Worst-fit

#include <iostream>

#include <cstring>

using namespace std;

// Function to allocate memory to blocks as per worst fit algorithm

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 and assign to it

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

// Find the worst fit block for the current process

int wstIdx = -1;

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

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

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

wstIdx = j;

}

}

}

// If we could find a block for the current process

if (wstIdx != -1) {

// allocate block wstIdx to process i

allocation[i] = wstIdx;

// Reduce available memory in this block.

blockSize[wstIdx] -= processSize[i];

}

}

cout << "\nProcess No.\tProcess Size\tBlock no.\n";

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

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

if (allocation[i] != -1)

cout << allocation[i] + 1;

else

cout << "Not Allocated";

cout << endl;

}

}

// Driver code

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]);

worstFit(blockSize, m, processSize, n);

return 0;

}

OUTPUT:

