#include <bits/stdc++.h>

using namespace std;

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

{

int allocation[n];

memset(allocation, -1, sizeof(allocation));

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

{

// Find the best fit block for current process

int bestIdx = -1;

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

{

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

{

if (bestIdx == -1)

bestIdx = j;

else if (blockSize[bestIdx] > blockSize[j])

bestIdx = j;

}

}

// If we could find a block for current process

if (bestIdx != -1)

{

// allocate block j to p[i] process

allocation[i] = bestIdx;

// Reduce available memory in this block.

blockSize[bestIdx] -= processSize[i];

}

}

cout << "No.\tSize\tBlock no.\n";

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

{

cout << i + 1 << "\t" << processSize[i] << "\t";

if (allocation[i] != -1)

cout << allocation[i] + 1;

else

cout << "Not Allocated";

cout << endl;

}

}

void firstFit(int blockSize[], int m,

int processSize[], int n)

{

// Stores block id of the

// block allocated to a process

int allocation[n];

// Initially no block is assigned to any process

memset(allocation, -1, sizeof(allocation));

// pick each process and find suitable blocks

// according to its size ad assign to it

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

{

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

{

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

{

// allocate block j to p[i] process

allocation[i] = j;

// Reduce available memory in this block.

blockSize[j] -= processSize[i];

break;

}

}

}

cout << "No.\tSize\tBlock no.\n";

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

{

cout << i + 1 << '\t' << processSize[i] << '\t';

if (allocation[i] != -1)

cout << allocation[i] + 1;

else

cout << "Not Allocated";

cout << endl;

}

}

void NextFit(int blockSize[], int m, int processSize[], int n)

{

// Stores block id of the block allocated to a

// process

int allocation[n], j = 0;

// Initially no block is assigned to any process

memset(allocation, -1, sizeof(allocation));

// pick each process and find suitable blocks

// according to its size ad assign to it

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

// Do not start from beginning

while (j < m) {

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

// allocate block j to p[i] process

allocation[i] = j;

// Reduce available memory in this block.

blockSize[j] -= processSize[i];

break;

}

// mod m will help in traversing the blocks from

// starting block after we reach the end.

j = (j + 1) % m;

}

}

cout << "No.\tSize\tBlock no.\n";

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

cout << i + 1 << "\t" << processSize[i] << "\t";

if (allocation[i] != -1)

cout << allocation[i] + 1;

else

cout << "Not Allocated";

cout << endl;

}

}

void worstFit(int blockSize[], int m, int processSize[],

int n)

{

// Stores block id of the block allocated to a

// process

int allocation[n];

// Initially no block is assigned to any process

memset(allocation, -1, sizeof(allocation));

// pick each process and find suitable blocks

// according to its size ad assign to it

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

{

// Find the best fit block for current process

int wstIdx = -1;

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

{

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

{

if (wstIdx == -1)

wstIdx = j;

else if (blockSize[wstIdx] < blockSize[j])

wstIdx = j;

}

}

// If we could find a block for current process

if (wstIdx != -1)

{

// allocate block j to p[i] process

allocation[i] = wstIdx;

// Reduce available memory in this block.

blockSize[wstIdx] -= processSize[i];

}

}

cout << "No.\tSize\tBlock no.\n";

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

{

cout << i+1 << "\t" << processSize[i] << "\t";

if (allocation[i] != -1)

cout << allocation[i] + 1;

else

cout << "Not Allocated";

cout << endl;

}

}

// Driver code

int main()

{

int n, m, processSize[20], blockSize[20], choice;

while (1)

{

cout << "\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*MENU\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\n";

cout << "1. Best Fit\n2. First Fit\n3. Next Fit\n4. Worst Fit\n5. Exit\n\n";

cout << "Enter your choice : ";

cin >> choice;

if(choice == 5){

exit(0);

}

cout << "Enter number of processes : ";

cin >> n;

cout << "Enter the size of the processes \n";

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

{

cout << "Process no. " << i << ":";

cin >> processSize[i];

}

cout << "Enter number of blocks : ";

cin >> m;

cout << "Enter the size of the blocks \n";

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

{

cout << "Block no. " << i << ":";

cin >> blockSize[i];

}

switch (choice)

{

case 1:

{

bestFit(blockSize, m, processSize, n);

break;

}

case 2:

{

firstFit(blockSize, m, processSize, n);

break;

}

case 3:

{

NextFit(blockSize, m, processSize, n);

break;

}

case 4:

{

worstFit(blockSize, m, processSize, n);

break;

}

default:

break;

}

}

return 0;

}

