Alineaciones

Autora del problema: Blanca Huergo Muñoz (Miembro del comité organizador)

Para resolver este problema, lo mejor era partir a los futbolistas en dos categorías: porteros y no porteros. De los porteros, teníamos que coger al que tuviera la máxima puntuación.

Nos quedaba entonces escoger a diez del resto. Había que escoger a los diez con mejor puntuación y, en caso de empate, desempatar por cercanía al portero. Hay varias formas de hacer esto. El código que mostramos ordena a los futbolistas por puntuación dentro de su categoría y va cogiendo uno a uno hasta rellenar la alineación, en cada ronda cogiendo al mejor de los que quedan y, en caso de empate, escogiendo defensas antes que centrocampistas y a su vez antes que delanteros. Otra opción distinta era guardarlos todos en un mismo vector y hacer una función de ordenación que te los ordenara según su puntuación y, en caso de empate, por su categoría.

Solución en C++

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

void encontrarAlineacion(vector<int> & porteros, vector<int> & defensas, vector<int> &
centrocampistas, vector<int> & delanteros) {
	sort(porteros.begin(), porteros.end(), greater<int>());
	int puntuacion = porteros[0];

	sort(defensas.begin(), defensas.end(), greater<int>());
	sort(centrocampistas.begin(), centrocampistas.end(), greater<int>());
	sort(delanteros.begin(), delanteros.end(), greater<int>());

	int cogidosDef = 0, cogidosCen = 0, cogidosDel = 0;
	while(cogidosDef + cogidosCen + cogidosDel < 10) {
	int siguienteDef = -1e9;
	if (cogidosDef < (int) defensas.size())
		siguienteDef = defensas[cogidosDef];

	int siguienteCen = -1e9;
	if (cogidosCen < (int) centrocampistas.size())
		siguienteCen = centrocampistas[cogidosCen];

	int siguienteDel = -1e9;
	if (cogidosDel < (int) delanteros.size())
		siguienteDel = delanteros[cogidosDel];

	if (siguienteDef >= max(siguienteCen, siguienteDel)) {
		puntuacion += siguienteDef;
		cogidosDef++;
	} else if (siguienteCen >= max(siguienteDef, siguienteDel)) {
```

```
                puntuacion += siguienteCen;
                cogidosCen++;
        } else {
                puntuacion += siguienteDel;
                cogidosDel++;
        }
        }
        cout << puntuacion << ' ' << cogidosDef << '-' << cogidosCen << '-' << cogidosDel
<< '\n';
}

int main() {
        int t, numPorteros, numDefensas, numCentrocampistas, numDelanteros;
        cin >> t;
        while(t--) {
        cin >> numPorteros;
        vector<int> porteros(numPorteros);
        for (int i = 0; i < numPorteros; i++)
                cin >> porteros[i];

        cin >> numDefensas;
        vector<int> defensas(numDefensas);
        for (int i = 0; i < numDefensas; i++)
                cin >> defensas[i];

        cin >> numCentrocampistas;
        vector<int> centrocampistas(numCentrocampistas);
        for (int i = 0; i < numCentrocampistas; i++)
                cin >> centrocampistas[i];

        cin >> numDelanteros;
        vector<int> delanteros(numDelanteros);
        for (int i = 0; i < numDelanteros; i++)
                cin >> delanteros[i];

        encontrarAlineacion(porteros, defensas, centrocampistas, delanteros);
        }
        return 0;
}
```

Autora de la solución: Blanca Huergo Muñoz (Miembro del comité organizador)