1. Bir fonksiyonu hem öz yineli hem de iteratif yazdıysak ve çalışma zamanları aynıysa hangisini tercih ederiz? - Çalışma zamanları aynı ama öz yineli olanda bellek gereksinimi daha fazla, dolayısıyla iteratifi seçeriz.

2. Heapify() fonksiyonunu yazın.

3. Tek yönlü bağıl listeyi tersine çeviren fonksiyonu yazın.

4. Bir fabrikada çalışan personellerin sicil no ve adlarının n elemanlı bir dizide yer aldığını varsay. Bu diziyi personellerin adına göre alfabetik olarak sıralayacak seçmeli (selection) sıralama fonksiyonunu yaz.

5. [Rastgele 10 adet sayı] Verilen sayıları quick sort ile adımları çizerek sırala. (Kod yok, çizimle cevaplanacak)

6. Write a c function that finds and returns the minimum value stored in the nodes of a binary search tree. What is the running time of your function in big-o-notation, assuming that the binary search tree is also an AVL tree? Explain your answer.

7. To compress the string “ABABAAACABACDA” by using Huffman Coding tree, draw the coding and generate binary codes for each character given in the string.

8. Write a c function that removes all duplicates in an array A of N items. Return the number of items remain in A. Your function must run in O(Nlog2N) average time. Show running time and memory requirement of your algorithm in big-o-notation. (Hint: Use quick sort as a preprocessing step, and you may employ an auxiliary array for removing duplicates).

9. Write a new search algorithm that is used over sorted arrays and very similar to the binary search. Your new search algorithm is called as “quarterly search” which takes a sorted integer array A, integer key to be searched incites of the first and last elements of the array parameters. If key is found, it returns the array index of the key, otherwise returns -1. Quarterly search is very similar to binary search except that it divides the array to be searched four approximately equal sized parts and it continues the search process with one of these parts. In binary search, on the other hand, array is divided into two approximately equal sized parts. Write a C function “int QuarterlySearch(int A[], int key, int left, int right)” which makes quarterly search and give running time of your function big-o-notation.

10. Write a C function which searches a given key from a hash table by using linear probing collision resolution method. You can assume that key is an integer value and hash is an array of integers having TableSize elements and key mod. TableSize is used as a hash function.

11. Programs A and B are analyzed and found that the worst case running time for A is 1000NlogN and B it is N squared.
    a) Which program should be preferred for small N values when N smaller or equal than 1000. Explain.
    b) Which program should be preferred for large N values when N bigger or equal than 1000. Explain.

12. Let array A has the following elements A = [5,2,1,7,9,3,10,6,4]. Trace trough the steps of insertion sort will use an array A to sort the array. Write elements in the array for each step.

13. Given the input {111,123,633,499,999,199} a fixed table size of 10 and a hash function "h = x mod 10". Show the resulting
    a) quadratic probing hash table
    b) separate chaining hash table

14. Write a C function that counts and returns the number of rodes in circularly linked list. You should assume that circularly linked list is defined by using pointers and it is directional.

15. Let S be a stack of integers and x be an integer value. Use push, reset, size functions; and implement a new function "int delete(int x)" in C such that this function deletes all occurrences of x from S, leaving the other elements of S int he same order and returns the number of elements deleted. (Hint: use a second stack, you can either use array or linked list implementation. Write only delete function. Code for push, pop, reset and size functions are not required.)

16. For expression (((3+5)*8)^2)/((4-(a*x))+7) draw the expression tree, write the expression in prefix and postfix notations.

17. Bu soruda sınıfta öğrendiğiniz kuyruk yapısından biraz farklı bir kuyruk yapısı tasarlanmak istenmektedir. Bu yeni kuyruk yapısının her elemanında bir tamsayı değer yer almaktadır. Eğer eklenmek istenen tamsayı sıfırdan küçükse kuyruğun başına, sıfırdan büyük ya da eşitse kuyruğun sonuna eklenmektedir. Kuyruktan alma işlemi ise her zaman kuyruğun başından yapılmaktadır.
    a) Buna göre bu yeni kuyruk yapısı için dizi mi yoksa bağıl liste mi kullanmak daha avantajlıdır? Sebebini açıklayınız.
    b) Bu yeni kuyruk yapısı için ekleme fonksiyonunu "int ekle(int sayi)" şeklinde C dilinde yazınız.

18. Her düğümünde bir pozitif tamsayı değerin saklandığı herhangi bir ikili ağacın düğümlerindeki en büyük değeri bulup döndüren fonksiyonu "int bulEnBuyuk(AGAC2 * agacKok, int enBuyuk)" olarak C dilinde yazınız. Fonksiyon parametre olarak gönderilen

enBuyuk değerinin başlangıçta 0 olduğunu varsayınız. Yazdığınız fonksiyonun çalışma zamanı büyük O notasyonunda nedir?

a) Eğer bu ağaç ikili arama ağacı olsaydı cevabınız nasıl değişirdi? Her düğümünde bir pozitif tamsayı değerinin saklandığı ikili arama ağacındaki en büyük değeri bulup döndüren fonksiyonu "int bulEnBuyuk(AGAC2 *agacKok)" olarak C dilinde yazınız. Yazdığınız fonksiyonun çalışma zamanı büyük O notasyonunda nedir? Açıklayınız.

19. Herhangi bir ikili arama ağacının düğümlerinde yer alan en küçük değeri bulup döndüren fonksiyonu C dilinde yazınız. İkili arama ağacının aynı zamanda bir AVL ağacı olduğunu varsayarsanız, yazdığınız fonksiyonun çalışma zamanı büyük-O notasyonunda ne olur? Açıklayınız.

20. Write a C function "float avg(SIMPLE_LIST* first)" which takes a single directional linked list as a parameter and returns the average of the values stored in the linked list. You can assume that the linked list is defined by using pointers and each node of the linked list stores an integer value.

21. Write a C function "int enqueu(int item)" to insert an integer item to priority queue using a min heap where the root has the minimum value and each node is less than or equal to its children. The enqueuer algorithm should work as follows:
    - Insert the new element at the new element.
    - Call heapify() for the parent of the new element
    - Continue to call heapify() for each node until the root node.

    You can use heap functions heapify(), left(), right(), parent() without writing their source codes.

22. Write a C function "int compute(char* expression)" which take a postfix expression as a parameter, then computes this expression by using a stack. For example if the postfix expression is 12+3*52/- your function should return 7. You can assume that the postfix expression can only have +,-,*,/ operations. There is a single blank character between each values in the postfix expression and number are only digit integers. You only write the function "compute", you can directly use stock functions without their codes.

23. Write the name and running time of the search method that should be used for each of the below case.

    a) search for a value from a sorted array

    b) search for a value from a sorted linked list

    c) search for a value from an unsorted array

    d) search for a value from an unsorted linked list

24. Answer the following questions for a hash table which uses separate chaining for collision resolution:

a) define an array to store this hash table. You can assume that only integer items will be sorted int he hash table.

b) how can you determine the size of array? Explain briefly.

c) Can you define this hash table without using any array (i.e. by using only pointer-based a linked lists)? Explain briefly.

25. Insert the following values 9,2,1,19,15,25,5,3,7,30 into a binary search tree which is initially empty. Draw the resulting binary search tree. Then delete 25, after that delete 9. Draw the binary search tree after each deletion operation.

26. Insert the records with the keys 5,15,1,7,3,22,29,34 and 9 into a hash table using extendible hashing. Use h(key)= key as the hash function. Assume that the global depth is equal to 1 and the hash table is empty at the beginning. Each data ... may hold up to 3 records. Show each step of the insertion.