**UNIVERSIDAD PERUANA DE CIENCIAS APLICADAS**

**FACULTAD DE INGENIERÍA**

**CARRERA CIENCIAS DE LA COMPUTACIÓN**



**TRABAJO PARCIAL**

**CURSO**: Fundamentos de programación competitiva

**Sección:** CC51

**Docente**: Peter Jonathan Montalvo Garcia

|  |  |
| --- | --- |
| **Código** | **Apellidos y Nombres** |
| U202110902 | Poma Astete, Luis Felipe |
| U202115065 | Sovero Cubillas, John Davids |

Mayo 2023

**Ejercicio 1:**

Dado un conjunto de números enteros, se desea encontrar todos los pares de números que sumen un valor específico. Es decir, dado un valor k, se deben encontrar todos los pares de números en el conjunto que sumen k.

#include <iostream>

#include <set>

using *namespace* std;

*int* main()

{

// Crear el conjunto de números

set<*int*> numeros = {3, 5, 8, 1, 4, 6, 2, 7, 9};

// Definir el valor k

*int* k = 10;

// Crear el conjunto de pares que suman k

set<pair<*int*, *int*>> pares;

// Recorrer el conjunto de números

for (*auto* it = numeros.begin(); it != numeros.end(); ++it)

{

// Buscar el número complementario en el conjunto

*int* complementario = k - \*it;

*auto* it\_complementario = numeros.find(complementario);

if (it\_complementario != numeros.end())

{

// Se encontró el número complementario

if (\*it < complementario)

{

// Agregar el par al conjunto de pares

pares.insert(make\_pair(\*it, complementario));

}

else

{

// Agregar el par al conjunto de pares

pares.insert(make\_pair(complementario, \*it));

}

}

}

// Mostrar los pares que suman k

cout << "Los pares que suman " << k << " son: " << endl;

for (*auto* it = pares.begin(); it != pares.end(); ++it)

{

cout << "(" << it->first << ", " << it->second << ")" << endl;

}

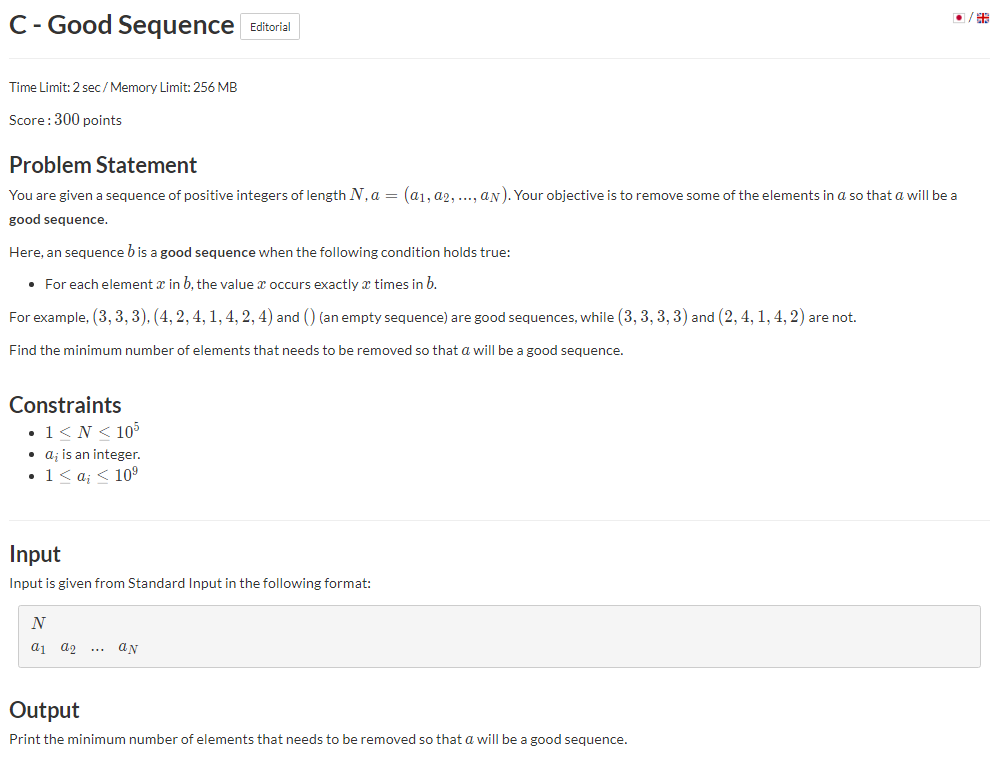
return 0;

}

La solución utiliza la estructura de datos SET de STL para almacenar los números del conjunto y un pair para almacenar los pares que sumen el valor k. Para ello, se recorre el conjunto de números y se busca el número complementario en el conjunto. Si se encuentra el número complementario, se agrega el par al conjunto de pares. En el código, se utiliza la función find de la STL para buscar el número complementario en el conjunto. Si se encuentra el número complementario, se verifica si el número actual es menor que el complementario para agregar el par ordenado al conjunto de pares. Finalmente, se muestran en pantalla todos los pares que suman el valor k.

**Ejercicio 2:**

**Link:** <https://atcoder.jp/contests/arc087/tasks/arc087_a>



#include <iostream>

#include <map>

typedef *long* *long* ll;

using *namespace* std;

*int* main()

{

*int* n;

cin >> n;

map<*int*, *int*> m;

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

{

*int* x;

cin >> x;

m[x]++;

}

ll sum = 0;

for (*auto* i : m)

{

if (i.second > i.first)

{

sum += i.second - i.first;

}

else if (i.first > i.second)

{

sum += i.second;

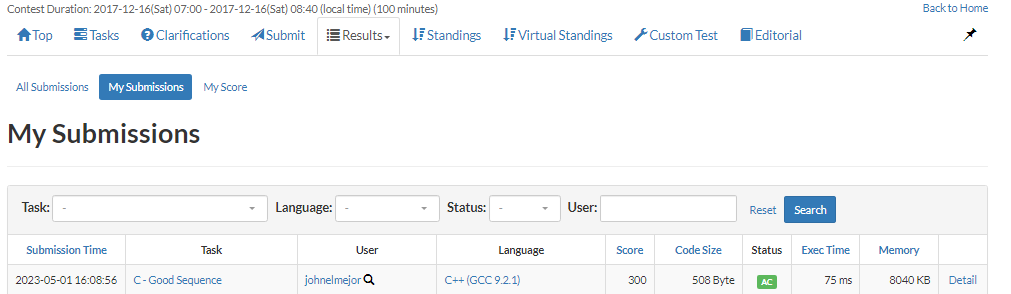
}

}

cout << sum;

return 0;

}



**Ejercicio 3:**

Desarrollar un ejercicio con texto y un patrón, implemente el algoritmo KMP para implementar lo siguiente:

* Ingresar un número entero T que indica el número de casos de prueba, el número debe ser mayor a 1 y menor a 10000.
* Para cada caso de prueba se debe ingresar o agregar los valores de manera aleatoria al patrón y al texto, el tamaño del patrón y el texto debe ser definido por el desarrollador.
* Encontrar todas las apariciones del patrón en el texto.
* Cuando se encuentre el patrón en el texto, debe remover los dígitos del patrón del texto, y mostrar la nueva cadena de texto.
* Si el patrón es encontrado por más de 3 veces debe mostrar un mensaje que diga: “El patrón es muy frecuente en el texto.
* Si no encuentra el patrón en el texto, mostrar un mensaje que diga “No encontró patrón”.

**#include <iostream>**

**#include <vector>**

**using *namespace* std;**

**vector<*int*> KMPpreprocess(string *pattern*)**

**{**

***int* m = pattern.length();**

**vector<*int*> pi(m);**

**pi[0] = 0;**

***int* j = 0;**

**for (*int* i = 1; i < m; i++)**

**{**

**while (j > 0 && pattern[j] != pattern[i])**

**{**

**j = pi[j - 1];**

**}**

**if (pattern[j] == pattern[i])**

**{**

**j++;**

**}**

**pi[i] = j;**

**}**

**return pi;**

**}**

***void* KMPsearch(string *text*, string *pattern*)**

**{**

***int* cant\_apariciones = 0;**

***int* n = text.length();**

***int* m = pattern.length();**

**vector<*int*> pi = KMPpreprocess(pattern);**

**cout << endl;**

***int* j = 0;**

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

**{**

**while (j > 0 && pattern[j] != text[i])**

**{**

**j = pi[j - 1];**

**}**

**if (pattern[j] == text[i])**

**{**

**j++;**

**}**

**if (j == m)**

**{**

**cout << "Patron encontrado en el indice: " << i - m + 1 << endl;**

**j = pi[j - 1];**

***int* first\_pos = i - m + 1;**

***int* last\_pos = i + 1;**

**text.erase(text.begin() + first\_pos, text.begin() + last\_pos);**

**cout << "Nuevo texto: " << text << endl;**

**cant\_apariciones++;**

**}**

**}**

**if (cant\_apariciones > 3)**

**{**

**cout << "El patron es muy frecuente en el texto" << endl;**

**}**

**else if (cant\_apariciones == 0)**

**{**

**cout << "No encontro patron" << endl;**

**}**

**}**

***int* main()**

**{**

**srand(time(NULL));**

***int* t;**

**cout << "Ingrese el numero de casos de prueba: ";**

**do**

**{**

**cin >> t;**

**} while (t <= 1 || t >= 10000);**

**while (t--)**

**{**

**string text, pattern;**

***int* tamano\_text = 60, tamano\_pattern = 3;**

**for (*int* i = 0; i < tamano\_text; i++)**

**{**

**text.push\_back(rand() % 26 + 97);**

**}**

**for (*int* i = 0; i < tamano\_pattern; i++)**

**{**

**pattern.push\_back(rand() % 26 + 97);**

**}**

**cout << endl;**

**cout << "Texto: " << text << endl;**

**cout << "Patron: " << pattern << endl;**

**KMPsearch(text, pattern);**

**}**

**return 0;**

**}**