Memory management algorithm

#include <stdio.h>

#define MAX 25

// First Fit Allocation

void firstFit(int blockSize[], int m, int processSize[], int n) {

int allocation[n];

for (int i = 0; i < n; i++) allocation[i] = -1;

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

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

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

allocation[i] = j;

blockSize[j] -= processSize[i];

break;

}

}

}

printf("\nFirst Fit Allocation:\n");

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

if (allocation[i] != -1)

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

else

printf("Process %d -> Not Allocated\n", i + 1);

}

}

// Best Fit Allocation

void bestFit(int blockSize[], int m, int processSize[], int n) {

int allocation[n];

for (int i = 0; i < n; i++) allocation[i] = -1;

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

int bestIdx = -1;

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

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

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

bestIdx = j;

}

}

}

if (bestIdx != -1) {

allocation[i] = bestIdx;

blockSize[bestIdx] -= processSize[i];

}

}

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

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

if (allocation[i] != -1)

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

else

printf("Process %d -> Not Allocated\n", i + 1);

}

}

// Worst Fit Allocation

void worstFit(int blockSize[], int m, int processSize[], int n) {

int allocation[n];

for (int i = 0; i < n; i++) allocation[i] = -1;

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

int worstIdx = -1;

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

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

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

worstIdx = j;

}

}

}

if (worstIdx != -1) {

allocation[i] = worstIdx;

blockSize[worstIdx] -= processSize[i];

}

}

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

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

if (allocation[i] != -1)

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

else

printf("Process %d -> Not Allocated\n", i + 1);

}

}

// Main Function

int main() {

int blockSize[] = {100, 500, 200, 300, 600};

int processSize[] = {212, 417, 112, 426};

int m = sizeof(blockSize) / sizeof(blockSize[0]);

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

// Copies of block sizes for each strategy

int blockFirstFit[m], blockBestFit[m], blockWorstFit[m];

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

blockFirstFit[i] = blockSize[i];

blockBestFit[i] = blockSize[i];

blockWorstFit[i] = blockSize[i];

}

firstFit(blockFirstFit, m, processSize, n);

bestFit(blockBestFit, m, processSize, n);

worstFit(blockWorstFit, m, processSize, n);

return 0;

}

