Introducción a la Programación

Algoritmos y Estructuras de Datos I

Segundo cuatrimestre de 2025

Departamento de Computación - FCEyN - UBA

Ejercicios tipo Parcial

Ejercicio tipo parcial - Nivel de ocupación del hospital

Vamos a representar un hospital con 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.

Ejercicio tipo parcial - Nivel de ocupación del hospital

Vamos a representar un hospital con 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.

`problema nivel_de_ocupacion` (in camas_por_piso:$seq\langle seq\langle$Bool$\rangle\rangle$) : $seq\langle\mathbb{R}\rangle$ {
  `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| es igual a la cantidad de pisos del hospital.}
  `asegura:` {Para todo $0 \leq$ i $<$ |res| se cumple que res[i] es igual a la cantidad de camas ocupadas del piso i dividido el total de camas del piso i).}
}

Ejercicio tipo parcial - Nivel de ocupación del hospital

Vamos a representar un hospital con 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.

`problema nivel_de_ocupacion` (in camas_por_piso:$seq\langle seq\langle$Bool$\rangle\rangle$) : $seq\langle\mathbb{R}\rangle$ {
  `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| es igual a la cantidad de pisos del hospital.}
  `asegura:` {Para todo $0 \leq$ i $<$ |res| se cumple que res[i] es igual a la cantidad de camas ocupadas del piso i dividido el total de camas del piso i).}
}

Ejemplo: dada la entrada **camas_por_piso = [[True, False, True], [False, False, True], [True, True, True]]**, devuelve **res = [2/3, 1/3, 1.0]**.

Ejercicio tipo parcial - Modificando una matriz

```
problema cambiar_matriz (inout A: seq⟨seq⟨ℤ⟩⟩) {
  requiere: {Todas las filas de A tienen la misma longitud}
  requiere: {El mínimo número que aparece en A es igual a 1}
  requiere: {El máximo número que aparece en A es igual a #filas de A por
        #columnas de A}
  requiere: {No hay enteros repetidos en A}
  requiere: {Existen al menos dos enteros distintos en A}
  modifica: {A}
  asegura: {A tiene exactamente las mismas dimensiones que A@pre}
  asegura: {El conjunto de elementos que aparecen en A es igual al conjunto de
        elementos que aparecen en A@pre}
  asegura: {A[i][j] != A@pre[i][j] para todo i, j en rango}
}
```

Ejercicio tipo parcial - Modificando una matriz

```
problema cambiar_matriz (inout A: seq⟨seq⟨ℤ⟩⟩) {
  requiere: {Todas las filas de A tienen la misma longitud}
  requiere: {El mínimo número que aparece en A es igual a 1}
  requiere: {El máximo número que aparece en A es igual a #filas de A por
        #columnas de A}
  requiere: {No hay enteros repetidos en A}
  requiere: {Existen al menos dos enteros distintos en A}
  modifica: {A}
  asegura: {A tiene exactamente las mismas dimensiones que A@pre}
  asegura: {El conjunto de elementos que aparecen en A es igual al conjunto de
        elementos que aparecen en A@pre}
  asegura: {A[i][j] != A@pre[i][j] para todo i, j en rango}
}
```

Ejemplo: dada la entrada **A = [[1,2,3], [4,5,6]]**, una posible solución es **A = [[4,5,6], [1,2,3]]**.

Ejercicio tipo parcial - Promedio de salidas

Dado un diccionario donde la clave es el nombre de cada amigo y el valor es una lista de los tiempos (en minutos) registrados para cada sala de escape en Capital, escribir una función en Python que devuelva un diccionario.

`problema promedio_de_salidas` (in registro: dict[str, $seq\langle\mathbb{Z}\rangle$]) : dict[str, $\mathbb{Z}\times\mathbb{R}$] {
  `requiere:` {*registro* tiene por lo menos un integrante.}
  `requiere:` {Todos los integrantes de *registro* tienen por lo menos un tiempo.}
  `requiere:` {Todos los valores de *registro* tienen la misma longitud.}
  `requiere:` {Todos los tiempos de los valores de *registro* están entre 0 y 61 inclusive.}
  `asegura:` {**res** tiene exactamente las mismas claves que registro.}
  `asegura:` {El primer elemento de la tupla de **res** para un integrante, es la cantidad de salas con tiempo mayor estricto a 0 y menor estricto a 61 que figuran en sus valores de registro.}
  `asegura:` {El segundo elemento de la tupla de res para un integrante, si la cantidad de salas de las que salió es mayor a 0: es el promedio de salas con tiempo mayor estricto a 0 y menor estricto a 61 que figuran en sus valores de registro; sino es 0.0.}
}

Ejercicio tipo parcial - Promedio de salidas

Dado un diccionario donde la clave es el nombre de cada amigo y el valor es una lista de los tiempos (en minutos) registrados para cada sala de escape en Capital, escribir una función en Python que devuelva un diccionario.

`problema promedio_de_salidas` (in registro: dict[str, $seq\langle\mathbb{Z}\rangle$]) : dict[str, $\mathbb{Z}\times\mathbb{R}$] {
  `requiere:` {*registro* tiene por lo menos un integrante.}
  `requiere:` {Todos los integrantes de *registro* tienen por lo menos un tiempo.}
  `requiere:` {Todos los valores de *registro* tienen la misma longitud.}
  `requiere:` {Todos los tiempos de los valores de *registro* están entre 0 y 61 inclusive.}
  `asegura:` {**res** tiene exactamente las mismas claves que registro.}
  `asegura:` {El primer elemento de la tupla de **res** para un integrante, es la cantidad de salas con tiempo mayor estricto a 0 y menor estricto a 61 que figuran en sus valores de registro.}
  `asegura:` {El segundo elemento de la tupla de res para un integrante, si la cantidad de salas de las que salió es mayor a 0: es el promedio de salas con tiempo mayor estricto a 0 y menor estricto a 61 que figuran en sus valores de registro; sino es 0.0.}
}

Ejemplo: dada la entrada {**“a”:[61,60,59,58], “b”:[1,2,3,0]** }, la salida es {**“a”:(3, 59.0), “b”:(3, 2.0)**}.

Ejercicio tipo parcial - Cola en el Banco

En el banco ExactaBank los clientes hacen cola para ser atendidos por un representante. Los clientes son representados por las tuplas (nombre, tipo afiliado) donde la primera componente es el nombre y el segundo componente es el tipo del afiliado, el cual puede ser "común" o "vip".

`problema reordenar_cola_priorizando_vips` (in filaClientes: Cola⟨str × str⟩) : Cola⟨str⟩ {
  `requiere:` {La longitud de los valores de la primera componente de las tuplas de la cola *filaClientes* es mayor a 0.}
  `requiere:` {Los valores de la segunda componente de las tuplas de la cola *filaClientes* son "común" o "vip".}
  `requiere:` {No hay dos tuplas en filaClientes que tengan la primera componente iguales entre sí.}
  `asegura:` {Todos los elementos de **res** aparecen como primera componente de alguna tupla de *filaClientes*}
  `asegura:` {$|\textbf{res}| = |filaCliente|$.}
  `asegura:` {res no tiene elementos repetidos.}
  `asegura:` {No hay ningún cliente "común" antes que un "vip" en **res**.}
  `asegura:` {Para todo cliente c1 y cliente c2 de tipo "común" pertenecientes a *filaClientes*, si c1 aparece antes que c2 en filaClientes entonces el nombre de c1 aparece antes que el nombre de c2 en **res**.}
  `asegura:` {Para todo cliente c1 y cliente c2 de tipo "vip" pertenecientes a filaClientes si c1 aparece antes que c2 en filaClientes entonces el nombre de c1 aparece antes que el nombre de c2 en **res**.}
}

Ejercicio tipo parcial - Cola en el Banco

En el banco ExactaBank los clientes hacen cola para ser atendidos por un representante. Los clientes son representados por las tuplas (nombre, tipo afiliado) donde la primera componente es el nombre y el segundo componente es el tipo del afiliado, el cual puede ser "común" o "vip".

`problema reordenar_cola_priorizando_vips` (in filaClientes: Cola⟨str × str⟩) : Cola⟨str⟩ {
  `requiere:` {La longitud de los valores de la primera componente de las tuplas de la cola *filaClientes* es mayor a 0.}
  `requiere:` {Los valores de la segunda componente de las tuplas de la cola *filaClientes* son "común" o "vip".}
  `requiere:` {No hay dos tuplas en filaClientes que tengan la primera componente iguales entre sí.}
  `asegura:` {Todos los elementos de **res** aparecen como primera componente de alguna tupla de *filaClientes*}
  `asegura:` {$|\textbf{res}| = |filaCliente|$.}
  `asegura:` {res no tiene elementos repetidos.}
  `asegura:` {No hay ningún cliente "común" antes que un "vip" en **res**.}
  `asegura:` {Para todo cliente c1 y cliente c2 de tipo "común" pertenecientes a *filaClientes*, si c1 aparece antes que c2 en filaClientes entonces el nombre de c1 aparece antes que el nombre de c2 en **res**.}
  `asegura:` {Para todo cliente c1 y cliente c2 de tipo "vip" pertenecientes a filaClientes si c1 aparece antes que c2 en filaClientes entonces el nombre de c1 aparece antes que el nombre de c2 en **res**.}
}

Ejemplo: dada la entrada filaClientes conteniendo los elementos en el orden ("Ana", "comun"), ("Juli", "vip") y ("Fede", "vip"), la salida esperada no modifica filaClientes, y devuelve una cola con los siguientes elementos en este orden: "Juli", "Fede","Ana".