#include <stdio.h>

#include <stdlib.h>

#include <limits.h>

#define MAX\_PROCESSES 100

typedef struct {

int id;

int arrival\_time;

int burst\_time;

int remaining\_time;

int priority;

int completed;

} Process;

void sort\_by\_arrival\_time(Process \*processes, int n) {

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - 1 - i; j++) {

if (processes[j].arrival\_time > processes[j + 1].arrival\_time) {

Process temp = processes[j];

processes[j] = processes[j + 1];

processes[j + 1] = temp;

}

}

}

}

int find\_highest\_priority\_process(Process \*processes, int n, int current\_time) {

int highest\_priority\_index = -1;

int highest\_priority = INT\_MAX;

for (int i = 0; i < n; i++) {

if (processes[i].arrival\_time <= current\_time && !processes[i].completed && processes[i].priority < highest\_priority) {

highest\_priority = processes[i].priority;

highest\_priority\_index = i;

}

}

return highest\_priority\_index;

}

void preemptive\_priority\_scheduling(Process \*processes, int n) {

int completed\_processes = 0;

int current\_time = 0;

while (completed\_processes < n) {

int current\_process\_index = find\_highest\_priority\_process(processes, n, current\_time);

if (current\_process\_index == -1) {

current\_time++;

continue;

}

Process \*current\_process = &processes[current\_process\_index];

current\_process->remaining\_time--;

current\_time++;

if (current\_process->remaining\_time == 0) {

current\_process->completed = 1;

completed\_processes++;

printf("Process %d completed at time %d\n", current\_process->id, current\_time);

}

}

}

int main() {

int n;

Process processes[MAX\_PROCESSES];

printf("Enter the number of processes: ");

scanf("%d", &n);

for (int i = 0; i < n; i++) {

printf("Enter arrival time, burst time, and priority for process %d: ", i + 1);

scanf("%d %d %d", &processes[i].arrival\_time, &processes[i].burst\_time, &processes[i].priority);

processes[i].id = i + 1;

processes[i].remaining\_time = processes[i].burst\_time;

processes[i].completed = 0;

}

sort\_by\_arrival\_time(processes, n);

preemptive\_priority\_scheduling(processes, n);

return 0;

}

