- ¿Cuál de las siguientes funciones Hash te parece apropiada y cuál no para insertar elementos en una tabla Hash abierta/cerrada? ¿Por qué?
  a) h(k) = k mod (MxN), con M, N primos
  b) h(k) = k^2 mod M, con M primo
  c) h(k) = (M- (k mod M)) -1, con M primo

- Insertar las claves {12, 41, 8, 34, 51, 20, 43, 39} en una tabla Hash cerrada de tamaño 11 usando como función Hash h(k) = k % 11 y, para resolver colisiones, rehashing doble usando como función secundaria h(k) = 7 - (k%7). ¿Cuál es el rendimiento?

- Febrero 2013: Supongamos que en una tabla Hash hacemos uso en cada cubeta de un árbol AVL en lugar de listas. ¿Cuál es la eficiencia (peor caso) de la función buscar? Mostrar la tabla Hash resultante de insertar los elementos:

	16	**72**	**826**	**1016**	**12**	**42**	**623**	**22**	**32**
h(k)	6	2	6	6	2	2	3	2	2

- Los empleados de una empresa se representan en una base de datos por su nombre, DNI y número de identificación personal como identificadores únicos, junto con la información adicional del empleado. Construir una estructura de tablas Hash que permita acceder en un tiempo O(1) de media al registro de un empleado por cualquiera de esos 3 campos, teniendo en cuenta que no hay espacio para duplicar registros.

- Un videoclub está desarrollando un aplicación que permita un acceso rápido a las películas de que disponen. No tienen claro qué estructura de datos elegir para almacenar los registros (compuestos por código, título, año,…). La búsqueda se hace por el título de cada película. Para la inserción y el borrado de películas se usa un código único asociado a cada película. Además, se necesita una función para imprimir de forma ordenada el catálogo de películas. Analizar la eficiencia de estas cuatro operaciones si se usa como estructura de datos:
  1. Vector ordenado
  2. Lista ordenada
  3. ABB
  4. AVL
  5. Tabla Hash abierta

En función del resultado, decidir qué estructura de datos resuelve mejor el problema

- Se dice que una pila es inversa a una cola cuando el listado de los elementos de la pila coincide con el listado de los elementos de la cola en orden inverso. Usando las clases stack y queue de la STL, diseñar una función para determinar si una pila y una cola dadas son inversas.

- Usando el TDA list<int>, diseñar una función void agrupar_elemento(list <int> &lista, int k) que agrupe de forma consecutiva todas las apariciones del elemento k en la lista a partir de la primera ocurrencia de éste. Así, p.ej., si lista = {1, 3, 4, 1, 4} y k=1, entonces debería dejar lista como {1, 1, 3, 4, 4}; si lista = {3, 1, 4, 1, 4, 1, 1} y k=1, entonces lista = { 3, 1, 1, 1, 1, 4, 4}

- Usando el TDA list<int>, diseñar una función template <class T> void dividir_lista (list<T> & lista, T k), que agrupe en la primera parte de la lista los elementos menores que k y en la segunda parte los mayores o iguales. Deben usarse iteradores y no se pueden usar estructuras auxiliares. Tampoco se puede modificar el tamaño de la lista y la función ha de ser O(n).

- Dada una lista de enteros con elementos repetidos, diseñar (usando el TDA lista) una función que construya a partir de ella una lista ordenada de listas, de forma que en la lista resultante los elementos iguales se agrupen en la misma sublista. Ejemplo:
  Entrada = {1, 3, 4, 5, 6, 3, 2, 1, 4, 5, 5, 1, 1, 7}
  Salida = {{1, 1, 1, 1}, {2}, {3, 3}, {4, 4}, {5, 5, 5}, {6}, {7}}

- Es posible mantener 2 pilas en un vector (denominaremos a la estructura piladoble) si una crece desde la posición 0 (pila 0) del vector y la otra decrece desde la última posición (pila 1). Dar una especificación (sintaxis y semántica) de ese nuevo TDA. Implementar una función void pop(int indicepila) que aplicada a una piladoble p (p.pop(0) ó p.pop(1)), borre un elemento de la pila referenciada por indicepila (0 ó 1)

- Usando el TDA Lista, construir una función que tenga como entrada dos listas y devuelva 1 si la primera está contenida en la segunda (tiene los mismos elementos, consecutivos y en el mismo orden) y 0 en caso contrario. P.ej.: Si l1=<1,2,3> y l2=<0,1,2,3>, salida=1; Si l1=<1,2,3> y l2=<1,2,2,3>, salida=0.

- TEST ¿Verdadero o falso?
  1. El TDA Cola con prioridad no es más que un caso particular del TDA Cola.
  2. El orden en el que se listan las hojas en los recorridos de preorden, inorden y postorden en un árbol binario es el mismo en los tres casos.
  3. Un AVL puede reconstruirse de forma unívoca dado su recorrido en inorden
  4. Es imposible que un árbol binario sea AVL y APO a la vez
  5. En un esquema de hashing doble nunca puede ocurrir que para dos claves $k1 \neq k2$ ocurra simultaneamente que $h(k1) = h(k2)$ y que $h_0(k1) = h_0(k2)$
  6. Un analista tiene que resolver un problema de tamaño 100 y para ello utiliza un algoritmo $O(n^2)$. Uno de sus colaboradores consigue obtener una solución que es O(n). El analista debe olvidarse de su primera solución y usar la de su colaborador, de mejor eficiencia.
  7. Si la eficiencia de un algoritmo viene dada por la función f(n) = 1 + 2 + ... + n, ese algoritmo es O(max(1,2,...,n)) es decir, O(n)
  8. Puede hacerse esta definición
     stack<int>::iterator p;
  9. Un APO puede reconstruirse de forma unívoca dado su recorrido en postorden.
  10. En un esquema de hashing doble es correcto usar como función hash secundaria la función $h_0(x) = [(B-1) - (x\%B)] \% B$, con B primo

- Sea T un árbol binario con n nodos. Se dice que un nodo es un k-nodo si cumple la condición de que el número de descendientes en su subárbol izquierdo difiere del número de descendientes de su subárbol derecho en, al menos, k. Usando el TDA bintree, diseñar una función que encuentre los 5-nodos de T.

- Matrices "a jirones" (*Ragged Arrays*): Una matriz "a jirones" se puede ver como un vector de filas en el que cada fila es un vector de elementos de un tipo base. A diferencia de los arrays bidimensionales de C++, no es preciso que todas las filas tengan la misma longitud: cada vector fila es independiente de los demás y la única restricción es que todas las filas contengan elementos de un mismo tipo (TDA homogéneo). Plantear una implementación de matrices a jirones usando el contenedor vector de la STL. Obtendremos un TDA menos eficiente, pero más versátil que el array bidimensional clásico.