1) Atención por guardia (1 punto)
Desde el Hospital Fernandez nos pidieron solucionar una serie de problemas relacionados con la información que maneja
sobre los pacientes y el personal de salud. En primer lugar debemos resolver en qué orden se deben atender los pacientes
que llegan a la guardia. En enfermería, hay una primera instancia que clasifica en dos colas a los pacientes: una urgente y
otra postergable (esto se llama hacer triage). A partir de dichas colas que contienen la identificación del paciente, se pide
devolver una nueva cola según la siguiente especificación.

problema orden_de_atencion (in urgentes: cola <Z>, in postergables: cola <Z>): cola <Z> {
requiere: {no hay elementos repetidos en urgentes}
requiere: {no hay elementos repetidos en postergables}
resquiere: {la intersección entre postergables y urgentes es vacía}
requiere: {|postergables| = |urgentes|}
asegura: {no hay repetidos en res}
asegura: {res es permutación de la concatenación de urgentes y postergables}
asegura: {Si urgentes no es vacía, en tope de res hay un elemento de urgentes}
asegura: {En res no hay dos seguidos de urgentes}
asegura: {En res no hay dos seguidos de postergables}
asegura: {Para todo c1 y c2 de tipo "urgente" pertenecientes a urgentes si c1 aparece antes que c2 en urgentes entonces c1
aparece antes que c2 en res}
asegura: {Para todo c1 y c2 de tipó "postergable" pertenecientes a postergables si c1 aparece antes que c2 en postergables
entonces c1 aparece antes que c2 en res}
}
2) Alarma epidemiológica (3 puntos)
Necesitamos detectar la aparición de posibles epidemias. Para esto contamos con una lista de enfermedades infecciosas y
los registros de atención por guardia dados por una lista de expedientes. Cada expediente es una tupla con ID paciente y
enfermedad que motivó la atención. Debemos devolver un diccionario cuya clave son las enfermedades infecciosas y su
valor es la proporción de pacientes que se atendieron por esa enfermedad. En este diccionario deben aparecer solo
aquellas enfermedades infecciosas cuya proporción supere cierto umbral.

problema alarma_epidemiologica(registros: seq<ZxString>, infecciosas: seq<String>, umbral: R): dict<String,R>{
requiere: {0 < umbral < 1}
asegura: {claves de res pertenecen a infecciosas}
asegura: {Para cada enfermedad perteneciente a infecciosas, si el porcentaje de pacientes que se atendieron por esa
enfermedad sobre el total de registros es mayor o igual al umbral, entonces res[enfermedad] = porcentaje}
asegura: {Para cada enfermedad perteneciente a infecciosas, si el porcentaje de pacientes que se atendieron por esa
enfermedad sobre el total de registros es menor que el umbral, entonces enfermedad no aparece en res}
}
3) Empleados del mes (2 puntos)
Dado un diccionaio con la cantidad de horas trabajadas por empleado, en donde la clave es el ID del empleado y el valor es
una lista de las horas trabajadas por día, queremos saber quienes trabajaron más para darles un premio.

Se deberá buscar
la o las claves para la cual se tiene el máximo valor de cantidad total de horas, y devolverlas en una lista.

problema empleados_del_mes(horas: dicc<Z,seq<Z>>): seq<Z> {
requiere: {No hay valores en horas que sean listas vacías}
asegura: {Si ID pertenece a res entonces ID pertenece a las claves de horas}
asegura: {Si ID pertenece a res, la suma de sus valores de horas es el máximo de la suma de elementos de horas de todos
los otros IDs}
asegura: {Para todo ID de claves de horas, si la suma de sus valores es el máximo de la suma de elementos de horas de los
otros IDs, entonces ID pertenece a res}
}

4) Nivel de ocupacion del hospital

Queremos saber qué porcentaje de ocupación de camas hay en el hospital. El hospital se representa por una matriz en
donde las filas son los pisos, y las columnas son las camas. Los valores de la matriz son booleanos que indican si la cama
está ocupada o no. Si el valor es verdadero (True) indica que la cama está ocupada. Se nos pide programar en python una
función que devuelve una secuencia de enteros, indicando la proporción de camas ocupadas en cada piso.

problema nivel_de_ocupacion(camas_por_piso: seq<Seq<Bool>>): seq<R> {
requiere: {Todos los pisos tienen la misma cantidad de camas}
requiere: {Hay por lo menos 1 piso en el hospital}
requiere: {Hay por lo menos una cama por piso}
asegura: {|res| = |camas|}
asegura: {Para todo 0 <= i < |res| se cumple que res[i] es igual a la cantidad de camas ocupadas del piso i dividido del total
de camas del piso i}
}