#include <stdio.h>

typedef struct { int job\_id;

int job\_size;

} Job;

#define MAX\_QUEUE\_SIZE 100

typedef struct {

Job jobs[MAX\_QUEUE\_SIZE];

int front, rear;

int size; } JobQueue;

int empty(JobQueue \*queue);

void enqueue(JobQueue \*queue, Job new\_job); Job front(JobQueue \*queue); void dequeue(JobQueue \*queue); void roundRobin(JobQueue \*queue, int time\_slice);

int empty(JobQueue \*queue) {

return queue->size == 0;

}

void enqueue(JobQueue \*queue, Job new\_job) {

if (queue->size == MAX\_QUEUE\_SIZE) { printf("Queue is full\n");

return; } if (empty(queue)) {

queue->front = 0;

}

queue->rear = (queue->rear + 1) % MAX\_QUEUE\_SIZE;

queue->jobs[queue->rear] = new\_job; queue->size++;

}

Job front(JobQueue \*queue) { if (empty(queue)) { printf("Queue is empty\n"); Job empty\_job = { -1, -1 }; return empty\_job;

}

return queue->jobs[queue->front];

}

void dequeue(JobQueue \*queue) { if (empty(queue)) {

printf("Queue is empty\n");

return; }

if (queue->front == queue->rear) {

queue->front = -1; queue->rear = -1;

} else { queue->front = (queue->front + 1) % MAX\_QUEUE\_SIZE;

}

queue->size--;

}

void roundRobin(JobQueue \*queue, int time\_slice) {

while (!empty(queue)) { Job current\_job = front(queue); dequeue(queue);

if (current\_job.job\_size > time\_slice) { printf("Processing Job %d with time slice %d\n", current\_job.job\_id, time\_slice); current\_job.job\_size -= time\_slice; enqueue(queue, current\_job);

} else { printf("Processing Job %d to completion\n", current\_job.job\_id); }

}

}

int main() {

JobQueue queue = { .front = -1, .rear = -1, .size = 0 }; // Initialize an empty queue

Job job1 = { .job\_id = 1, .job\_size = 8 };

Job job2 = { .job\_id = 2, .job\_size = 4 }; Job job3 = { .job\_id = 3, .job\_size = 6 };

enqueue(&queue, job1); enqueue(&queue, job2); enqueue(&queue, job3); roundRobin(&queue, 5);

return 0;

}

­­­­­­­­­