Backend Engineer Interview Questions

You can use any additional tools including your phone when answering the questions.

1. Use 1 line of code to calculate the sum of all odd numbers within 100
2. Remove duplicated elements from a list `lst: List[int]`
3. Reverse the string `"abc"`
4. Write a decorator `timeit` which:
   - Prints the execution time of a function
   - Accepts a parameter `max_time: int` ; when the function execution time exceeds `max_time` , prints an extra warning

```
@timeit(max_time=1)
def func(*args, **kw):
    time.sleep(2)



# Output in stdout:
# "func" function ran for 2s
# "func" function exceeds max_time
```

5. Complete the following class and make `InterInt` a generator. When first called, the generator returns `1` . After each succeeding `next()` call, the return value should increase by 1

```
class InterInt:
    def __init__(self):
        self.a = 1



# Expected results:
a = IterInt()
print(next(a))  # 1
print(next(a))  # 2
print(next(a))  # 3
```

6. Complete the following class. Emulate the key access & assignment behaviour of a dict **without** using the Python built-in dict

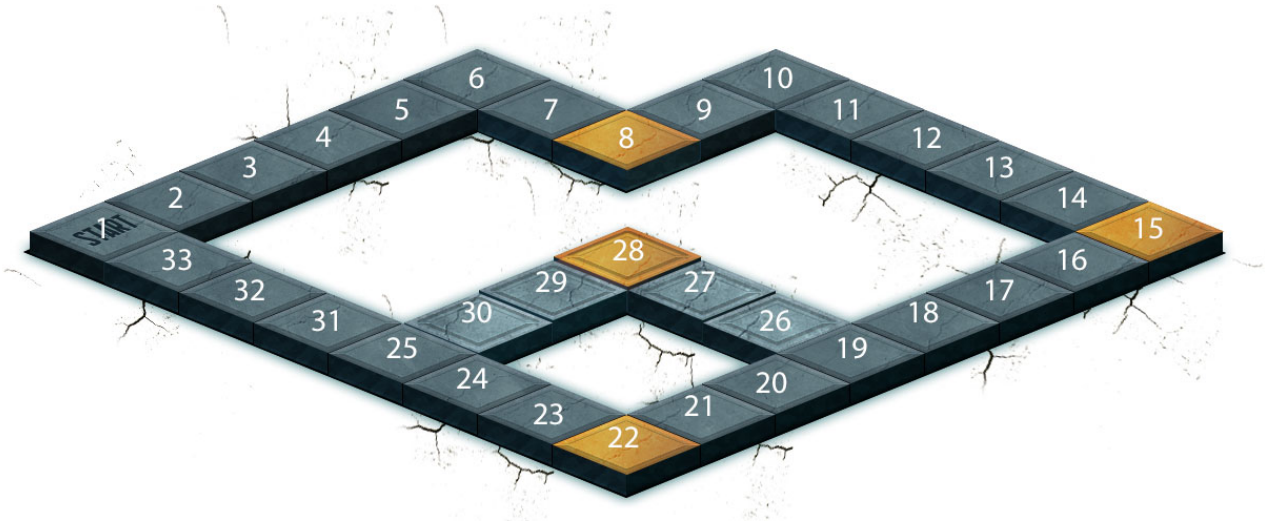
```
class CustomDict:
    pass



# Expected  results:
cdict = CustomDict()
cdict["a"] = 1
cdict["b"] = 2
```

```
print(cdict["a"])  # 1
print(cdict["c"])  # KeyError
```

7. Choose one of the following tasks

   These are real tasks from our daily work. You don't need to worry too much about the academic time/space complexity. Focus more on the engineering point of view.

   1. Find the next moving position on the map



      Implement a method

```
class Solution:
    def find_next_node(current_position: int, step: int) -> int:
        pass
```

      which returns the position the player should arrive after `step` steps.

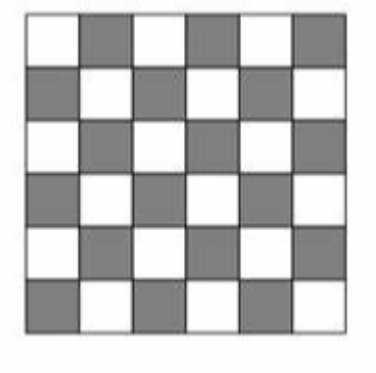
      - `step` may be negative: the player may go forward or backward
      - `abs(step)` is greater than 1 and less than 6
      - When `current_position == 19` and `step > 0`, go forward to path 26 -> 27 -> 28
      - When `current_position == 25` and `step < 0`, go backward to path 30 -> 29 -> 28
      - The map is circular: 33 goes forward to 1 and 1 goes backward to 33
      - Examples:

```
solution = Solution()
solution.find_next_node(1, 4)  # 5
solution.find_next_node(5, -4) # 1
solution.find_next_node(17, 5) # 22
solution.find_next_node(19, 2) # 27
solution.find_next_node(27, -3) # 18
solution.find_next_node(30, 3) # 32
solution.find_next_node(32, -4) # 23
solution.find_next_node(25, -2) # 29
solution.find_next_node(33, 3) # 3
solution.find_next_node(1, -2) # 32
```

      tips:

      - it is not recommended to use a mountain of `if-elif-else` statements in your code.

   2. Generate the initial game board of a Candy-Crush-like game



Implement a function

```
init_board() -> List[List[int]]
```

which returns a 6 by 6 matrix that fulfills the following requirements:

- The 36 numbers on the board must be 9 ones, 9 twos, 9 threes and 9 fours
- Any row or column must not contain 3 or more direct neighbours that are the same number
- The function return value must not be a constant
- Obviously it's not allowed to use pre-calculated answers
- Correct example answers:

```
[[3, 2, 4, 1, 3, 2],
 [2, 2, 1, 1, 4, 4],
 [4, 4, 1, 3, 3, 2],
 [4, 1, 3, 2, 2, 4],
 [3, 1, 2, 4, 3, 1],
 [3, 3, 1, 1, 2, 4]]

[[3, 3, 1, 2, 2, 4],
 [1, 1, 3, 3, 2, 4],
 [4, 4, 2, 1, 1, 3],
 [2, 2, 3, 4, 4, 1],
 [4, 4, 1, 1, 2, 2],
 [3, 1, 2, 3, 3, 4]]
```

- Wrong answers:

```
[[3, 3, 3, 2, 2, 4],
 [1, 1, 1, 3, 2, 4],
 [4, 4, 2, 1, 1, 3],
 [2, 2, 3, 4, 4, 1],
 [4, 4, 1, 1, 2, 2],
 [3, 1, 2, 3, 3, 4]]

[[3, 3, 1, 2, 2, 4],
 [1, 1, 2, 3, 2, 4],
 [4, 4, 1, 1, 2, 3],
 [2, 2, 3, 4, 4, 1],
 [4, 4, 1, 1, 2, 2],
 [3, 1, 2, 3, 3, 4]]
```