Write a program that implements the banker’s algorithm

```
import heapq

def bankers_algorithm(processes, resources):
    """
    Implements the banker's algorithm for deadlock avoidance.

    Args:
    processes: A list of processes, each represented as a dictionary with
    keys 'pid', 'max_resources', and 'allocated_resources'.
    resources: A dictionary of available resources and their maximum values.

    Returns:
    True if the system is in a safe state, False otherwise.
    """

    available_resources = {resource: resources[resource] for resource in resources}
    finish = [False] * len(processes)
    safe_sequence = []

    # Create a priority queue to track processes based on their available resources
    ready_queue = [(process['pid'], process['max_resources'], process['allocated_resources']) for process
in processes]
    heapq.heapify(ready_queue)

    while ready_queue:
    # Get the process with the least remaining resources
    _, max_resources, allocated_resources = heapq.heappop(ready_queue)
    process_id = max_resources['pid']

    # Check if the process can finish
    if all(max_resources[resource] - allocated_resources[resource] <= available_resources[resource] for
resource in resources):
    safe_sequence.append(process_id)
    finish[process_id] = True
    for resource in resources:
        available_resources[resource] += allocated_resources[resource]

    # Add any new processes that can now finish to the queue
    for i, process in enumerate(processes):
        if not finish[i] and all(process['max_resources'][resource] -
process['allocated_resources'][resource] <= available_resources[resource] for resource in resources):
        heapq.heappush(ready_queue, (process['pid'], process['max_resources'],
process['allocated_resources']))

    if all(finish):
    print("System is in a safe state. Safe sequence:", safe_sequence)
    return True
    else:
    print("Deadlock has occurred.")
    return False
```

```
# Example usage
processes = [
        {'pid': 1, 'max_resources': {'A': 3, 'B': 2, 'C': 2}, 'allocated_resources': {'A': 1, 'B': 0, 'C': 1}},
        {'pid': 2, 'max_resources': {'A': 1, 'B': 2, 'C': 1}, 'allocated_resources': {'A': 1, 'B': 1, 'C': 0}},
        {'pid': 3, 'max_resources': {'A': 1, 'B': 0, 'C': 2}, 'allocated_resources': {'A': 0, 'B': 0, 'C': 1}}
]

resources = {'A': 7, 'B': 5, 'C': 5}

bankers_algorithm(processes, resources)
```