#include <stdio.h>

#include <stdlib.h>

#define MEMORY\_SIZE 1000 // Total memory size

#define MAX\_PROCESSES 100 // Maximum number of processes

typedef struct {

int id;

int size;

int start\_address;

} Process;

void first\_fit(int memory[], int memory\_size, Process processes[], int num\_processes) {

for (int i = 0; i < num\_processes; i++) {

int process\_size = processes[i].size;

int allocated = 0;

for (int j = 0; j < memory\_size; j++) {

int fit = 1;

for (int k = 0; k < process\_size; k++) {

if (j + k >= memory\_size || memory[j + k] != 0) {

fit = 0;

break;

}

}

if (fit) {

processes[i].start\_address = j;

for (int k = 0; k < process\_size; k++) {

memory[j + k] = processes[i].id;

}

allocated = 1;

break;

}

}

if (!allocated) {

printf("Process %d of size %d cannot be allocated\n", processes[i].id, processes[i].size);

processes[i].start\_address = -1;

}

}

}

void print\_memory(int memory[], int memory\_size) {

printf("Memory: ");

for (int i = 0; i < memory\_size; i++) {

printf("%d ", memory[i]);

}

printf("\n");

}

void print\_processes(Process processes[], int num\_processes) {

printf("Processes:\n");

for (int i = 0; i < num\_processes; i++) {

printf("Process %d, Size: %d, Start Address: %d\n", processes[i].id, processes[i].size, processes[i].start\_address);

}

}

int main() {

int memory[MEMORY\_SIZE] = {0}; // Initialize memory to 0 (free)

Process processes[MAX\_PROCESSES];

int num\_processes;

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

scanf("%d", &num\_processes);

for (int i = 0; i < num\_processes; i++) {

processes[i].id = i + 1;

printf("Enter the size of process %d: ", processes[i].id);

scanf("%d", &processes[i].size);

processes[i].start\_address = -1; // Initially, no process is allocated

}

first\_fit(memory, MEMORY\_SIZE, processes, num\_processes);

print\_memory(memory, MEMORY\_SIZE);

print\_processes(processes, num\_processes);

return 0;

}

