', (20, 23): 'b
862
863
864
             (29, 22): 'black', (29, 21): 'black', (29, 20): 'black', (29, 19): 'black', (29, 18): 'black', (30, 18):
865
             'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black', (31, 14): 'black', (32,
866
             14): 'black', (33, 14): 'black', (34, 14): 'black', (35, 14): 'black', (36, 14): 'black', (37, 14): 'black',
867
             (38, 14): 'black', (38, 15): 'black', (38, 16): 'black', (31, 5): 'green', (4, 19): 'green', (20, 21):
868
              'green', (33, 25): 'blue', (32, 25): 'blue', (31, 25): 'blue', (20, 28): 'blue', (19, 28): 'blue', (18, 28): 'blue'}), (32, {(21, 22): 'black', (21, 23): 'black', (22, 23): 'black', (23, 23): 'black', (24, 23):
869
870
             'black', (25, 23): 'black', (26, 23): 'black', (27, 23): 'black', (28, 23): 'black', (29, 23): 'black', (29,
871
            22): 'black', (29, 21): 'black', (29, 20): 'black', (29, 19): 'black', (29, 18): 'black', (30, 18): 'black',
872
             (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black', (31, 14): 'black', (32, 14):
873
             'black', (33, 14): 'black', (34, 14): 'black', (35, 14): 'black', (36, 14): 'black', (37, 14): 'black', (38,
874
            14): 'black', (38, 15): 'black', (31, 5): 'green', (4, 19): 'green', (20, 21): 'green', (34, 25): 'blue',
```

```
(33, 25): 'blue', (32, 25): 'blue', (21, 28): 'blue', (20, 28): 'blue', (19, 28): 'blue'}), (32, {(21, 21):
876
             'black', (21, 22): 'black', (21, 23): 'black', (22, 23): 'black', (23, 23): 'black', (24, 23): 'black', (25,
877
            23): 'black', (26, 23): 'black', (27, 23): 'black', (28, 23): 'black', (29, 23): 'black', (29, 22): 'black',
878
            (29, 21): 'black', (29, 20): 'black', (29, 19): 'black', (29, 18): 'black', (30, 18): 'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black', (31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (31, 14): 'bl
879
880
            14): 'black', (34, 14): 'black', (35, 14): 'black', (36, 14): 'black', (37, 14): 'black', (38, 14): 'black',
881
            (31, 5): 'green', (4, 19): 'green', (20, 21): 'green', (34, 25): 'blue', (33, 25): 'blue', (32, 25): 'blue',
882
            (21, 28): 'blue', (20, 28): 'blue', (19, 28): 'blue'}), (32, {(21, 20): 'black', (21, 21): 'black', (21, 22):
883
884
             'black', (21, 23): 'black', (22, 23): 'black', (23, 23): 'black', (24, 23): 'black', (25, 23): 'black', (26,
            23): 'black', (27, 23): 'black', (28, 23): 'black', (29, 23): 'black', (29, 22): 'black', (29, 21): 'black',
885
            (29, 20): 'black', (29, 19): 'black', (29, 18): 'black', (30, 18): 'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black', (31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (34,
886
887
            14): 'black', (35, 14): 'black', (36, 14): 'black', (37, 14): 'black', (31, 5): 'green', (4, 19): 'green',
888
            (20, 21): 'green', (35, 25): 'blue', (34, 25): 'blue', (33, 25): 'blue', (22, 28): 'blue', (21, 28): 'blue', (20, 28): 'blue'}), (32, {(20, 20): 'black', (21, 20): 'black', (21, 21): 'black', (21, 22): 'black', (21, 21): 'black', (21, 22): 'black', (21, 21): '
889
890
            23): 'black', (22, 23): 'black', (23, 23): 'black', (24, 23): 'black', (25, 23): 'black', (26, 23): 'black',
891
            (27, 23): 'black', (28, 23): 'black', (29, 23): 'black', (29, 22): 'black', (29, 21): 'black', (29, 20):
892
             'black', (29, 19): 'black', (29, 18): 'black', (30, 18): 'black', (30, 17): 'black', (30, 16): 'black', (30,
893
            15): 'black', (30, 14): 'black', (31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (34, 14): 'black',
894
             (35, 14): 'black', (36, 14): 'black', (31, 5): 'green', (4, 19): 'green', (20, 21): 'green', (35, 25):
             'blue', (34, 25): 'blue', (33, 25): 'blue', (22, 28): 'blue', (21, 28): 'blue', (20, 28): 'blue'}), (32,
896
            {(19, 20): 'black', (20, 20): 'black', (21, 20): 'black', (21, 21): 'black', (21, 22): 'black', (21, 23):
897
            'black', (22, 23): 'black', (23, 23): 'black', (24, 23): 'black', (25, 23): 'black', (26, 23): 'black', (27,
898
            23): 'black', (28, 23): 'black', (29, 23): 'black', (29, 22): 'black', (29, 21): 'black', (29, 20): 'black',
899
            (29, 19): 'black', (29, 18): 'black', (30, 18): 'black', (30, 17): 'black', (30, 16): 'black', (30, 15):
900
             'black', (30, 14): 'black', (31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (34, 14): 'black', (35,
901
            14): 'black', (31, 5): 'green', (4, 19): 'green', (20, 21): 'green', (36, 25): 'blue', (35, 25): 'blue', (34,
902
            25): 'blue', (23, 28): 'blue', (22, 28): 'blue', (21, 28): 'blue'}), (32, {(19, 21): 'black', (19, 20):
            'black', (20, 20): 'black', (21, 20): 'black', (21, 21): 'black', (21, 22): 'black', (21, 23): 'black', (22,
904
            23): 'black', (23, 23): 'black', (24, 23): 'black', (25, 23): 'black', (26, 23): 'black', (27, 23): 'black',
905
906
             (28, 23): 'black', (29, 23): 'black', (29, 22): 'black', (29, 21): 'black', (29, 20): 'black', (29, 19):
             'black', (29, 18): 'black', (30, 18): 'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30,
907
            14): 'black', (31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (34, 14): 'black', (31, 5): 'green',
908
            (4, 19): 'green', (20, 21): 'green', (36, 25): 'blue', (35, 25): 'blue', (34, 25): 'blue', (23, 28): 'blue', (22, 28): 'blue', (21, 28): 'blue'}), (32, {(19, 22): 'black', (19, 21): 'black', (19, 20): 'black', (20,
909
910
            20): 'black', (21, 20): 'black', (21, 21): 'black', (21, 22): 'black', (21, 23): 'black', (22, 23): 'black',
911
912
            (23, 23): 'black', (24, 23): 'black', (25, 23): 'black', (26, 23): 'black', (27, 23): 'black', (28, 23):
913
            'black', (29, 23): 'black', (29, 22): 'black', (29, 21): 'black', (29, 20): 'black', (29, 19): 'black', (29,
914
            18): 'black', (30, 18): 'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black',
915
            (31, 14): 'black', (32, 14): 'black', (33, 14): 'black', (31, 5): 'green', (4, 19): 'green', (20, 21):
916
              'green', (37, 25): 'blue', (36, 25): 'blue', (35, 25): 'blue', (24, 28): 'blue', (23, 28): 'blue', (22, 28):
917
             'blue'}), (32, {(19, 23): 'black', (19, 22): 'black', (19, 21): 'black', (19, 20): 'black', (20, 20):
918
            'black', (21, 20): 'black', (21, 21): 'black', (21, 22): 'black', (21, 23): 'black', (22, 23): 'black', (23,
919
            23): 'black', (24, 23): 'black', (25, 23): 'black', (26, 23): 'black', (27, 23): 'black', (28, 23): 'black',
920
            (29, 23): 'black', (29, 22): 'black', (29, 21): 'black', (29, 20): 'black', (29, 19): 'black', (29, 18):
921
922
             'black', (30, 18): 'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black', (31,
            14): 'black', (32, 14): 'black', (31, 5): 'green', (4, 19): 'green', (20, 21): 'green', (37, 25): 'blue',
923
924
            (36, 25): 'blue', (35, 25): 'blue', (24, 28): 'blue', (23, 28): 'blue', (22, 28): 'blue'}), (32, {(19, 24):
925
             'black', (19, 23): 'black', (19, 22): 'black', (19, 21): 'black', (19, 20): 'black', (20, 20): 'black', (21,
            20): 'black', (21, 21): 'black', (21, 22): 'black', (21, 23): 'black', (22, 23): 'black', (23, 23): 'black',
926
             (24, 23): 'black', (25, 23): 'black', (26, 23): 'black', (27, 23): 'black', (28, 23): 'black', (29, 23):
927
928
             'black', (29, 22): 'black', (29, 21): 'black', (29, 20): 'black', (29, 19): 'black', (29, 18): 'black', (30,
            18): 'black', (30, 17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black', (31, 14): 'black',
929
            (31, 5): 'green', (4, 19): 'green', (20, 21): 'green', (38, 25): 'blue', (37, 25): 'blue', (36, 25): 'blue', (25, 28): 'blue', (24, 28): 'blue', (23, 28): 'blue'}), (32, {(19, 25): 'black', (19, 24): 'black', (19, 23):
930
931
            'black', (19, 22): 'black', (19, 21): 'black', (19, 20): 'black', (20, 20): 'black', (21, 20): 'black', (21,
932
            21): 'black', (21, 22): 'black', (21, 23): 'black', (22, 23): 'black', (23, 23): 'black', (24, 23): 'black',
933
            (25, 23): 'black', (26, 23): 'black', (27, 23): 'black', (28, 23): 'black', (29, 23): 'black', (29, 22): 'black', (29, 21): 'black', (29, 20): 'black', (29, 19): 'black', (29, 18): 'black', (30, 18): 'bl
934
935
            17): 'black', (30, 16): 'black', (30, 15): 'black', (30, 14): 'black', (31, 5): 'green', (4, 19): 'green',
936
            (20, 21): 'green', (38, 25): 'blue', (37, 25): 'blue', (36, 25): 'blue', (25, 28): 'blue', (24, 28): 'blue', (23, 28): 'blue'), (32, {(19, 26): 'black', (19, 25): 'black', (19, 24): 'black', (19, 23): 'black', (19, 22): 'black', (19, 21): 'black', (19, 20): 'black', (20, 20): 'black', (21, 20): 'black', (21, 21): 'b
937
938
939
            (21, 22): 'black', (21, 23): 'black', (22, 23): 'black', (23, 23): 'black', (24, 23): 'black', (25, 23):
940
             'black', (26, 23): 'black', (27, 23): 'black', (28, 23): 'black', (29, 23): 'black', (29, 22): 'black', (29,
941
            21): 'black', (29, 20): 'black', (29, 19): 'black', (29, 18): 'black', (30, 18): 'black', (30, 17): 'black',
942
            (30, 16): 'black', (30, 15): 'black', (31, 5): 'green', (4, 19): 'green', (20, 21): 'green', (39, 25):
```

```
'blue', (38, 25): 'blue', (37, 25): 'blue', (26, 28): 'blue', (25, 28): 'blue', (24, 28): 'blue'}), (32,
944
                 {(19, 27): 'black', (19, 26): 'black', (19, 25): 'black', (19, 24): 'black', (19, 23): 'black', (19, 22): 'black', (19, 21): 'black', (19, 20): 'black', (21, 20): 'black', (21, 21): 'b
945
946
                 22): 'black', (21, 23): 'black', (22, 23): 'black', (23, 23): 'black', (24, 23): 'black', (25, 23): 'black',
                  (26, 23): 'black', (27, 23): 'black', (28, 23): 'black', (29, 23): 'black', (29, 22): 'black', (29, 21):
948
                  'black', (29, 20): 'black', (29, 19): 'black', (29, 18): 'black', (30, 18): 'black', (30, 17): 'black', (30,
949
                  16): 'black', (31, 5): 'green', (4, 19): 'green', (20, 21): 'green', (39, 25): 'blue', (38, 25): 'blue', (37,
950
                 25): 'blue', (26, 28): 'blue', (25, 28): 'blue', (24, 28): 'blue'), (32, {(19, 28): 'black', (19, 27):
951
952
                  'black', (19, 26): 'black', (19, 25): 'black', (19, 24): 'black', (19, 23): 'black', (19, 22): 'black', (19,
                  21): 'black', (19, 20): 'black', (20, 20): 'black', (21, 20): 'black', (21, 21): 'black', (21, 22): 'black',
953
                  (21, 23): 'black', (22, 23): 'black', (23, 23): 'black', (24, 23): 'black', (25, 23): 'black', (26, 23): 'black', (27, 23): 'black', (28, 23): 'black', (29, 23): 'black', (29, 23): 'black', (29, 21): 'bl
954
955
                  20): 'black', (29, 19): 'black', (29, 18): 'black', (30, 18): 'black', (30, 17): 'black', (31, 5): 'green',
956
                 (4, 19): 'green', (20, 21): 'green', (39, 25): 'blue', (38, 25): 'blue', (27, 28): 'blue', (26, 28): 'blue', (25, 28): 'blue'), (32, {(19, 29): 'black', (19, 28): 'black', (19, 27): 'black', (19, 26): 'black', (19, 27): 'black', (19, 28): 'b
957
958
                 25): 'black', (19, 24): 'black', (19, 23): 'black', (19, 22): 'black', (19, 21): 'black', (19, 20): 'black',
959
                  (20, 20): 'black', (21, 20): 'black', (21, 21): 'black', (21, 22): 'black', (21, 23): 'black', (22, 23):
960
                   'black', (23, 23): 'black', (24, 23): 'black', (25, 23): 'black', (26, 23): 'black', (27, 23): 'black', (28,
961
                 23): 'black', (29, 23): 'black', (29, 22): 'black', (29, 21): 'black', (29, 20): 'black', (29, 19): 'black',
962
                  (29, 18): 'black', (30, 18): 'black', (31, 5): 'green', (4, 19): 'green', (20, 21): 'green', (39, 25):
                   'blue', (38, 25): 'blue', (27, 28): 'blue', (26, 28): 'blue', (25, 28): 'blue'}), (32, {(19, 29): 'black',
964
                  (19, 28): 'black', (19, 27): 'black', (19, 26): 'black', (19, 25): 'black', (19, 24): 'black', (19, 23):
965
                  'black', (19, 22): 'black', (19, 21): 'black', (19, 20): 'black', (20, 20): 'black', (21, 20): 'black', (21,
966
                 21): 'black', (21, 22): 'black', (21, 23): 'black', (22, 23): 'black', (23, 23): 'black', (24, 23): 'black',
967
                  (25, 23): 'black', (26, 23): 'black', (27, 23): 'black', (28, 23): 'black', (29, 23): 'black', (29, 22):
968
                  'black', (29, 21): 'black', (29, 20): 'black', (29, 19): 'black', (29, 18): 'black', (31, 5): 'green', (4,
969
                 19): 'green', (20, 21): 'green', (39, 25): 'blue', (28, 28): 'blue', (27, 28): 'blue', (26, 28): 'blue'})]
970
```

7 snake.py

```
1
    # FILE : snake.py
   # WRITERS : Nimrod Mallis, mallis ; Dor Kaiser, dor_kaiser
   # EXERCISE : intro2cs1 ex10 2023
4
    # DESCRIPTION: Implements the Snake for the Snake Game
   # STUDENTS I DISCUSSED THE EXERCISE WITH: N/A
   # WEB PAGES I USED: N/A
    # NOTES: N/A
    9
10
11
    from typing import List
    from common import BaseDynamicGameObject, Coordinate, Direction
12
13
    class Snake(BaseDynamicGameObject):
14
15
        Represents the Snake Game Object in the game.
16
        The object is intended to be placed on a board, although it's not mandatory.
17
18
19
        def __init__(self, location: Coordinate, length: int = 3) -> None:
20
21
            Creates a new Snake with its head in the given coordinate and initial given length.
22
23
           The Snake starts at an upwards orientation, with the starting coordinates being
           from head to the tail perpendicular to the X-Axis.
24
25
           \# The coordinates which the snake occupies, the order of this list is
26
27
               integral to the operation of the object
           super().__init__(Direction.UP, self._get_initial_position(location, length), "black")
28
29
            self.\_expansion = 0
           self._to_split = None
30
31
        def move(self) -> bool:
33
34
           Moving the Snake in the set direction.
           To set the direction use change_direction.
35
36
37
           If the Snake is during expansion phase, it will not empty
            the coordinates it occupies until the expansion phase is done.
38
39
            (e.g. the tail is not removed during this phase)
40
           if self._to_split is not None:
41
42
               self._coordinates = self._coordinates[:self._to_split]
43
                self._to_split = None
44
45
           new_coordinate = self.movement_requirements()
           self._coordinates.insert(0, new_coordinate) # Adding the new head
46
47
            # If expanding the snake length is not necessary, remove the tail element
           if 0 == self._expansion:
49
50
               self._coordinates.pop() # Removing the last element (the tail)
            else: # Otherwise, expand by 1 and decrement the expanion ratio
51
               self._expansion -= 1
52
53
            # Moving the snake cannot fail
54
55
           return True
        def change_direction(self, direction: Direction) -> bool:
57
58
           Changes the direction the snake is moving to.
```

```
60
             If the direction is invalid False is returned, True otherwise.
61
             \# Checking if the direction is in the valid directions
62
                 for the current snake orientation
63
             if direction not in self._get_valid_directions():
64
65
                 return False
66
             # Updating the current direction
67
68
             self._current_direction = direction
             return True
69
70
71
         def expand(self, amount: int) -> None:
72
             Ordering the Snake to expand on the next moves by the given amount.
73
74
             Expansion process is taking 'amount' of calls to move to fully complete.
75
76
             self._expansion += amount
77
         def split(self, coordinate: Coordinate) -> bool:
78
79
             Ordering the Snake to split part of its body from the given
80
             coordinate\ ({\it included})\ to\ the\ tail
81
             Returns True whether the split would be successful, False otherwise
82
83
84
             self._to_split = self._coordinates.index(coordinate)
85
             return not (self._to_split in [0,1])
86
         def _get_valid_directions(self) -> List[Direction]:
87
88
89
             Returns the valid directions for the snake to move at its current orientation.
90
             It is not permitted for the snake to move down (up) if it's oriented up (down)
             or to move left (right) if it's oriented right (left).
91
92
93
             if (Direction.LEFT == self._current_direction) or \
                      (Direction.RIGHT == self._current_direction):
94
95
                  return [Direction.UP, Direction.DOWN, self._current_direction]
96
             else: # Up or Down
                 return [Direction.RIGHT, Direction.LEFT, self._current_direction]
97
98
         def _get_initial_position(self, head_location: Coordinate, length: int) -> List[Coordinate]:
99
100
             Returns the initial coordinates considering the location of the head and length.
101
             The function expects that the snake would be oriented \mbox{\it UPWARDS}.
102
103
              # Initializing the starting coordinates of the snake to be under the head location
104
105
             return [Coordinate(head_location.row-index, head_location.column)
106
                          for index in range(length)]
```

8 snake game.py

```
# FILE : snake_game.py
   # WRITERS : Nimrod Mallis, mallis ; Dor Kaiser, dor_kaiser
   # EXERCISE : intro2cs1 ex10 2023
   # DESCRIPTION: Implements the primary game functionality
   # STUDENTS I DISCUSSED THE EXERCISE WITH: N/A
   # WEB PAGES I USED: N/A
    # NOTES: N/A
   9
10
    # External imports
11
   import math
12
   from typing import Optional
   from game_display import GameDisplay
14
15
   from game_utils import get_random_apple_data, get_random_wall_data
16
   # Internal imports
17
18
   from common import Coordinate, BaseGameObject, Direction
19
   from snake import Snake
   from board import Board
20
21
    from apple import Apple
   from wall import Wall
22
23
    class SnakeGame:
24
25
26
        The primary game object for Snake
27
28
29
       def __init__(
30
               self,
               board: Board,
31
               snake: Optional[Snake],
               max_apples: int,
33
34
               max_walls: int,
               max_rounds: int) -> None:
35
36
37
           Creates a new Snake Game with the given Board and Snake.
           max_apples and max_walls specify the maximum amount of
38
               apples and walls to be on the wall at a given time.
39
40
           max_rounds specifies when the game will terminate automatically
           if it didn't terminate naturally.
41
42
43
           self._new_direction = None
44
45
           self._snake = snake
           self._board = board
46
47
           # Apple info
           self._max_apples = max_apples
49
50
           self._current_apples = 0
51
           # Wall info
52
53
           self._max_walls = max_walls
           self._current_walls = 0
54
55
           self._score = 0
           self._is_over = False
           self._current_round = 0
57
58
           self._max_rounds = max_rounds
```

```
60
              # If we received a snake object add it to the board
             if self._snake is not None:
61
62
                  self._board.add_game_object(self._snake)
63
              # Edge case: Adding apples and walls on the Snake itself will cause
64
65
              # them not to generate in the first round
66
              self._add_apples_and_walls()
67
68
         def set_snake_direction(self, direction: Optional[Direction]) -> None:
69
              Setting the moving direction of the snake.
70
71
72
              self._new_direction = direction
73
74
         def add_points(self) -> None:
75
76
              Adding to the score tally.
              The points are calculated by the floor of the
77
                 square root of the snake's current length.
78
79
              self._score += math.floor(math.sqrt(len(self._snake)))
80
81
         def update_objects(self) -> None:
82
83
84
85
              # Changing the direction of the snake if we have a snake
             if self._snake is not None:
86
87
                  self._snake.change_direction(self._new_direction)
88
89
              self._board.move_game_objects(self._interaction_callback,
90
                                           self._out_of_bounds_callback)
91
92
              self._add_apples_and_walls()
93
         def _out_of_bounds_callback(self, source: BaseGameObject, off_board: bool) -> None:
94
95
96
              Deals with all interactions of game objects and the game board.
97
             Called from within Board.
98
              if type(source) is Wall and off_board:
99
                  # Wall is completely off the board, we should remove it
100
                  self._board.remove_game_object(source)
101
                  self._current_walls -= 1
102
103
              if type(source) is Snake:
104
                  # The snake met with the board boundries, we end the game
105
106
                  self._set_is_over()
107
108
         def _interaction_callback(self, source: BaseGameObject, dest: BaseGameObject) -> None:
109
              Deals with all interactions of the game objects with one another.
110
              {\it Called from within Board.}
111
112
113
              if type(dest) is Apple and type(source) is Snake:
                  # The snake with an apple, remove it,
114
                     increment the score and expand the snake
115
116
                 self.add_points()
117
                  self._board.remove_game_object(dest)
                  self._current_apples -= 1
118
119
                  self._snake.expand(3)
120
121
              if type(dest) is Apple and type(source) is Wall:
                  # The wall met with an apple, destroy it
122
                  self._board.remove_game_object(dest)
123
124
                  self._current_apples -= 1
125
              if type(dest) is Snake and type(source) is Wall:
126
127
                  # The wall met with a snake, split the snake
```

```
128
                  # If the split left the snake with 0 or 1 cells,
129
                     the game ends
130
                  if not dest.split(source.get_coordinates()[0]):
                      self._set_is_over()
131
132
              # Ending the game if the snake hits a wall, or hit itself
133
              if (type(dest) is Snake or type(dest) is Wall) and type(source) is Snake:
134
                  self._set_is_over()
135
136
         def _wall_move_callback(self) -> bool:
137
138
139
              Deals with the conditions for the wall to move.
              Currently the wall moves if and only if the round is even number.
140
141
142
              return 0 == (self._current_round % 2)
143
144
         def _add_apples_and_walls(self) -> None:
145
              Adds apples and walls to the wall, if any are missing.
146
              The coordinates for each are generated randomly, if the
147
                  generated coordinates overlap with any other object on the game
148
                  board the addition fails and it is not tried again
149
150
151
              # Adding walls if missing any
152
              if self._max_walls > self._current_walls:
153
                  x_coord, y_coord, direction = get_random_wall_data()
                  if self._board.add_game_object(
154
155
                      Wall(Coordinate.from_legacy_coordinate((x_coord, y_coord)),
156
                           direction,
157
                           self._wall_move_callback)):
158
                      self._current_walls += 1
159
160
              # Adding apples if missing
161
              if self._max_apples > self._current_apples:
                  if self._board.add_game_object(
162
163
                      Apple(Coordinate.from_legacy_coordinate(get_random_apple_data()))):
164
                      self._current_apples += 1
165
         def draw_board(self, gui: GameDisplay) -> None:
166
167
168
              Drawing the current board on the screen and setting the score value
169
170
              self. board.draw board(gui)
171
              gui.show_score(self._score)
172
173
         def end_round(self) -> None:
174
              Finishing the current round
175
176
              Placing apples and walls if needed, and incrementing the round counter
177
              self._current_round += 1
178
179
180
         def _set_is_over(self) -> None:
181
              Setting the is_over flag, signaling that the game should end
182
183
184
              self._is_over = True
185
         def is_over(self) -> bool:
186
187
188
              Returns if any of the end conditions were satisfied.
189
              Possible ending routes:
                  1) The max round has been reached.
190
                  2) The Snake met with a wall or the board boundries
191
                 3) The Snake was split to an invalid length (currently 1 or 0)
192
193
             return self._is_over or (self._current_round > self._max_rounds if self._max_rounds != -1 else False)
194
```

9 snake main.py

```
1
    # FILE : snake_main.py
   # WRITERS : Nimrod Mallis, mallis ; Dor Kaiser, dor_kaiser
   # EXERCISE : intro2cs1 ex10 2023
4
    # DESCRIPTION: Runs the Snake Game
   # STUDENTS I DISCUSSED THE EXERCISE WITH: N/A
   # WEB PAGES I USED: N/A
    # NOTES: N/A
   9
10
    # External imports
11
   import argparse
12
   from game_display import GameDisplay
13
14
    # Internal imports
15
   from snake_game import SnakeGame
16
   from common import Coordinate
17
18
    from board import Board
19
   from snake import Snake
20
21
    def _initialize_game(args: argparse.Namespace) -> SnakeGame:
22
23
        Initializing the game with the given arguments.
24
       board = Board(Coordinate(args.height, args.width))
25
26
27
        # If game is not on debug mode, create the snake
       snake = None
28
29
       if not args.debug:
           snake = Snake(Coordinate(args.height//2, args.width//2))
30
31
        return SnakeGame(board, snake, args.apples, args.walls, args.rounds)
33
    def main_loop(gd: GameDisplay, args: argparse.Namespace) -> None:
34
35
        Runs the Snake Game with the given arguments and Game Display.
36
37
       game = _initialize_game(args)
38
39
       gd.show_score(0)
40
       # ROUND O STARTS HERE
41
42
       # No movements are made in the round O
       game.draw_board(gd)
43
       game.end_round()
44
       gd.end_round()
45
       # ROUND O ENDS HERE
46
47
        # Begin primary game loop
48
       while not game.is_over():
49
50
            # Changing the direction of the snake if necessary
51
           game.set_snake_direction(gd.get_key_clicked())
52
53
            # Updating the objects on the game board
           game.update_objects()
54
55
           # Round finalization
           # Drawing the board and ending the round
57
           game.draw_board(gd)
           game.end_round()
```

10 wall.py

```
# FILE : wall.py
   # WRITERS : Nimrod Mallis, mallis ; Dor Kaiser, dor_kaiser
   # EXERCISE : intro2cs1 ex10 2023
    # DESCRIPTION: Implements the Wall for the Snake Game
   # STUDENTS I DISCUSSED THE EXERCISE WITH: N/A
   # WEB PAGES I USED: N/A
    # NOTES: N/A
   9
10
    from typing import List, Callable
11
    from common import BaseDynamicGameObject, Coordinate, Direction
12
    # Callback passed to the wall, it is called when the wall is requested to move
14
    \# and the callback should determine whether the movement should occur, so
15
    # a True should be returned if the move is permitted, False otherwise
16
    MoveCallback = Callable[[], bool]
17
18
    class Wall(BaseDynamicGameObject):
19
20
21
        Represents a wall in the Snake Game.
        A wall is a moving object, moving perpendicular to the Y or X Axis,
22
23
        depending on its starting orientation.
24
25
26
        def __init__(
           self, location: Coordinate, direction: Direction, move_callback: MoveCallback) -> None:
27
28
            Creates a new wall at the given location and orientation.
29
            The received location is a coordinate of where the middle of the wall lies.
30
31
           The move\_callback is called when the wall is requested to be moved.
            super().__init__(direction, self._get_initial_position(location, direction), "blue")
33
34
            self._move_callback = move_callback
35
        def move(self) -> bool:
36
37
           Moves the wall one step
38
           The function calls the move callback received in the Ctor, and expects
39
40
           that the callback return whether the move is permitted or not.
41
42
            # Check if moving is permitted before actually moving
43
           if self._move_callback():
               self._coordinates.insert(0, self.movement_requirements()) # Adding the new head
44
                self._coordinates.pop() # Removing the last element (the tail)
45
               return True
46
           return False # If no movement made, return so
47
        def get initial position(
49
50
            self, mid_location: Coordinate, direction: Direction) -> List[Coordinate]:
51
           Returns the initial position of the wall on the play board.
52
53
            The wall occupies total of 3 cells in a board.
54
           if direction == Direction.UP:
55
               return [Coordinate(mid_location.row+row_index, mid_location.column)
                           for row_index in range(1, -2, -1)]
57
58
            elif direction == Direction.DOWN:
               return [Coordinate(mid_location.row+row_index, mid_location.column)
```

```
for row_index in range(-1, 2)]

elif direction == Direction.LEFT:

return [Coordinate(mid_location.row, mid_location.column+column_index)

for column_index in range(-1, 2)]

elif direction == Direction.RIGHT:

return [Coordinate(mid_location.row, mid_location.column+column_index)

for column_index in range(1, -2, -1)]
```