1. Task task\_k = pInfo.tasks.get(k);

//tasks vector 에서 첫번째 task 부터 꺼낸다. == task\_k

1. double interval = task\_k.execTime;

//interval 은 꺼낸 task 의 exectime.(실행시간)

1. while (interval <= task\_k.Deadline)

//task\_k 의 시작시간이 task\_k 의 deadline 보다 작거나 같을때까지 돌린다.

1. double sum = task\_k.execTime;

//sum 에 task 의 시작시간을 더해놓는다.

1. 두번째 for(taskset 전체범위)
2. if (i < k) {  
    Task task\_i = pInfo.tasks.get(i);  
    sum += task\_i.execTime \* (Math.*ceil*(interval / task\_i.Period));  
   }

6-1

같은 taskset 두개를 놓고 taskset 에서 k 번째에 있는 task\_k 와 그전까지의 task\_i를 사용.

6-2

i < k 일때, task\_k 이전까지의 task\_i 를 하나씩 꺼내어 다음을 실행.

6-3

그리고 (task\_i의 시작시간 X 올림(task\_K의 시작시간 / task\_i의 주기)) 를 sum 에 더한다.

Sum이 interval과 같아지는 순간에 break 하고 response를 시작.

1. if (interval > task\_k.Deadline)  
    return 0;  
   else if (interval <= task\_k.Deadline) {  
    continue;  
   }

interval이 첫번째 for문의 task의 deadline을 넘어가면 실패.

Interval이 첫번째 for문의 task의 deadline과 같거나 작으면

다음 루프로 넘어가 반복.

1. Return 1;

모든 반복이 끝날때까지 task의 deadline을 넘기지 않으면 (7번 과정) 성공 1 반환