```
# Joydeep Mukherjee
# April 6th, 2020
# Senior 5
# Enloe HS
# Contest #4

def is_prime(n):
    return n in [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47]

def is_pefect_square(n):
    return n in [9, 16, 25, 36, 49]

def check_occupied_space(board, piece_in_question, a, b):
    direction = int((b-a)/abs(b-a))
    for i in range(a + (1 if direction > 0 else -1), b + (1 if direction > 0 else
-1), direction):
        if i < 52 and board[i - 1] != '-' and i != piece_in_question:
            return i + (-1 if direction > 0 else 1)
    return b

def setup(board, opp_markers, player_markers):
    for marker in opp_markers:
        board[marker - 1] = '-' if marker == 52 else 'O'
    for marker in player_markers:
        board[marker - 1] = '-' if marker == 52 else 'P'
    return board

def make_move(board, player_markers, piece_to_move, move):
    if move == 52:
        board[piece_to_move - 1] = '-'
    else:
        board[piece_to_move - 1], board[move - 1] = board[move - 1],
board[piece_to_move - 1]
    for index, marker in enumerate(player_markers):
        if(marker == piece_to_move):
            player_markers[index] = move
    return board, player_markers

def contains_horizontal_vertical_sequence(piece_to_move, move):
    valid_sequences = [
        set(range(6,9)),
        set(range(11,14)),
        set(range(16,19)),
        set(range(21,24)),
        set(range(26,29)),
```

```python
        set(range(34,37)),
        set(range(39,42)),
        set(range(44,47)),
        set(range(49,52))
    ]

    path = set(range(piece_to_move, move + 1))

    for valid_sequence in valid_sequences:
        if valid_sequence.issubset(path):
            return True
    return False

def valid_move_from_lowest_multiple(piece_to_move, move, roll):
    valid_moves = []
    for i in range(piece_to_move, move + 1):
        if i % roll == 0 and board[i - 1] == '-':
            valid_moves.append(i)
    if valid_moves:
        return max(valid_moves)
    else:
        return piece_to_move

def sum_of_markers(opp_markers, player_markers):
    opp_sum, player_sum = 0, 0
    for marker in opp_markers:
        if marker != 52:
            opp_sum += marker
    for marker in player_markers:
        if marker != 52:
            player_sum += marker
    print(str(opp_sum) + " " + str(player_sum))

def play(board, opp_markers, player_markers, rolls):
    turn = True
    for roll in rolls:
        markers_to_use = opp_markers if turn else player_markers
        piece_to_move = min(markers_to_use)
        move = piece_to_move + roll
        if move > 52:
            turn = not turn
            continue
        elif board[move - 1] == 'O' or board[move - 1] == 'P':
            turn = not turn
            continue
        elif is_prime(move):
            move = check_occupied_space(board, piece_to_move, move, move + 6)
        elif is_pefect_square(move):
            move = check_occupied_space(board, piece_to_move, move, move - 6)
        elif contains_horizontal_vertical_sequence(piece_to_move, move) and move
!= 52:
            move = valid_move_from_lowest_multiple(piece_to_move, move, roll)
        board, markers_to_use = make_move(board, markers_to_use, piece_to_move,
move)
        if turn:
            opp_markers = markers_to_use
        else:
            player_markers = markers_to_use
        turn = not turn
    sum_of_markers(opp_markers, player_markers)

with open("sr.txt") as f:
    for line in f.read().splitlines():
        board = ['-'] * 52
        inp = [int(e) for e in line.split()]
        opp_markers = inp[:3]
        player_markers = inp[3:6]
        rolls = inp[7:]
        play(setup(board, opp_markers, player_markers), opp_markers,
player_markers, rolls)
```