2.1. Sean A= {1, 2, 3, 7, 8} y B= {3, 4, 5, 7, 9}, mostrar la ejecución y los resultados de las siguientes operaciones, suponiendo una representación con arrays de booleanos y con listas de elementos (ordenadas y no ordenadas).

a) Unión (A, B, C)
b) Intersección (A, B, C)
c) Diferencia (A, B, C)
d) Miembro (1, A)
e) Inserta (1, A)
f) Suprime (1, A)
g) Min (A)

2.2. La realización de conjuntos mediante arrays de booleanos se puede usar siempre que el conjunto universal se pueda "traducir" a los enteros de 1 a N. Describir cómo se haría esta traducción (en caso de ser posible) si el conjunto universal fuera:

a) los enteros de n a m, para cualquier $n \leq m$
b) los enteros n, n+2, n+4, ..., n+2k, para cualesquiera n y k
c) los cuadrados perfectos 1, 4, 9, 16, 25, 36, 49, ...
d) los caracteres 'a', 'b', ..., 'z'
e) arrays de dos caracteres, cada uno de ellos entre 'a' y 'z'
f) los números reales de 0 a 10
g) las cadenas de hasta 20 caracteres.

2.3. Suponiendo la definición del TAD **Conjunto [T]** ¿cómo varía la especificación si se realiza la implementación con arrays de booleanos o con listas de elementos?

2.4. Suponer que estamos insertando enteros en una tabla de dispersión de 7 cubetas, utilizando la función de dispersión h(k)= k mod 7.

a) Mostrar la tabla de dispersión abierta si se insertan los valores 1, 8, 27, 125, 216, 343.
b) Repetir el apartado a) usando una tabla de dispersión cerrada con resolución lineal de colisiones.
c) ¿En cuál de los casos anteriores se deben realizar menos operaciones? ¿En cuál se utiliza menos memoria?

2.5. Estamos usando una tabla de dispersión con cinco cubetas y la función de dispersión $h(k)= \lfloor k^2/10 \rfloor \bmod 5$. Mostrar la tabla de dispersión cerrada con resolución lineal de colisiones que resulta de insertar (en este orden) 23, 48, 35, 4, 10, en una tabla inicialmente vacía.

2.6. Un compilador necesita almacenar las variables del programa que está compilando. Para cada una de ellas se almacenará el tipo de la variable, el nombre y la posición de memoria donde se almacena. El compilador sólo necesita funciones para guardar información de una variable y recuperar información de una variable. ¿A qué tipo abstracto de datos corresponde esta necesidad? Si disponemos de suficiente memoria, ¿qué estructura de representación será más eficiente en cuanto a tiempo? ¿Por qué?

2.7. ¿Puede tener sentido en algún caso utilizar una tabla de dispersión con un número de cubetas, M, mucho menor que los tamaños de los conjuntos utilizados, por

ejemplo 10 veces menor? ¿En qué situación, y qué beneficios se obtendrían respecto a una representación con listas ordenadas y con listas no ordenadas?

2.8. En una tabla de dispersión abierta hacemos que las listas almacenadas en cada cubeta sean listas ordenadas. ¿Qué mejora se obtendría para las operaciones sobre el tipo de datos? ¿En qué circunstancias la mejora sería más notable?

2.9. Un sistema operativo necesita controlar la memoria libre y la que está siendo usada por algún proceso. ¿Cómo sería la implementación mediante arrays de booleanos y mediante listas de elementos? Discutir sobre la implementación de las operaciones: obtener cantidad de memoria libre, pedir memoria y liberar un bloque de memoria. ¿Cuál sería su complejidad en cada una de las implementaciones?

2.10. En una tabla de dispersión, reservamos de forma fija espacio en cada cubeta para *q* elementos (esto puede verse como una estrategia mixta entre la dispersión abierta y la cerrada). Con esta estructura, ¿se soluciona el problema de la redispersión? Señala alguna ventaja e inconveniente de esta estrategia sobre la dispersión abierta.

2.11. Considerar el método de hashing en una tabla cerrada con M registros enumerados de 0 a M-1, siendo las claves k enteros en el intervalo 0..3M-1. La función de dispersión es h(k) = k mod M, y las colisiones se resuelven con la función de redispersión h(k, i) = (h(k) + i*h_2(k)) mod M (redispersión doble) . Obtener una función h_2 de modo que claves distintas con igual valor de h(k) tengan secuencias de búsqueda distintas (valores de h_2 distintos). ¿En qué casos se recorre toda la tabla con la función obtenida?

2.12. (TG 3.2) La reestructuración de una tabla de dispersión consiste en sustituirla por otra con más cubetas cuando la tabla está muy llena. Por ejemplo, suponiendo dispersión cerrada, si partimos de una tabla T_1 de tamaño M_1, la nueva tabla es T_2 de tamaño M_2, siendo M_2>M_1. Deberíamos meter todos los elementos de T_1 en T_2, con una nueva función de dispersión que dé valores en el intervalo 0..M_2-1. ¿Cuál sería el orden de complejidad del algoritmo de reestructuración en un caso promedio?

2.13. (TG pág. 97) Supongamos que se utiliza una técnica de dispersión cerrada con tamaño M y redispersión lineal, en un sistema donde se espera que las inserciones y eliminaciones sean frecuentes. Por este motivo se desea que al eliminar un elemento se reutilice ese espacio, en lugar de marcarlo como "eliminado". De esta forma, en algunos casos puede ser necesario mover a la posición borrada otros elementos que hayan producido colisión. Proponer un esquema de algoritmo para la eliminación de un elemento que se encuentra en la tabla. Tener en cuenta que en la redispersión lineal los elementos sinónimos se colocan en posiciones consecutivas (h(k), h(k)+1, h(k)+2, ...) y que en general puede ser necesario mover más de un elemento.

2.14. Para almacenar las variables a, b, c, d y e, se utiliza un método de hashing, en una tabla con 5 posiciones numeradas de 0 a 4, y siendo la función de dispersión h_1(k) y la redispersión h(k, n) = (h_1(k) + n*h_2(k)) mod 5 (empezando con n=1, 2, 3, ...), con:

$h_1(a) = h_1(d) = 0$, $h_1(b) = h_1(e) = 1$, $h_1(c) = 2$,
$h_2(a) = 1$, $h_2(b) = h_2(d) = 2$, $h_2(c) = h_2(e) = 3$.

Obtener el número de comparaciones en la inserción y la búsqueda (es decir el número de pasos en las secuencias de búsqueda) si la secuencia de aparición de las claves es:

a c b d e b a b d e d d e c c b a d

Tener en cuenta que la primera vez que aparece una clave es una inserción, y la siguiente vez es una búsqueda. Hacer el ejercicio:

a) utilizando dispersión abierta.
b) utilizando dispersión cerrada.
c) Comparar los resultados. ¿Dónde son más largas las secuencias de búsqueda?

2.15. En una aplicación donde queremos usar dispersión disponemos de relativamente poca memoria. En este caso, ¿qué estrategia de dispersión es más adecuada suponiendo que se conoce a priori el tamaño de los conjuntos que se van a usar? ¿Por qué? ¿Y si no se conoce el tamaño a priori?

2.16. Suponer una tabla de dispersión cerrada con 20 registros, donde las claves están en el intervalo 0..80000. Definir una estrategia de dispersión que use la técnica de plegado (o *folding*) para este caso. Aplicarla a la introducción de los siguientes elementos: 40734, 71263, 01371, 41, 28497, 10101, 02747, 10230, 1, 12890, 50902, 0, 65126, 54879, 3, 11111, 21. Comentar brevemente la bondad de la función diseñada para este ejemplo.

2.17. Considerar una tabla de dispersión cerrada con M=10 cubetas, en la que queremos insertar los elementos: 422, 12, 72, 392, 763, 842, 652. Definir una función de dispersión y una estrategia de redispersión óptimas para este caso (que evite los sinónimos). Mostrar la tabla cerrada resultante tras la inserción de los elementos.

2.18. (TG 3.3) En una tabla de dispersión cerrada, las claves son enteros compuestos por **n** bytes: $K = k_1, k_2, ..., k_n$. Tenemos definidas dos funciones de dispersión y de redispersión:

- $h_1(K) = (k_1 \cdot k_2) \bmod M$ $\qquad$ $h_{i1} = (h_1(K) + i) \bmod M$
- $h_2(K) = \left\lfloor \sum_{i=1}^{n} \pi \cdot 10^{(ki+10i)} \right\rfloor \bmod M$ $\qquad$ $h_{i2} = (h_2(K) + i^2) \bmod M$

Indicar en una tabla comparativa las principales ventajas e inconvenientes de cada una de las anteriores estrategias de dispersión y de redispersión.

2.19. (TG 3.4) Las tablas de dispersión (en sus distintas variantes), son una técnica muy utilizada para tener un rápido acceso directo a los datos según una clave. Pero, ¿qué ocurre si queremos acceder de forma ordenada (por ejemplo, recorrer todos los datos de menor a mayor)? Explica cómo se podría hacer el recorrido secuencial (por orden de clave) en una tabla de dispersión cerrada. ¿Es adecuada la estructura a ese tipo de acceso?

2.20. Un anagrama es una palabra que se obtiene permutando las letras de otra palabra. Supongamos que tenemos un diccionario especial formado por todas las palabras reales de cinco letras, y que queremos organizarlas en grupos, de forma que cada grupo contenga todas aquellas palabras formadas por permutaciones de las mismas cinco letras, es decir, cada palabra de un grupo será un anagrama del resto de palabras de su grupo. Para conseguirlo vamos a emplear una tabla de dispersión. Describir el problema, la función de dispersión que se va a utilizar, la estructura de la tabla empleada, el método de resolución de colisiones y escribir un algoritmo para resolver este problema. ¿Cómo podríamos aplicar el algoritmo propuesto a todas las palabras del diccionario de la RAE?

2.21. En una tabla de dispersión cerrada con M = 10 cubetas, no queremos usar marcas de "eliminado", sino que al eliminar un valor se desplacen los elementos adecuados. La función de dispersión es h(k) = k mod M, y la redispersión es $h_i(k)$ = (k + i) mod M. Mostrar los elementos que se deben desplazar y la tabla resultante, al eliminar 12 y 59 de la siguiente tabla.

0	1	2	3	4	5	6	7	8	9
89	90	102	12	2	45	13	24		59

2.22. (TG 3.5) Para solucionar el problema de los sinónimos en las tablas de dispersión definimos la siguiente estrategia. Aparte de la tabla de dispersión, tenemos una estructura de lista enlazada llamada "lista de desbordamiento". Cuando se produce una colisión, el elemento que colisiona es añadido a la lista de desbordamiento en la última posición (después de comprobar que no se encuentra ya en la misma).

Escribir un algoritmo en pseudocódigo para eliminar un elemento *x* de la estructura de dispersión, sin usar marcas de "eliminado". Suponer que la tabla de dispersión es *T*, la lista de desbordamiento *L* y la función de dispersión *h*; por lo tanto, la llamada tendrá la forma *Eliminar (x, T, L)*.

2.23. En una aplicación que usa dispersión cerrada tenemos M=20 cubetas, y una función de dispersión con la forma h(k) = k mod M. Además, en esta aplicación se conocen a priori los elementos que se van a almacenar, que son: 72, 51, 132, 32, 631, 391, 211. ¿Por qué la función de dispersión elegida no es buena para esta aplicación concreta? ¿Qué pasaría si usáramos redispersión lineal? Definir una buena función de redispersión para este que caso, que evite el problema que aparece.

2.24. (TG 3.6) En cierta aplicación decidimos usar tablas de dispersión para representar conjuntos de números de teléfono con prefijos, como por ejemplo {968121314, 968236677, 968990012, 606172614, 968367722}. Tenemos disponible una librería que implementa la dispersión abierta y la cerrada, y que permite definir todos los parámetros a usar (tipo de dispersión, tamaño de la tabla, tipo de datos almacenado en la tabla, función de redispersión, etc.), excepto la función de dispersión, que es fija, y vale h(k) = (k div 10^6) mod M. Elegir los parámetros configurables de la implementación más adecuados para este caso, de manera que se eviten los problemas de la función de dispersión (que, para esta aplicación, es previsible que produzca muchos sinónimos).