Best fit, worst fit and first fit

#include <stdio.h>

#define MAX 25

// Function to reset the block sizes before each algorithm

void resetBlocks(int dest[], int src[], int size) {

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

dest[i] = src[i];

}

}

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

int allocation[n];

int used[m]; // Array to track if a block is already used

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

allocation[i] = -1;

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

used[i] = 0;

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

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

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

allocation[i] = j;

used[j] = 1; // Mark this block as used

break;

}

}

}

printf("\nFirst Fit Allocation (One Process per Block):\n");

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

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

printf("%d\t\t%d\t\t", i + 1, processSize[i]);

if (allocation[i] != -1)

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

else

printf("Not Allocated\n");

}

}

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");

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

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

printf("%d\t\t%d\t\t", i + 1, processSize[i]);

if (allocation[i] != -1)

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

else

printf("Not Allocated\n");

}

}

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

int allocation[n];

int used[m]; // Array to track if a block is already used

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

allocation[i] = -1;

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

used[i] = 0;

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

int worstIdx = -1;

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

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

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

worstIdx = j;

}

}

if (worstIdx != -1) {

allocation[i] = worstIdx;

used[worstIdx] = 1; // Mark this block as used

}

}

printf("\nWorst Fit Allocation (One Process per Block):\n");

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

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

printf("%d\t\t%d\t\t", i + 1, processSize[i]);

if (allocation[i] != -1)

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

else

printf("Not Allocated\n");

}

}

int main() {

int m, n, blockSize[MAX], processSize[MAX];

int blockCopy[MAX];

printf("Enter number of memory blocks: ");

scanf("%d", &m);

printf("Enter size of each block:\n");

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

scanf("%d", &blockSize[i]);

printf("Enter number of processes: ");

scanf("%d", &n);

printf("Enter size of each process:\n");

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

scanf("%d", &processSize[i]);

// First-Fit

resetBlocks(blockCopy, blockSize, m);

firstFit(blockCopy, m, processSize, n);

// Best-Fit

resetBlocks(blockCopy, blockSize, m);

bestFit(blockCopy, m, processSize, n);

// Worst-Fit

resetBlocks(blockCopy, blockSize, m);

worstFit(blockCopy, m, processSize, n);

return 0;

}

Output:

