Description

You are a software developer for a major e-commerce platform that handles millions of transactions per day. Your platform runs on a distributed system consisting of multiple servers, and you are responsible for designing a process scheduling algorithm that can handle these transactions efficiently.

• The system consists of multiple servers, each running a different service. Each service handles a specific type of transaction. For example, one service handles payment processing, another handles order processing, and so on.

• Each service consists of a pool of worker threads, which are responsible for processing incoming requests. Each worker thread has a priority level assigned to it, and a certain amount of resources assigned to it. The priority level determines the order in which the threads are scheduled to process requests. The resources assigned to each thread determine the maximum number of requests it can handle simultaneously.

• Incoming requests are queued up and assigned to the appropriate service based on the type of transaction. Once a request is assigned to a service, it is further queued up and assigned to a worker thread based on its priority level.

• Your task is to design a process scheduling algorithm that can efficiently allocate resources to worker threads to process incoming requests. The algorithm should take into account the following factors:

➢The priority level of each worker thread

➢The number of available resources assigned to each worker thread

➢The type of transaction associated with each request

• Your algorithm should prioritize worker threads with higher priority levels and assign them resources accordingly. If a worker thread with a higher priority level is not available, the algorithm should assign resources to the next available worker thread with a lower priority level.

• The algorithm should also take into account the type of transaction associated with each request. Some types of transactions may require more resources than others. For example, payment processing may require more resources than order processing.

• Your program should read the input from the standard input and write the output to the standard output. The input will contain the following information:

➢The number of services n in the system

➢The number of worker threads m for each service

➢ The priority level and resources assigned to each worker thread. Each worker thread should be on a separate line and its information should be separated by spaces in the following format: priority_level resources

➢The type of transaction associated with each request, and the number of resources

required for that transaction. Each request should be on a separate line and its information should
be separated by spaces in the following format: transaction_type resources_required • Your program should output the following information:

➢Theorder in which requests were processed

➢The average waiting time for requests

➢The average turnaround time for requests

➢The number of requests that were rejected due to lack of resources

• Additionally, your program should output the following information during periods of high traffic:

➢The number of requests that were forced to wait due to lack of available resources ➢ The number of requests that were blocked due to the absence of any available worker threads