Construct a C program for implementation the various memory allocation strategies

#include <stdio.h>

#define MAX 20

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

int allocation[MAX];

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++) {

printf("Process %d (%d KB) -> ", i + 1, processSize[i]);

if (allocation[i] != -1)

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

else

printf("Not Allocated\n");

}

}

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

int allocation[MAX];

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++) {

printf("Process %d (%d KB) -> ", i + 1, processSize[i]);

if (allocation[i] != -1)

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

else

printf("Not Allocated\n");

}

}

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

int allocation[MAX];

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++) {

printf("Process %d (%d KB) -> ", i + 1, processSize[i]);

if (allocation[i] != -1)

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

else

printf("Not Allocated\n");

}

}

int main() {

int m, n;

int blockSize[MAX], processSize[MAX];

int b1[MAX], b2[MAX], b3[MAX];

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

scanf("%d", &m);

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

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

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

b1[i] = b2[i] = b3[i] = 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]);

firstFit(b1, m, processSize, n);

bestFit(b2, m, processSize, n);

worstFit(b3, m, processSize, n);

return 0;

}