import java.util.Scanner;

import java.util.ArrayList;

import java.util.List;

public class MemoryAllocation {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

// Input number of blocks and block sizes

System.out.print("Enter the number of blocks: ");

int numBlocks = scanner.nextInt();

int[] blocks = new int[numBlocks];

System.out.println("Enter the sizes of the blocks:");

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

blocks[i] = scanner.nextInt();

}

// Input number of files and file sizes

System.out.print("Enter the number of files: ");

int numFiles = scanner.nextInt();

int[] files = new int[numFiles];

System.out.println("Enter the sizes of the files:");

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

files[i] = scanner.nextInt();

}

int choice;

do{

// Input allocation strategy

System.out.println("Choose allocation strategy: ");

System.out.println("1. First Fit");

System.out.println("2. Best Fit");

System.out.println("3. Worst Fit");

System.out.println("4. Next Fit");

choice = scanner.nextInt();

switch (choice) {

case 1:

firstFit(blocks, files);

break;

case 2:

bestFit(blocks, files);

break;

case 3:

worstFit(blocks, files);

break;

case 4:

nextFit(blocks, files);

break;

default:

System.out.println("Invalid choice.");

break;

}

}while(choice!=5);

scanner.close();

}

public static void firstFit(int[] blocks, int[] files) {

System.out.println("First Fit Allocation:");

int[] allocation = new int[files.length];

for (int i = 0; i < files.length; i++) {

allocation[i] = -1; // -1 indicates no block allocated

for (int j = 0; j < blocks.length; j++) {

if (blocks[j] >= files[i]) {

allocation[i] = j;

blocks[j] -= files[i];

break;

}

}

}

printAllocation(files, allocation);

}

public static void bestFit(int[] blocks, int[] files) {

System.out.println("Best Fit Allocation:");

int[] allocation = new int[files.length];

for (int i = 0; i < files.length; i++) {

int bestIdx = -1;

for (int j = 0; j < blocks.length; j++) {

if (blocks[j] >= files[i]) {

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

bestIdx = j;

}

}

}

allocation[i] = bestIdx;

if (bestIdx != -1) {

blocks[bestIdx] -= files[i];

}

}

printAllocation(files, allocation);

}

public static void worstFit(int[] blocks, int[] files) {

System.out.println("Worst Fit Allocation:");

int[] allocation = new int[files.length];

for (int i = 0; i < files.length; i++) {

int worstIdx = -1;

for (int j = 0; j < blocks.length; j++) {

if (blocks[j] >= files[i]) {

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

worstIdx = j;

}

}

}

allocation[i] = worstIdx;

if (worstIdx != -1) {

blocks[worstIdx] -= files[i];

}

}

printAllocation(files, allocation);

}

public static void nextFit(int[] blocks, int[] files) {

System.out.println("Next Fit Allocation:");

int[] allocation = new int[files.length];

int m = blocks.length;

int j = 0; // Start from the first block

for (int i = 0; i < files.length; i++) {

allocation[i] = -1; // -1 indicates no block allocated

while (true) {

if (blocks[j] >= files[i]) {

allocation[i] = j;

blocks[j] -= files[i];

break;

}

j = (j + 1) % m; // Circularly move to the next block

if (j == 0) { // If we looped back to the beginning

System.out.println("Not enough memory to allocate file " + i);

break;

}

}

}

printAllocation(files, allocation);

}

public static void printAllocation(int[] files, int[] allocation) {

System.out.println("File Number\tFile Size\tBlock Number");

for (int i = 0; i < files.length; i++) {

System.out.println(i + "\t\t" + files[i] + "\t\t" + (allocation[i] != -1 ? allocation[i] : "Not Allocated"));

}

}

}