Tarea 2

#include <iostream>

#include <algorithm>

using namespace std;

bool verificarMateriales(int tablasLargas, int tablasCortas, int ganchosPequenos, int ganchosGrandes, int tornillos, int& estantesExtras, int& tablasFaltantes, int& cortasFaltantes, int& ganchosPequenosFaltantes, int& ganchosGrandesFaltantes, int& tornillosFaltantes);

void mostrarResultado(bool posible, int tablasFaltantes, int cortasFaltantes, int ganchosPequenosFaltantes, int ganchosGrandesFaltantes, int tornillosFaltantes, int estantesExtras);

int main() {

int tablasLargas = 26;

int tablasCortas = 33;

int ganchosPequenos = 200;

int ganchosGrandes = 20;

int tornillos = 510;

int estantesExtras = 0;

int tablasFaltantes = 0, cortasFaltantes = 0, ganchosPequenosFaltantes = 0, ganchosGrandesFaltantes = 0, tornillosFaltantes = 0;

bool posible = verificarMateriales(tablasLargas, tablasCortas, ganchosPequenos, ganchosGrandes, tornillos, estantesExtras, tablasFaltantes, cortasFaltantes, ganchosPequenosFaltantes, ganchosGrandesFaltantes, tornillosFaltantes);

mostrarResultado(posible, tablasFaltantes, cortasFaltantes, ganchosPequenosFaltantes, ganchosGrandesFaltantes, tornillosFaltantes, estantesExtras);

return 0;

}

bool verificarMateriales(int tablasLargas, int tablasCortas, int ganchosPequenos, int ganchosGrandes, int tornillos, int& estantesExtras, int& tablasFaltantes, int& cortasFaltantes, int& ganchosPequenosFaltantes, int& ganchosGrandesFaltantes, int& tornillosFaltantes) {

int requeridasTablasLargas = 4 \* 15;

int requeridasTablasCortas = 6 \* 15;

int requeridasGanchosPequenos = 12 \* 15;

int requeridasGanchosGrandes = 2 \* 15;

int requeridasTornillos = 14 \* 15;

if (tablasLargas >= requeridasTablasLargas && tablasCortas >= requeridasTablasCortas &&

ganchosPequenos >= requeridasGanchosPequenos && ganchosGrandes >= requeridasGanchosGrandes &&

tornillos >= requeridasTornillos) {

int sobrantesTablasLargas = tablasLargas - requeridasTablasLargas;

int sobrantesTablasCortas = tablasCortas - requeridasTablasCortas;

int sobrantesGanchosPequenos = ganchosPequenos - requeridasGanchosPequenos;

int sobrantesGanchosGrandes = ganchosGrandes - requeridasGanchosGrandes;

int sobrantesTornillos = tornillos - requeridasTornillos;

estantesExtras = min({sobrantesTablasLargas / 4, sobrantesTablasCortas / 6, sobrantesGanchosPequenos / 12, sobrantesGanchosGrandes / 2, sobrantesTornillos / 14});

return true;

} else {

tablasFaltantes = max(0, requeridasTablasLargas - tablasLargas);

cortasFaltantes = max(0, requeridasTablasCortas - tablasCortas);

ganchosPequenosFaltantes = max(0, requeridasGanchosPequenos - ganchosPequenos);

ganchosGrandesFaltantes = max(0, requeridasGanchosGrandes - ganchosGrandes);

tornillosFaltantes = max(0, requeridasTornillos - tornillos);

return false;

}

}

void mostrarResultado(bool posible, int tablasFaltantes, int cortasFaltantes, int ganchosPequenosFaltantes, int ganchosGrandesFaltantes, int tornillosFaltantes, int estantesExtras) {

if (posible) {

cout << "Es posible realizar los 15 estantes pedidos" << endl;

cout << "Con el material sobrante puedes construir " << estantesExtras << " estantes adicionales." << endl;

} else {

cout << "No es posible cubrir el pedido porque faltan: " << endl;

cout << tablasFaltantes << " tablas largas" << endl;

cout << cortasFaltantes << " tablas cortas" << endl;

cout << ganchosPequenosFaltantes << " ganchos pequeños" << endl;

cout << ganchosGrandesFaltantes << " ganchos grandes" << endl;

cout << tornillosFaltantes << " tornillos" << endl;

}

}