There are N spots for fishing in the fishing center.  
The center has 3 gates, and a number of customers standing before each gate.  
  
To avoid disorder, the customers must enter the center follow these rules :  
1. Only one Gate is open at a time, and it will be closed after all customers of that gate entered.  
2. When open a gate, the customers stand before that gate will enter one by one, and go to the closest & empty spot from their position.  
- The distance from a gate to the spot right above it is 1m  
- Whenever a customer goes further one spot (to left or right), it takes additionally 1m further.  
- Ex : The distance from Gate 1 to spot 4 is 1m, and to spot 3, 5 is 2m  
3. If there are 2 two spots that closest to a customer, he can choose any one of them (you should consider this case)  
4. After all customers enter the gate, choose the next gate to open and proceed same as above  
  
You should find a way so that the sum of the moving distance of all customers is minimum and print out that sum.  
Ex) In above figure :  
- The number of fishing spots : 10  
- Gate 1 : location is 4, number of waiting customer is 5  
- Gate 2 : location is 6, number of waiting customer is 2  
- Gate 3 : location is 10, number of waiting customer is 2  
  
Case 1) We open the gate by the order : Gate 1 > Gate 2 > Gate 3  
  
  
  
For this case, the sum of moving distance is : 3+2+1+2+3+2+3+2+1 = 19  
  
Case 2) We open the gate by the order : Gate 2 > Gate 1 > Gate 3  
When open Gate 3, the 1st customer will go to spot 6, the second one can go to spot 5 or 7  
OR   
Case 2-1)  
  
In this case, the sum is : 4+3+2+1+2+1+4+2+1 = 20  
  
Case 2-2)  
  
In this case, the sum is : 4+3+2+1+2+1+2+2+1 = 18  
  
[Input]  
- The first line given the number of test case T (T <= 50)  
- For each test case:  
+ The first line given the number of spots N (10 <= N <= 60)  
+ The next three lines give the information of 3 gates :  
> Gate's position P ( 1 <= P <= N)  
> The number of customers standing before that gate C ( 1 <= C <= 20 )  
  
[Output]  
The minimum moving distance of all customers  
Case #1  
18  
Case #2  
25  
Case #3  
57  
Case #4  
86  
Case #5  
339

package APS2;  
  
import java.util.Scanner;  
  
class Fishing {  
static int N, min, sum;  
static int arr[], gate[][], vs[];  
  
static void Backtrack(int cnt) {  
if (cnt == 3) {  
sum = 0;  
for (int j = 0; j < 3; j++)  
for (int i = 0; i < N; i++) {  
if (arr[i] == j + 1) {  
if (i >= gate[j][0])  
sum += i - gate[j][0] + 1;  
else  
sum += gate[j][0] - i + 1;  
}  
}  
if (min > sum)  
min = sum;  
return;  
}  
for (int i = 0; i < 3; i++) {  
if (vs[i] == 0) {  
vs[i] = 1;  
Sort(gate[i][0], gate[i][1], i + 1);  
Backtrack(cnt + 1);  
for (int j = 0; j < N; j++) {  
if (arr[j] == i + 1)  
arr[j] = 0;  
}  
vs[i] = 0;  
}  
}  
}  
  
static void Sort(int pos, int s, int d) {  
int i = 0;  
while (s > 0) {  
int x = pos - i;  
int y = pos + i;  
if (arr[pos] == 0) {  
arr[pos] = d;  
s--;  
}  
if (x >= 0 && s > 0 && arr[x] == 0) {  
arr[x] = d;  
s--;  
}  
if (y < N && s > 0 && arr[y] == 0) {  
arr[y] = d;  
s--;  
}  
i++;  
}  
}  
  
static void Backtrack2(int cnt) {  
if (cnt == 3) {  
sum = 0;  
for (int j = 0; j < 3; j++)  
for (int i = 0; i < N; i++) {  
if (arr[i] == j + 1) {  
if (i >= gate[j][0])  
sum += i - gate[j][0] + 1;  
else  
sum += gate[j][0] - i + 1;  
}  
}  
if (min > sum)  
min = sum;  
return;  
}  
for (int i = 0; i < 3; i++) {  
if (vs[i] == 0) {  
vs[i] = 1;  
Sort2(gate[i][0], gate[i][1], i + 1);  
Backtrack2(cnt + 1);  
for (int j = 0; j < N; j++) {  
if (arr[j] == i + 1)  
arr[j] = 0;  
}  
vs[i] = 0;  
}  
}  
}  
  
static void Sort2(int pos, int s, int d) {  
int i = 0;  
while (s > 0) {  
int x = pos - i;  
int y = pos + i;  
if (arr[pos] == 0) {  
arr[pos] = d;  
s--;  
}  
if (y < N && s > 0 && arr[y] == 0) {  
arr[y] = d;  
s--;  
}  
if (x >= 0 && s > 0 && arr[x] == 0) {  
arr[x] = d;  
s--;  
}  
i++;  
}  
}  
  
public static void main(String args[]) throws Exception {  
Scanner sc = new Scanner([System.in](http://system.in/));  
int T;  
int Answer;  
T = sc.nextInt();  
/\*  
\* Read each test case from standard input.  
\*/  
  
for (int test\_case = 1; test\_case <= T; test\_case++) {  
Answer = 0;  
N = sc.nextInt();  
arr = new int[N];  
gate = new int[3][2];  
vs = new int[3];  
for (int i = 0; i < 3; i++) {  
gate[i][0] = sc.nextInt() - 1;  
gate[i][1] = sc.nextInt();  
}  
min = 1000;  
Backtrack(0);  
Backtrack2(0);  
// Print the answer to standard output(screen).  
Answer = min;  
System.out.println("#" + test\_case + " " + Answer);  
}  
}  
}

package Fishing;  
  
import java.io.FileInputStream;  
import java.util.Scanner;  
  
public class Fishing {  
static int T;  
static int N;  
static int[] Cong;  
static int[] sokhach;  
  
static int[] visit;  
static int[] kc;  
static int min;  
  
static void BT(int n, int sum){  
if(n == 3){  
if(sum < min ) min = sum;  
} else {  
if (sum > min ) return;  
else {  
  
}  
}  
}  
  
public static void main(String args[]) throws Exception{  
System.setIn(new FileInputStream("src/Fishing/input"));  
Scanner sc = new Scanner([System.in](http://system.in/));  
  
T = sc.nextInt();  
  
for(int tc = 1; tc <= T; tc++){  
N = sc.nextInt();  
visit = new int[N];  
kc = new int[N];  
Cong = new int[3];  
sokhach = new int[3];  
  
for(int i = 0; i < 3; i++){  
Cong[i] = sc.nextInt();  
sokhach[i] = sc.nextInt();  
}  
for(int i = 0; i < 3; i++)   
System.out.println(Cong[i] + " "+ sokhach[i]);  
}  
}  
}