Curso de Programación en C++.
TPL4-2021. Trabajo Práctico de Laboratorio. [2021-07-28]

Ejercicios

[Ej. 1] **intsermax (20 puntos)**: Escribir una función
`set<int> intsermax(vector< set<int> > &VS,set<int> &Q);`
que, dado un vector de conjuntos `VS` determina cuál de ellos es el que tiene mayor intersección con el conjunto `Q`.

```
VS:[[0,1,2,3],[2,3,4,6]] Q:[1,2,3]                          -> Smax:[0,1,2,3]
VS:[[1,3,5,7],[0,1,6,7,8],[1,3,5,7,8],[1,4,5,8]] Q:[5,7,8] -> Smax:[1,3,5,7,8]
VS:[[3,4,8],[2,3,4,5,6],[1,4,6]] Q:[0,1,5,8]                -> Smax:[3,4,8]
```

Por ejemplo, en el primer caso el conjunto `Q` tiene intersección de tamaño 3 y 2 respectivamente con los conjuntos de `VS` por lo tanto debe retornar el primer conjunto `[0,1,2,3]`.

[Ej. 2] **flatten (30 puntos)**: Escribir una función `void flatten(json &nested,json &flat);`
que toma un `json nested` compuesto de elementos de tipo *number*, *string*, o *null* y posiblemente anidados en arreglos (el anidamiento puede tener varios niveles) y lo convierte en un arreglo plano (*flat*) de *number*, *string* o *null*. Por ejemplo,

```
nested=[1,2,3,[4,5],[6,7,8]] -> flat=[1,2,3,4,5,6,7,8],
nested=["Tokio",2,"Paris",["dog",null],[23,["pi",3.14],8]]
-> flat=["Tokio",2,"Paris","dog",null,23,"pi",3.14,8]
nested=["dog",[["Tokio",23,2,"Paris"],null,[2,["Paris","dog",null,"Paris"],"Tokio"]]]
-> flat=["dog","Tokio",23,2,"Paris",null,2,"Paris","dog",null,"Paris","Tokio"],
```

Ayuda: Usar **recursividad**,

- Si `nested` es un arreglo aplicar recursivamente `flatten()` a los hijos.
- Caso contrario apendizar el elemento `nested` a `flat`.
- Recordar que `nlohmann::json` tiene *predicados* para determinar los diferentes tipos básicos que puede ser un dado `json`.

Nota: Notar que en la práctica equivale a remover los corchetes de apertura y cierre, salvo el par más externo. Pero no deben usar esta estrategia, lo deben hacer en forma recursiva.

[Ej. 3] **maxvol (30 puntos)**: Escribir una función `double maxvol(MatrixXd &xnodes);`
que retorna el máximo volumen de todos los tetrahedros que se pueden formar con los vértices en `xnodes`.

Siendo `xnodes` una matriz de `(nnodes,3)` donde cada fila representa un vector en 3D, para cada 4 de esos vértices podemos formar un tetrahedro y calcular su volumen. La consigna consiste en buscar el máximo volumen de todos esos posibles tetrahedros.

Ayuda:

- Recordar que el volumen V del tetrahedro que tiene por vértices los vectores x_0, x_1, x_2, x_3 está dado por

$$V = \frac{1}{6}|\det(A)|, \quad \text{donde } A = \begin{bmatrix} x_0 & 1 \\ x_1 & 1 \\ x_2 & 1 \\ x_3 & 1 \end{bmatrix}$$

  Donde $\det(\,\cdot\,)$ denota el determinante de la matriz correspondiente y notar que los vectores x_j se colocan como filas en la matriz. La última columna de la matriz debe tener todos 1.
- Hacer un cuádruple lazo anidado sobre **`i,j,k,l`** en **`[0,nnod)`**.
- Dados esos cuatro índices, extraer las filas correspondientes de **`xnodes`** y ubicarlas en **`A`** completando la última columna con 1's.
- Calcular el volumen del tetrahedro usando la fórmula anterior (recordar usar el valor absoluto, ya que el determinante puede dar negativo).
- Mantener el máximo de los volúmenes en una variable auxiliar y retornar al finalizar el lazo.
- **Nota:** Durante la ejecución del lazo vamos a pasar por tuplas **`i,j,k,l`** donde algunos de los índices pueden ser iguales. En ese caso el volumen del tetrahedro colapsa en un triángulo y el volumen es nulo. No hay problema con esto, ya que estamos calculando el volumen máximo. Igualmente, pasamos el mismo tetrahedro varias veces, por ejemplo al calcular la tupla **`(0,1,2,3)`** y la **`(3,2,1,0)`**, pero de nuevo, no importa porque estamos calculando el máximo.
- El primer caso de prueba en **`maxvol.json`** consiste de 8 puntos en los vértices del cubo $[0,1] \times [0,1] \times [0,1]$. Como chequeo, en ese caso los volúmenes que se van a obtener al recorrer las tuplas pueden ser 0, $1/6$ y $1/3$. Por supuesto la función debe retornar el máximo, o sea $1/3$.