```
import uuid
import random
from datetime import datetime, timedelta


def generate_job_id():
    return uuid.uuid4()

def generate_random_number(min: int, max: int):
    return random.randint(min, max)

def generate_job_name():
    job_names = ['Dig', 'Crawl', 'Jump', 'Stroll', 'Meander', 'Run', 'Sprint', 'Stride', 'Lob', 'Skate',
'Cycle', 'Fly']
    job_name_index = generate_random_number(0, len(job_names)-1)
    return job_names[job_name_index]

def generate_submitter_name():
    submitter_names = ['ron', 'dan', 'vince', 'mack', 'dom', 'ace', 'will', 'jones', 'pat', 'klint',
'mick', 'rick', 'jack']
    submiiter_name_index = generate_random_number(0, len(submitter_names)-1)
    return submitter_names[submiiter_name_index]

def generate_submission_date():
    number_of_days = generate_random_number(100, 200)
    return datetime.utcnow() + timedelta(days=number_of_days)


class Job:

    def __init__(self):
        self.number = generate_job_id()
        self.name = generate_job_name()
        self.submitter_name = generate_submitter_name()
        self.submission_date = generate_submission_date()
        self.priority = generate_random_number(1, 25)


def print_jobs(jobs):
    for job in jobs:
        print(f'Job: {job.number}  {job.submission_date}    {job.priority}      {job.submitter_name}
{job.name}')

def heapify(arr, N, i):
    largest = i
    left = 2 * i + 1
    right = 2 * i + 2

    # If left node exists and left node is greater than root, change largest to left
    if left < N and arr[largest].priority < arr[left].priority:
        largest = left

    # If right node exists and right node is greater than root, change largest to right
    if right < N and arr[largest].priority < arr[right].priority:
```

```
        largest = right

    # Change root if needed
    if largest != i:
        # Swapping the largest index of the array with i element of the array
        arr[i], arr[largest] = arr[largest], arr[i]

        # Heapify the root
        heapify(arr, N, largest)

def heap_sort(arr):
    N = len(arr)

    # Build a max heap
    for i in range(N//2 - 1, -1, -1):
        heapify(arr, N, i)

    # One by one extract elements
    for i in range(N-1, 0, -1):
        # Swap
        arr[i], arr[0] = arr[0], arr[i]

        heapify(arr, i, 0)


def __init__():
    wait_queue: list = []
    ready_queue: list = []

    # Number of jobs to be added to Wait Queue
    number_of_jobs = generate_random_number(10, 15)

    # Enter jobs into Wait Queue
    for job in range(0, number_of_jobs):
        job = Job()
        wait_queue.append(job)

    # Display the jobs in Wait Queue
    print('Printing jobs in Wait Queue \n')
    print_jobs(wait_queue)

    # Heapify the Wait Queue
    heap_sort(wait_queue)

    # Display the jobs sorted by priority in Wait Queue
    print('\n\nPrinting jobs sorted by priority in Wait Queue \n')
    print_jobs(wait_queue)

    # Move 5 jobs from Wait Queue to Ready Queue
    for i in range(0, 5, 1):
        size_of_wait_queue = len(wait_queue)

        # Extract largest priority job from Wait Queue
        largest_priority_job = wait_queue.pop(size_of_wait_queue - 1)
```

```
    # Heapify Wait Queue
    heap_sort(wait_queue)

    # Append the extracted job to Ready Queue
    ready_queue.append(largest_priority_job)

    # Heapify Ready Queue
    heap_sort(ready_queue)

print('\n\nPrinting the jobs in Wait Queue\n')
print_jobs(wait_queue)

print('\n\nPrinting the jobs in Ready Queue\n')
print_jobs(ready_queue)


# Delete 3 jobs from Ready Queue
for i in range(0, 3, 1):
    # Popping highest priority job from Ready Queue
    ready_queue.pop(len(ready_queue) - 1)

    # Heapifying the popped Ready Queue
    heap_sort(ready_queue)

    print('\n\nPrinting the jobs in popped and sorted Ready Queue\n')
    print_jobs(ready_queue)

    # Adding jobs to Wait Queue
    job = Job()
    wait_queue.append(job)

heap_sort(wait_queue)

for i in range(0, 4, 1):
    high_priority_wait_queue_job = wait_queue.pop(len(wait_queue) - 1)

    ready_queue.append(high_priority_wait_queue_job)

for i in range(0, 2, 1):
    ready_queue.pop(len(ready_queue) - 1)

heap_sort(wait_queue)
heap_sort(ready_queue)

print('\n\nSorted Wait Queue\n')
print_jobs(wait_queue)

print('\n\nSorted Ready Queue\n')
print_jobs(ready_queue)

# Change priority of 3 jobs in Wait Queue
for i in range(0, 3, 1):
    index_of_job_priority_to_be_changed = generate_random_number(0, len(wait_queue)- 1)
```

```
        wait_queue[index_of_job_priority_to_be_changed].priority = generate_random_number(1,
25)

    heap_sort(wait_queue)
    print('\n\nPrinting Wait Queue after changing priority of 3 jobs\n')
    print_jobs(wait_queue)

    wait_queue.clear()

    ready_queue.clear()


__init__()
```

```
Printing the jobs in Wait Queue

Job: e6223400-d77c-400e-bea0-4d74c221f46e  2023-01-27 04:01:39.638886    4      dom   Jump
Job: 78d188c9-c1be-42fa-86a7-ea11ac1409fc  2023-03-31 04:01:39.638903    6      jack   Fly
Job: 0a1ccef5-9216-4f97-ac47-14bda94982ba  2023-03-29 04:01:39.638926    11      will   Dig
Job: c4778259-7ac1-4c2c-a4b4-2eab6b8314b5  2023-04-16 04:01:39.638914    12      jack   Cycle
Job: 6b59d48d-bbab-4bd3-a37a-92405b8c9672  2023-03-10 04:01:39.638920    13      jones   Stroll
Job: 4e46e5da-b182-454f-8587-6f0b0a48d547  2023-03-30 04:01:39.638879    14      dom   Stride


Printing the jobs in Ready Queue

Job: cd549c7a-c19a-48d5-9e7b-62d6e5540b9a  2023-04-22 04:01:39.638931    20      jones   Stroll
Job: 124ff1b9-c75e-4ee4-9e81-a153abb376a3  2023-03-16 04:01:39.638908    20      rick   Meander
Job: 7b2ff438-4122-480a-a7d0-ae8c68bb191d  2023-03-06 04:01:39.638736    22      jones   Fly
Job: 91df7c96-ea1f-4c98-8f16-0159c528814a  2023-02-04 04:01:39.638898    23      klint   Meander
Job: a3ed454a-3ee7-4b9b-b569-4a96c07b9a1f  2023-02-25 04:01:39.638892    25      dom   Crawl


Printing the jobs in popped and sorted Ready Queue

Job: cd549c7a-c19a-48d5-9e7b-62d6e5540b9a  2023-04-22 04:01:39.638931    20      jones   Stroll
Job: 124ff1b9-c75e-4ee4-9e81-a153abb376a3  2023-03-16 04:01:39.638908    20      rick   Meander
Job: 7b2ff438-4122-480a-a7d0-ae8c68bb191d  2023-03-06 04:01:39.638736    22      jones   Fly
Job: 91df7c96-ea1f-4c98-8f16-0159c528814a  2023-02-04 04:01:39.638898    23      klint   Meander


Printing the jobs in popped and sorted Ready Queue

Job: 124ff1b9-c75e-4ee4-9e81-a153abb376a3  2023-03-16 04:01:39.638908    20      rick   Meander
Job: cd549c7a-c19a-48d5-9e7b-62d6e5540b9a  2023-04-22 04:01:39.638931    20      jones   Stroll
Job: 7b2ff438-4122-480a-a7d0-ae8c68bb191d  2023-03-06 04:01:39.638736    22      jones   Fly


Printing the jobs in popped and sorted Ready Queue

Job: cd549c7a-c19a-48d5-9e7b-62d6e5540b9a  2023-04-22 04:01:39.638931    20      jones   Stroll
Job: 124ff1b9-c75e-4ee4-9e81-a153abb376a3  2023-03-16 04:01:39.638908    20      rick   Meander


Sorted Wait Queue

Job: e6223400-d77c-400e-bea0-4d74c221f46e  2023-01-27 04:01:39.638886    4      dom   Jump
Job: 4ea2096d-f213-4254-9265-106fd57f3749  2023-04-14 04:01:39.639154    5      mick   Dig
Job: 78d188c9-c1be-42fa-86a7-ea11ac1409fc  2023-03-31 04:01:39.638903    6      jack   Fly
Job: b417a94b-c9bb-4dc7-8c0a-a8c7bcbd99f0  2023-04-14 04:01:39.639174    9      pat   Stride
Job: b3959762-393a-4f89-8b54-f4436c521da5  2023-04-20 04:01:39.639189    9      mick   Run


Sorted Ready Queue

Job: 6b59d48d-bbab-4bd3-a37a-92405b8c9672  2023-03-10 04:01:39.638920    13      jones   Stroll
Job: 4e46e5da-b182-454f-8587-6f0b0a48d547  2023-03-30 04:01:39.638879    14      dom   Stride
Job: 124ff1b9-c75e-4ee4-9e81-a153abb376a3  2023-03-16 04:01:39.638908    20      rick   Meander
Job: cd549c7a-c19a-48d5-9e7b-62d6e5540b9a  2023-04-22 04:01:39.638931    20      jones   Stroll


Printing Wait Queue after changing priority of 3 jobs

Job: 78d188c9-c1be-42fa-86a7-ea11ac1409fc  2023-03-31 04:01:39.638903    5      jack   Fly
Job: e6223400-d77c-400e-bea0-4d74c221f46e  2023-01-27 04:01:39.638886    8      dom   Jump
Job: b417a94b-c9bb-4dc7-8c0a-a8c7bcbd99f0  2023-04-14 04:01:39.639174    9      pat   Stride
Job: b3959762-393a-4f89-8b54-f4436c521da5  2023-04-20 04:01:39.639189    9      mick   Run
Job: 4ea2096d-f213-4254-9265-106fd57f3749  2023-04-14 04:01:39.639154    14      mick   Dig
(env) rohitkrishnanvidyasagar@Rohits-MacBook-Air Algo-analysis-assignments %
```