**ASSIGNMENT 9**

#include <stdio.h>

#include <stdbool.h>

#define NUM\_PARTITIONS 4

#define NUM\_JOBS 8

int partition\_sizes[NUM\_PARTITIONS] = {4, 8, 20, 2};

int job\_sizes[NUM\_JOBS] = {2, 14, 3, 6, 6, 10, 20, 2};

int job\_times[NUM\_JOBS] = {4, 10, 2, 1, 4, 1, 8, 6};

// Each partition: if occupied, remaining time of job; -1 = free

int partition\_status[NUM\_PARTITIONS];

int current\_time = 0;

void free\_finished\_jobs() {

for (int i = 0; i < NUM\_PARTITIONS; i++) {

if (partition\_status[i] > 0) {

partition\_status[i]--;

if (partition\_status[i] == 0) {

partition\_status[i] = -1; // job finished

}

}

}

}

int find\_best\_fit(int job\_size) {

int best\_index = -1;

int min\_waste = 100000;

for (int i = 0; i < NUM\_PARTITIONS; i++) {

if (partition\_status[i] == -1 && partition\_sizes[i] >= job\_size) {

int waste = partition\_sizes[i] - job\_size;

if (waste < min\_waste) {

min\_waste = waste;

best\_index = i;

}

}

}

return best\_index;

}

int main() {

// Initially all partitions are free

for (int i = 0; i < NUM\_PARTITIONS; i++) {

partition\_status[i] = -1;

}

int job\_index = 0;

int time\_completed\_20k = -1;

while (job\_index < NUM\_JOBS || time\_completed\_20k == -1) {

printf("Time %d:\n", current\_time);

// Free jobs that are done

free\_finished\_jobs();

// Try to allocate new jobs

while (job\_index < NUM\_JOBS) {

int best = find\_best\_fit(job\_sizes[job\_index]);

if (best == -1) break; // no fit for this job now

partition\_status[best] = job\_times[job\_index];

printf("Job %d (%dk) allocated to partition %d (%dk) for %d units\n",

job\_index + 1, job\_sizes[job\_index], best + 1, partition\_sizes[best], job\_times[job\_index]);

if (job\_index == 6) { // 20k job

time\_completed\_20k = current\_time + job\_times[job\_index];

}

job\_index++;

}

current\_time++;

}

printf("\n✅ 20KB job (Job 7) completes at time: %d\n", time\_completed\_20k);

return 0;

}