#include <stdio.h>

#define MAX\_MEMORY 1000

int memory[MAX\_MEMORY];

// Function to initialize memory

void initializeMemory() {

for (int i = 0; i < MAX\_MEMORY; i++) {

memory[i] = -1; // -1 indicates that the memory is unallocated

}

}

// Function to display memory status

void displayMemory() {

int i, j;

int count = 0;

printf("Memory Status:\n");

for (i = 0; i < MAX\_MEMORY; i++) {

if (memory[i] == -1) {

count++;

j = i;

while (memory[j] == -1 && j < MAX\_MEMORY) {

j++;

}

printf("Free memory block %d-%d\n", i, j - 1);

i = j - 1;

}

}

if (count == 0) {

printf("No free memory available.\n");

}

}

// Function to allocate memory using best-fit algorithm

void allocateMemory(int processId, int size) {

int start = -1;

int blockSize = MAX\_MEMORY;

int bestStart = -1;

int bestSize = MAX\_MEMORY;

for (int i = 0; i < MAX\_MEMORY; i++) {

if (memory[i] == -1) {

if (blockSize == MAX\_MEMORY) {

start = i;

}

blockSize++;

} else {

if (blockSize >= size && blockSize < bestSize) {

bestSize = blockSize;

bestStart = start;

}

blockSize = 0;

}

}

if (bestSize >= size) {

for (int i = bestStart; i < bestStart + size; i++) {

memory[i] = processId;

}

printf("Allocated memory block %d-%d to Process %d\n", bestStart, bestStart + size - 1, processId);

} else {

printf("Memory allocation for Process %d failed (not enough contiguous memory).\n", processId);

}

}

// Function to deallocate memory

void deallocateMemory(int processId) {

for (int i = 0; i < MAX\_MEMORY; i++) {

if (memory[i] == processId) {

memory[i] = -1;

}

}

printf("Memory released by Process %d\n", processId);

}

int main() {

initializeMemory();

displayMemory();

allocateMemory(1, 200);

displayMemory();

allocateMemory(2, 300);

displayMemory();

deallocateMemory(1);

displayMemory();

allocateMemory(3, 400);

displayMemory();

return 0;

}