Universidad de Granada

Práctica 2: Programación

Pablo Cordero Romero

77035152X

Ejercicio 1

Como el ejercicio pide asignar un valor en el rango de 0 a 9 para cada letra, he creado una variable para cada letra que pueda tomar un valor en este rango.

En cuanto a las restricciones, para la primera, que no haya dos letras con el mismo valor asignado, he comparado todas las letras entre sí asegurándome de que todas tienen un valor diferente. Por otra parte, para que la suma se cumpla, me he basado en la siguiente regla:

$$A \cdot 1000 + B \cdot 100 + C \cdot 10 + D = ABCD$$

Multiplicando el valor de cada letra por el número correspondiente de 0's, según su posición, podemos generar una ecuación similar a la del enunciado.

Ejercicio 2

Para este problema, he definido el número X como un vector de tamaño 10 (de 0 a 9). Cada posición del array corresponde a un dígito del número en su orden correspondiente.

Para que se cumpla lo que se indica en el enunciado, he usado la función `forall` y `count` para la única restricción del programa. La restricción se encarga de comprobar que para cada posición del vector, denotada como i, se cumpla que el número de veces que aparece i en el vector (calculado por la función `count`) sea igual que el valor de esa posición del vector.

Por último he modificado la salida para que muestre el número y no el vector.

Ejercicio 3

Para este, he creado un vector con índices de 9 a 14 (6 posiciones), donde cada posición representa a una hora. Las posiciones de este vector pueden tomar valores de 1 a 6, siendo estos el identificador de cada profesor.

En cuanto a restricciones, he tenido en cuenta dos. La primera es que todos los valores del vector deben ser diferentes. Esto se traduce en que un profesor no puede tomar el aula dos veces, ya que hay 6 horas disponibles y cada profesor imparte una clase de 1 hora. La segunda restricción se traduce realmente en 6. Teniendo como identificador de un profesor i, i solo puede aparecer 1 vez en el subvector correspondiente a su horario. Para comprobar las veces que aparece en el subvector uso la función `count`.

Por último, para que la solución sea más visible he modificado la salida, haciendo que aparezca cada horario asociado al profesor correspondiente.

Ejercicio 4

Para este ejercicio, he usado una matriz como única variable. Cada fila de la matriz corresponde a una asignatura junto con su grupo como viene indicado en el archivo del ejercicio. La columnas de la

matriz son dos, correspondiendo la columna 1 a el aula asignada a cada clase y la columna 2 la hora asignada a cada clase.

En cuanto a restricciones, he tomado las siguientes:

- Como los valores de la matriz son enteros, he tenido que añadir una restricción que compruebe que los valores de la columna de las aulas están entre 1 y 4 y los valores entre de la columna de las horas están entre 9 y 12. esto lo he hecho sobre un `forall` que indica que esto se tiene que cumplir para todas las filas.
- Para que a un profesor no se le asigne un horario cuando no está disponible, he indicado que en las filas correspondientes a los grupos que imparten no podrá tomarse como valor la hora en la que no está disponible. Esto lo he hecho con simples comparaciones y operadores `AND`.
- Para que a un profesor no se le asigne la misma hora dos veces, he tenido que comparar que todas las asignaturas que imparte tengan una hora diferente asignada.
- Para que un grupo no tenga dos clases a la misma hora, he comparado al igual que antes que las diferentes asignaturas de un grupo no se puedan impartir a la misma hora.
- Por último, para asegurarme de que un aula no sea asignada dos veces a la misma hora, he indicado mediante un `forall` que en la matriz, dos filas diferentes de la matriz pueden tener distinta hora, pero en caso de tener la misma, deben tener también distinta aula.

Por último, he modificado el output para que la salida sea más legible. He indicado para cada grupo su asignación de aula y hora. La salida obtenida en mi caso sería este horario:

	9:00	10:00	11:00	12:00
Aula 1	FBD-G2 (P2)	IA-G3 (P4)	IA-G4 (P4)	IA-G2 (P1)
Aula 2	TSI-G1 (P1)	TSI-G4 (P3)	FBD-G1 (P2)	TSI-G3 (P3)
Aula 3	FBD-G4 (P3)	IA-G1 (P1)	FBD-G3 (P3)	
Aula 4			TSI-G2 (P1)	

Ejercicio 5

En este caso he usado una matriz como en el anterior ejercicio donde cada fila representa a un bloque al que se le tiene que asignar un horario. En cuanto a las columnas, la primera representa el día asignado para ese bloque y la segunda representa la hora asignada.

En cuanto a las restricciones he tomado las siguientes:

- La primera columna 1 solo podrá tomar valores del 1 al 5 ya que representará el día asignado (de lunes a viernes). La segunda columna solo pordrá tomar valores de las 8 a las 13.
- Las siguientes restricciones del código vienen a controlar la asignación de horas a los bloques que necesitan dos horas. El primer conjunto de restricciones controla que un bloque de 2 horas

no pueda ser puesto a las 10 (ya que solaparía con el recreo) y a las 13 (ya que pasaríamos de las 14:00). El segundo conjunto de restricciones lo que busca es que cuando a un bloque de dos horas tenga asignada una hora, no se le asigne otro bloque a la hora siguiente, ya que este ocupará dos horas.

- El siguiente conjunto de instrucciones, busca cumplir que dos bloques de la misma asignatura no tengan el mismo día asignado. Para esto compruebo que los días de los bloques de la misma asignatura no sean el mismo.
- La siguiente condición que busco cumplir es la de que los profesores P1, P2 y P3 no pueden tener asignado dos clases el mismo día. Para esto comparo que todos los bloques de distintas asignaturas que imparten tengan diferente día asignado.
- La siguiente restricción es que ninguna clase puede ser asignada a las 11 (hora del recreo).
- La última restricción viene a cumplir que no coincidan para dos bloques distintos la hora y el día de clase.

Por último, al igual que el ejercicio anterior, he modificado la salida par que se muestre de forma más legible. Muestro para cada bloque la asignación dada en día y hora. La salida que he obtenido mostrada como horario es la siguiente:

	Lunes	Martes	Miércoles	Jueves	Viernes
8:00	A5 (P2)	A8 (P4)	A5 (P2)	A4 (P2)	A4 (P2)
9:00					
10:00	A7 (P4)	A2 (P4)	A9 (P3)	A2 (P4)	A6 (P3)
11:00			RECREO		
12:00	A3 (P1)	A3 (P1)	A1 (P1)	A7 (P4)	A1 (P1)
13:00				A6 (P3)	

Ejercicio 6

Para este ejercicio he tomado una matriz de enteros del 1 al 5 donde cada fila representa a una persona y cada columna representa un atributo de la persona. La correspondencia de valores tanto de las filas como las columnas se puede ver en los comentarios del archivo `ejer6.mzn`.

Para las restricciones he hecho una orden por cada condición mostrada en el enunciado. Para las condiciones del tipo *"El vasco vive en la casa roja"* lo único que he hecho ha sido aplicar la restricción de que la posición correspondiente de la matriz tiene que ser el valor indicado por la condición. Por otra parte, las condiciones del tipo *"El de la casa verde bebe café"* donde implica a varias características uso un `forall` donde dentro hay una condición `XNOR` que implica que cualquier individuo debe cumplir o incumplir ambas características, pero no una sola. Por último para las condiciones que implican dos individuos como puede ser *"La casa verde está al lado de la blanca y a su derecha"* he usado un `forall` con los dos elementos que, en el caso de que cumplan las condiciones, por ejemplo, casa verde y casa blanca, tiene que cumplir sí o sí la condición que indica el enunciado.

Además, he añadido una restricción para indicar que en cada una de las columnas no se puede repetir ningún valor. He modificado la salida para que muestre la matriz más entendible y he llegado al resultado: **El gallego tiene a la cebra y el andaluz bebe agua**.

Ejercicio 7

Para resolver este ejercicio he usado una matriz donde cada fila representa una tarea, la columna 1 representa el día de inicio de la tarea y la columna 2 el día que ya ha finalizado (día siguiente al último día).

La primera restricción tomada es que las casillas de la matriz deben ser todas ≥ 0, ya que no hay días negativos. El segundo bloque de restricciones tiene como objetivo determinar la duración de las tareas. Por ejemplo, para la tarea A se indica que debe de haber 7 días de diferencia entre el día de inicio y el día de fin. El último bloque de restricciones, busca cumplir que las tareas siempre vayan después de sus predecesores. Para esto indico que el día de inicio de una tarea, tiene que ser $\leq$ que el tiempo de fin de todos sus predecesores.

Para optimizar el resultado, he pedido a minizinc, mediante la orden `solve minimize`, que minimice la variable `t`, la cual siempre tomará valor de la fecha de finalización de la última tarea. Esto se podría resumir como tomar el valor más alto de la columna correspondiente a los días de finalización.

La construcción de la casa con estas restricciones se puede realizar en **18 días**.

Ejercicio 10

En este ejercicio he usado un array y una matriz. El array represente si he tomado o no los diferentes objetos, siendo el valor 1 si lo he tomado y 0 si no. Por otra parte, la matriz representará el peso y preferencia de cada uno de los objetos, siendo 0 ambos valores cuando ese objeto no ha sido tomado.

Como restricciones, he realizado una por objeto, donde, si es cogido el objeto, se introduce su peso y preferencia en la matriz correspondiente. Si no es cogido, estos valores son 0. El peso es una variable calculada como la suma de todos los pesos de la columna de la matriz. La preferencia será más de lo mismo. La última restricción será que la variable que representa el peso no puede sobrepasar los 275 kg.

He indicado que el ejercicio sea resuelto maximizando la variable correspondiente a la preferencia y he obtenido una mochila de **peso=274, preferencia=705, con los objetos: Mapa, Compás, Agua, Sandwich, Azúcar, Queso y Protector solar**.