1. 10/30 개선사항

초기 명세의 모든 요구사항을 충족시키기 위해 함수에 필요한 매개변수들을 추가했습니다.

가독성 개선

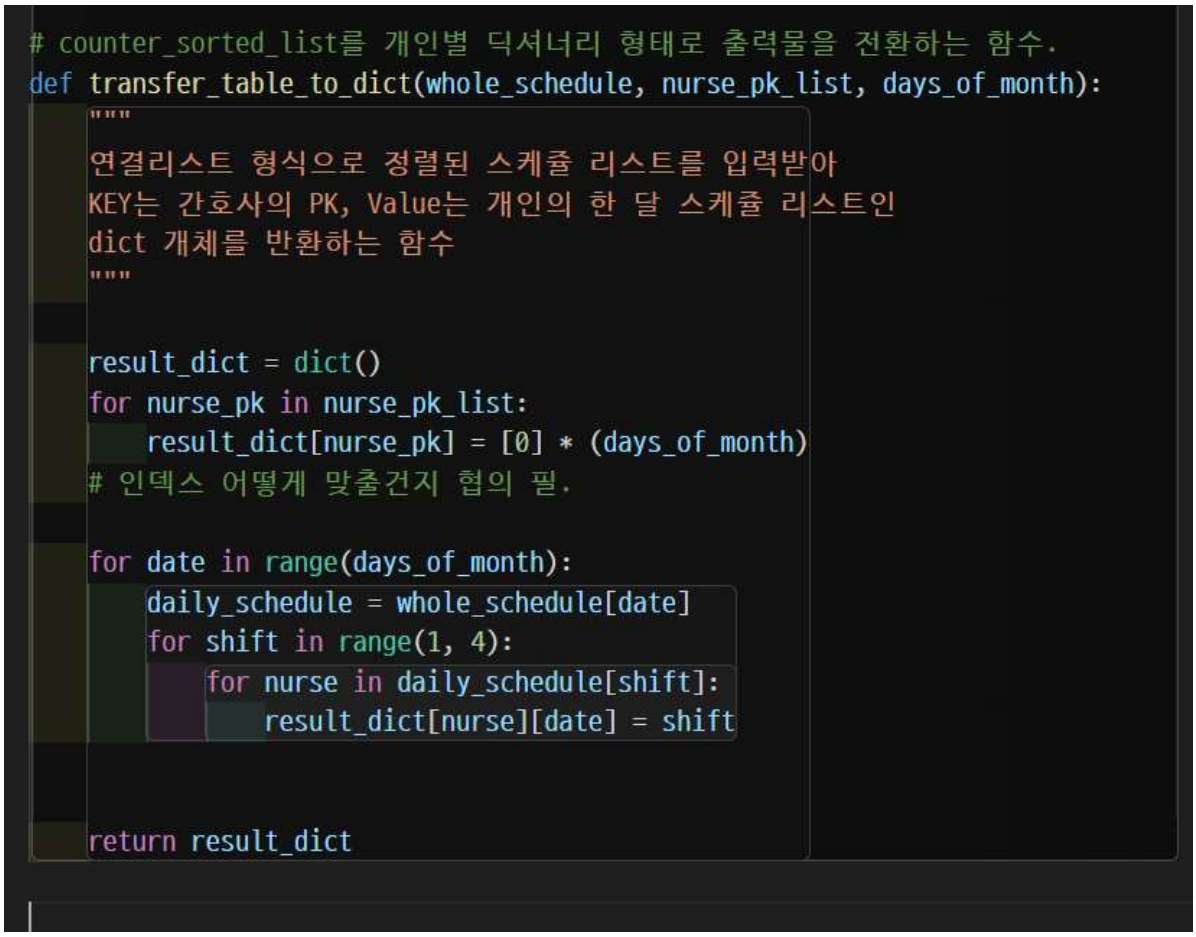
전반적인 주석 작업.

병동 전체 근무표 제작: grade 관련 제약사항

1. 함수 수정사항

1) make_monthly_schedule 출력부

`transfer_table_to_dict` 함수 생성.



```
# counter_sorted_list를 개인별 딕셔너리 형태로 출력물을 전환하는 함수.
def transfer_table_to_dict(whole_schedule, nurse_pk_list, days_of_month):
    """
    연결리스트 형식으로 정렬된 스케쥴 리스트를 입력받아
    KEY는 간호사의 PK, Value는 개인의 한 달 스케쥴 리스트인
    dict 개체를 반환하는 함수
    """

    result_dict = dict()
    for nurse_pk in nurse_pk_list:
        result_dict[nurse_pk] = [0] * (days_of_month)
    # 인덱스 어떻게 맞출건지 협의 필.

    for date in range(days_of_month):
        daily_schedule = whole_schedule[date]
        for shift in range(1, 4):
            for nurse in daily_schedule[shift]:
                result_dict[nurse][date] = shift


    return result_dict
```

출력부 변경

명세에 맞게 `whole_schdule_dict`를 반환.

```
57
58      # 출력 형식 변경.
59      whole_schedule_dict = transfer_table_to_dict(
60          whole_schedule,
61          nurse_pk_list,
62          MONTHS_LAST_DAY[current_month]
63          )
64
65      return whole_schedule_dict, nurse_info
66
```

출력 예시

```
0: [0, 12, 0, 0, 0, 0, 2, 0]]

1 [3, 3, 3, 3, 3, 0, 0, 2, 0, 0, 0, 0, 0, 1, 3, 3, 3, 0, 0, 2, 2, 0, 0, 2, 2, 2, 2, 2, 0, 0, 1]
2 [2, 2, 0, 0, 0, 3, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 3, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 3, 3]
3 [0, 0, 1, 1, 1, 0, 0, 0, 3, 3, 3, 0, 0, 0, 2, 2, 0, 0, 0, 0, 0, 0, 0, 0, 3, 3, 3, 3, 3, 0, 0, 0]
4 [1, 0, 0, 2, 2, 2, 2, 0, 0, 0, 2, 2, 2, 0, 0, 1, 0, 0, 3, 3, 3, 3, 3, 0, 0, 0, 1, 1, 2, 2, 0]
5 [0, 0, 0, 0, 0, 1, 1, 1, 2, 2, 0, 0, 0, 2, 0, 0, 2, 2, 2, 0, 0, 2, 2, 0, 0, 1, 0, 0, 1, 1, 2]
6 [0, 1, 2, 0, 0, 0, 3, 3, 0, 0, 0, 3, 3, 3, 0, 0, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 3, 0, 0]
실행 시간
0.016953706741333008
```

2) nurse_info 객체 정보 업데이트

아래 명세를 반영하기 위해 nurse_info 객체 정보 업데이트.

> ### 팀
>
> > 한 병동에 3팀, 한 팀에는 6명씩이라고 가정한다.
>
> 1. 세 팀에서 한 명씩 병동의 `DAY`, `EVENING`, `NIGHT` 근무를 맡는다.
>   - 즉, 하루에 `DAY` 3명, `EVENING` 3명, `NIGHT` 3명 씩 9명이 일을 하게 된다.

전

```
def update_nurse_info(nurse_info, temporary_schedule):
    """
    0 NURSE_NUMBER 간호사 일련번호
    1 SHIFTS, 이번 다 근무 일수
    2 SHIFT_STREAKS, 연속 근무일 수
    3 OFFS, 그.. 마크다운에 있는 'OFF' 참조.
    4 MONTHLY_NIGHT_SHIFTS, 한 달에 night 근무한 횟수
    5 VACATION_INFO,   휴가 정보(딕셔너리로 수정 예정)
    6 OFF_STREAKS,          연속 휴무
    7 LAST_SHIFT,            마지막 근무 정보
    """
```

후

```
0 NURSE_NUMBER 간호사 일련번호
1 NURSE_GRADE 간호사 grade
2 TEAM_NUMBER 팀넘버
3 SHIFTS, 이번 달 근무 일수
4 SHIFT_STREAKS, 연속 근무일 수
5 OFFS, 그.. 마크다운에 있는 'OFF' 참조.
6 MONTHLY_NIGHT_SHIFTS, 한 달에 night 근무한 횟수
7 VACATION_INFO,   휴가 정보(외부 딕셔너리로 수정 예정)
8 OFF_STREAKS,          연속 휴무
9 LAST_SHIFT,            마지막 근무 정보
"""
```

3) GRADE 섞기 기능 구현을 위한 일부 함수 업데이트

사용한 알고리즘: 비트마스킹

아래의 명세를 구현.

> 한 팀마다 `GRADE`가 적절하게 섞여야 한다.

- `GRADE`란 간호사들의 연차를 뜻한다. 뚜렷한 기준은 없으므로 기준(저연차, 고연차 등)은 직접 정한다.
- 저연차 3명이 한 팀에 들어갈 수 없다.

schedule_maker_module.py

```
    grade_counter_bit = 0
    placed_nurse_set = set()

    while priority_que:
        current_priority, nurse_number, nurse_grade, shift =
heapq.heappop(priority_que)
        current_schedule_counter[shift] += 1
        schedule_table[shift].append(nurse_number)
        placed_nurse_set.add(nurse_number)

        if nurse_grade > 0:
            grade_counter_bit |= (1 << shift)

    return schedule_table, grade_counter_bit
```

간호사가 배치되고, 그 간호사의 grade가 1 이상이라면

1 이상의 간호사가 배치되었음을 `grade_counter_bit`에 작성한다.

이후 `grade_counter_bit` 에 함께 반환.

make_schedule.py

```
        whole_team_grade_checker = 1
        whole_team_temp_schedule = []
        # 2. 스케쥴 생성
        # 팀별 스케쥴 제작 및 검증 알고리즘 필요함. 제작중.
        temporary_schedule, grade_counter_bit = make_daily_schedule(
            nurse_pk_list = nurse_pk_list,
            nurse_info = nurse_info,
            ideal_schedule = ideal_schedule,
            current_date = current_date
            )

        whole_team_grade_checker |= grade_counter_bit
```

`whole_team_grade` 비트에 체크하며 유효성 검증(구현 예정)

4) check_priority함수 (in schedule_maker_module.py) 업데이트.

세밀한 가중치 구현을 위해 아래와 같이 변경.

기존

```
    if OFF_STREAKS == 1 and not CURRENT_SHIFT:
        return random.randrange(1, 40)
```

변경

```
    if OFF_STREAKS == 1 and not CURRENT_SHIFT:
        priority_token = random.randrange(1, 40)
        return priority_token
```

2. 로직 업데이트.

스택 형식으로 이전 `nurse_info` 를 기록, 스케쥴이 오류를 일으킬 때마다 되돌아갈 정보를 불러옴. (구현 예정)

```
# 1. 선언
# 1) 최종 결과값을 저장할 리스트 .
whole_schedule = []

NUMBER_OF_NURSES = len(nurse_pk_list)
NUMBER_OF_TEAMS = len(team_list)


# 2) 예외 처리를 위한 변수들 선언
# (1) 이전 시점의 nurse_info정보를  저장하는 스택
nurse_info_stack = [[] for _ in range(NUMBER_OF_TEAMS + 1)]
# (2) 무한루프 방지를 위한 변수.
recursion_time = 0


# 2. 연산
# 1) 연산 전 준비
ideal_schedule = make_ideal_counter(needed_nurses_shift_by_team)

# 1. 종료 조건
# 정상 종료
while current_date != MONTHS_LAST_DAY[current_month] + 1:

    whole_team_grade_checker = 1
    whole_team_temp_schedule = []
    # 2. 스케쥴 생성
    # 팀별 스케쥴 제작 및 검증 알고리즘 필요함. 제작중.
    temporary_schedule, grade_counter_bit = make_daily_schedule(
        nurse_pk_list = nurse_pk_list,
        nurse_info = nurse_info,
        ideal_schedule = ideal_schedule,
        current_date = current_date
        )

    whole_team_grade_checker |= grade_counter_bit
```