```
//---------------------------------------------------------------------------
// File:   informe_P4_839841_841537.pdf
// Author: Ignacio Millán y Carlota Moncasi
// Date:   noviembre 2022
// Coms:   práctica 4 trabajo previo de PSCD
//---------------------------------------------------------------------------

#include <Semaphore_V4.hpp >
using namespace std;
// -----------------------------------------------------
const int N_EST = 60; //# de estudiantes
const int N_FIL = N_EST /2; //# de filas en la matriz
const int N_COL = 1000; //# de columnas

// -----------------------------------------------------
// Pre : <fila > es un índice de fila de <D>
// Post : devuelve el máximo de la fila <fila >
int maxFila (int D[ N_FIL ][ N_COL ] , int fila ) {
int max = D[fila][0];
for (int i = 1; i<N_COL; i++){
      if(D[fila][i] > max){
                  max = D[fila][i];
      }
 }
return max ;
}

// -----------------------------------------------------
// Pre : <fila > es un índice de fila de <D>
// Post : devuelve la suma de los els. de la fila <fila >
int sumaFila ( int D[ N_FIL ][ N_COL ] , int fila ) {
int sum = 0;
for (int i = 0; i<N_COL; i++){
  sum = sum + D[fila][i];
 }
 return sum ;
}

// -----------------------------------------------------
void Estudiante (int nip, int D[ N_FIL ][ N_COL ], int pareja[], int
nips[], int& sillasOcupadas)  {
// esperar por una silla libre
<await(sillasOcupadas<2)
      nips[sillasOcupadas]=nip;
      sillasOcupadas++;
>
// esperar me sea asignada pareja y fila
<await(pareja[nip]!=-1)
      if (nip < pareja[nip]) {
```

```
      // calcular máx de mi fila
      int max= maxFila(D, fila);
      // hacérselo llegar a mi pareja
      resultados[nip]=max;
      <aviso[nip]=true;>
      }
      else
      {
        // calcular la suma de mi fila
      suma= sumaFila(D, fila);
        // coger info de max (de mi pareja )
      <await(aviso[pareja[nip]])
           resultados[nip]=suma;
           // mostrar resultados
           cout <<setw(3)<< filas[nip] << "|" <<setw(3)<< pareja[nip]
           << "-" <<setw(2)<< nip << " |" <<setw(6)<<
           resultados[pareja[nip]] << " | " << suma << endl;

         // comunicar finalización
         <aviso[nip]=true;>
         parejasTerminadas++;
      >
      }
>
}

// ------------------------------------------------
void Profesor (int pareja[], int nips[],
int& sillasOcupadas) {
for( int i=0; i< N_FIL ; i++) {
// esperar a que haya dos
<   await (sillasOcupadas == 2)
// comunicar a cada uno su pareja , y la fila que les toca
   pareja[nips[0]]=nips[1];
   pareja[nips[1]]=nips[0];
   fila[nip[0]]=i;
   fila[nips[1]]=i;
   sillasOcupadas=0;
>
}
// esperar que todos hayan terminado
<await (parejasTerminadas==N_FIL)>
}

// ------------------------------------------------
                int main () {

int D[ N_FIL ][ N_COL ]; // para almacenar los datos
int fila = 0; // cada pareja cogerá una
int pareja [ N_EST ]; // pareja [i] será la pareja asignada
int sillasOcupadas = 0;
int nips[2];
int resultados[N_EST];
int filas[N_EST];
```

```
bool aviso[N_EST];
int nip = 0;
int parejasTerminadas=0;

//inicializar vectores
  for(unsigned i=0; i<N_EST; i++){
        resultados[i]=-1;
        filas[i]=-1;
        aviso[i]=false;
        pareja[i]=-1;
   }


// cargar " datos.txt " en "D"

    ifstream f;
    f.open("datos.txt");
    if (f.is_open()) {
        while(!f.eof()) {
            for(int i = 0; i < N_FIL; i++){
                for(int j = 0; j<N_COL; j++){
                    f>>D[i][j];
                }
            }
        }
        f.close();
    }
    else{
        cerr << "NO SE HA PODIDO ABRIR." << endl;
    }
}
// threads
cout << "\n Prueba finalizada \n";
return 0;
}
```