AED (2024-2) - Homework

1. Escribir un programa que haga el ordenamiento de una cantidad n de números, de tipo T. registrando su tiempo de ejecución. Se puede utilizar para esto, por ejemplo, los algoritmos Mergesort o Quicksort.
2. Modificar el programa del punto 1, para que se pueda ordenar de forma ascendente o descendente, utilizando polimorfismo.
3. Modificar el programa del punto 1, para que se pueda ordenar de forma ascendente o descendente, utilizando function objects.
4. Modificar el programa del punto 3, para que se pueda ordenar de forma ascendente o descendente, utilizando function objects, y funciones inline.
5. Modificar el programa del punto 1, para que se pueda ordenar de forma ascendente o descendente, utilizando punteros a funciones.
6. Aprender a utilizar las clases de STL array<T> y vector<T>
7. Implementar la clase CVector, que es vector de datos, que se redimensiona cuando necesita más espacio, con las operaciones push_front, push_back, pop_front, pop_back, operator[].
8. Aprender a utilizar la clase de STL forward_list<T>
9. Implementar la clase CForwardList, que es una lista simplemente enlazada, con las operaciones push_front, push_back, pop_front, pop_back, operator[].
10. Aprender a utilizar la clase de STL list<T>
11. Implementar la clase CList, que es una lista doblemente enlazada, con las operaciones push_front, push_back, pop_front, pop_back, operator[].
12. Aprender a utilizar la clase de STL deque<T>
13. Implementar la clase CDeque, tomando como ejemplo la clase deque<T> de STL, con las operaciones push_front, push_back, pop_front, pop_back, operator[].
14. Aprender a utilizar las clases de STL stack<T>
15. Implementar la clase CStack, utilizando el concepto de adaptors, con las operaciones push, pop, top.
16. Aprender a utilizar las clases de STL queue<T>
17. Implementar la clase CQueue, con las operaciones push, pop, top.
18. Implementar la clase CSortedList, que es una clase simplemente enlazada, ordenada, y que no admite elementos repetidos.
19. Implementar un árbol binario de búsqueda, con las operaciones find, ins y rem.
20. Implementar, en un árbol binario de búsqueda, el recorrido inorder, preorder y postorder utilizando la recursividad de c++.
21. Implementar, en un árbol binario de búsqueda, el recorrido inorder, preorder y postorder utilizando un std::stack.
22. Implementar, en un árbol binario de búsqueda,el recorrido por niveles en un árbol binario de búsqueda utilizando std::queue.
23. Utilizar la implementación de un árbol binario del punto 19. Agregar las funciones necesarias para crear un árbol binario de búsqueda balanceado, con la técnica AVL.

24. Aprender a utilizar la clase std::priority_queue .
25. Implementar una clase CHeap<T,Ord> donde T es el tipo de dato y Ord es un function object que determina el orden del heap. Implementar esta idea utilizando nodos y punteros.
26. Implementar una clase CHeap<T,Ord> donde T es el tipo de dato y Ord es un function object que determina el orden del heap. Implementar esta idea utilizando como base la clase std::vector<T>
27. Aprender a utilizar las clases de STL std::thread y std::mutex
    Ejemplos de threads vistos en clases => link
28. Implementar un algoritmo de ordenamiento usando std::threads. Durante el ordenamiento, todos los procesadores del computador usado, deben estar al 100%.
29. Implementar la clase CSparseMatrix<T>, utilizando dos arrays de punteros a nodos, para filas y para columnas. La clase debe tener las funciones findx, findy, set, get, operator().
30. Implementar la clase CBitVector, que es un array de bits utilizando internamente un array de tipo char. La clase debe hacer todas sus operaciones mediante el operator[]
31. Implementar una clase CGraph<N,E>, utilizando la idea de listas de adyacencia, donde N es el tipo de dato que contiene el nodo y E es el tipo de dato que contiene la arista.
32. Implementar el Algoritmo de Dijkstra sobre la implementación del punto 31, considerando que el tipo N es el tipo char
33. Implementar el Algoritmo de Dijkstra sobre la implementación del punto 31, considerando que el tipo N es una coordenada con valores x,y. Aquí se debe considerar la heurística de la distancia euclidiana.
34. Implementar el Algoritmo A* sobre la implementación del punto anterior
35. Implementar la idea de precalcular todos los posibles caminos de un grafo y responder a una búsqueda del camino entre dos nodos en tiempo constante.