#include <stdio.h>

#include <stdlib.h>

#define MEMORY\_SIZE 100 // Define the size of memory

#define NUM\_BLOCKS 5 // Define the number of memory blocks

// Function prototypes

void firstFit(int blocks[], int numBlocks, int processSize[], int numProcesses);

void bestFit(int blocks[], int numBlocks, int processSize[], int numProcesses);

void worstFit(int blocks[], int numBlocks, int processSize[], int numProcesses);

void printAllocation(int allocation[], int numProcesses);

int main() {

// Memory blocks of different sizes

int blocks[NUM\_BLOCKS] = {10, 20, 30, 40, 50};

// Processes of different sizes

int processSize[] = {15, 10, 25, 30, 35};

int numProcesses = sizeof(processSize) / sizeof(processSize[0]);

printf("First-Fit Allocation:\n");

firstFit(blocks, NUM\_BLOCKS, processSize, numProcesses);

// Reset memory blocks

int blocks2[NUM\_BLOCKS] = {10, 20, 30, 40, 50};

printf("\nBest-Fit Allocation:\n");

bestFit(blocks2, NUM\_BLOCKS, processSize, numProcesses);

// Reset memory blocks

int blocks3[NUM\_BLOCKS] = {10, 20, 30, 40, 50};

printf("\nWorst-Fit Allocation:\n");

worstFit(blocks3, NUM\_BLOCKS, processSize, numProcesses);

return 0;

}

void firstFit(int blocks[], int numBlocks, int processSize[], int numProcesses) {

int allocation[numProcesses];

// Initialize allocation array with -1

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

allocation[i] = -1;

}

// Allocate memory to processes

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

for (int j = 0; j < numBlocks; j++) {

if (blocks[j] >= processSize[i]) {

allocation[i] = j;

blocks[j] -= processSize[i];

break;

}

}

}

printAllocation(allocation, numProcesses);

}

void bestFit(int blocks[], int numBlocks, int processSize[], int numProcesses) {

int allocation[numProcesses];

// Initialize allocation array with -1

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

allocation[i] = -1;

}

// Allocate memory to processes

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

int bestIdx = -1;

for (int j = 0; j < numBlocks; j++) {

if (blocks[j] >= processSize[i]) {

if (bestIdx == -1 || blocks[j] < blocks[bestIdx]) {

bestIdx = j;

}

}

}

if (bestIdx != -1) {

allocation[i] = bestIdx;

blocks[bestIdx] -= processSize[i];

}

}

printAllocation(allocation, numProcesses);

}

void worstFit(int blocks[], int numBlocks, int processSize[], int numProcesses) {

int allocation[numProcesses];

// Initialize allocation array with -1

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

allocation[i] = -1;

}

// Allocate memory to processes

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

int worstIdx = -1;

for (int j = 0; j < numBlocks; j++) {

if (blocks[j] >= processSize[i]) {

if (worstIdx == -1 || blocks[j] > blocks[worstIdx]) {

worstIdx = j;

}

}

}

if (worstIdx != -1) {

allocation[i] = worstIdx;

blocks[worstIdx] -= processSize[i];

}

}

printAllocation(allocation, numProcesses);

}

void printAllocation(int allocation[], int numProcesses) {

printf("Process No.\tProcess Size\tBlock No.\n");

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

printf("%d\t\t%d\t\t", i + 1, allocation[i] != -1 ? allocation[i] : -1);

if (allocation[i] != -1) {

printf("%d\n", allocation[i] + 1);

} else {

printf("Not Allocated\n");

}

}

}

