VERİ YAPILARI DERSİ TEST SORULARIbiz

1. **Bir problemin çözümünde kullanılan komutlar dizisine ne ad verilir?**

a)Veri(data)

b)Array

c)Char

d)Algoritma +2

e)Structure(yapı)

2. **Aşağıda ağaçla ilgili verilen bilgilerden hangisi yanlıştır?**

a)Ağaçlar hiyerarşik ilişkileri göstermek için kullanılır.

b)Her ağaç node'ler ve kenarlardan (edge) oluşur.

c)Herbir node düğüm bir nesneyi gösterir.

d)Herbir kenar (bağlantı) iki node arasındaki bağlantıyı gösterir.

e)Arama işlemine bağlı dizelere göre çok yavaş yapılır. +1(It is done very slowly compared to strings connected to the search process.)

3. **Aşağıdakilerden hangisi veri yapıların temel veri tiplerinden değildir?**

a)Char(8 bit) b)İnt c)float d)Boolean **e)string +4**

4. [1]**Aşağıdakilerden hangisi ağacın temel kavramlarından değildir?**

a)Düğüm b)Kök **c)Kodlama Ağacı(coding tree)** d)Aile e)Yaprak

5. **Aşağıda verilen bilgilerden hangisi yanlıştır?**

a)Ağacın her bir elemanına düğüm adı verilir.

b)Kök en üst seviyedeki tek düğümdür.

c)Herhangi bir çocuğu bulunmayan düğümlere yaprak denir.

d)Veri yapıları statik ve dinamik olarak 2 ye ayrılır. **(kuyruk, yığın ve ağaçlar; birer veri yapısı değil mi?)**

e)Veri tiplerinde tam sayı ve karakter kullanılmaz. +2(Data types do not use integers and characters.

6. **Veri tabanı ile ilgili verilen bilgilerden hangisi doğrudur?**

a)Dinamik ve statik olarak 2 ye ayrılır

b)Kesin çözüme götürür.

c)İşlenecek ve işlenmiş oluşan bilgi bankasıdır.

d)Kolayca çözülebilen temel bir durumdur

e)Herbir node'lar ve kenarlardan oluşmaz.

7. **Aşağıdakilerden hangisi ağaç türlerinden değildir?**

1

a)Derinlik Ağacı(depth tree)

b)Kodlama Ağacı

c)Sözcük Ağacı

d)Kümeleme Ağacı

e)İkili arama Ağacı

8. **Aşağıdakilerden veri yapılarının basit (basic) ilkelerinden değildir?**

a)String

b)Array

c)Structure(yapı)

d)Union

e)Float +2

9. **Veri yapıları aşağıdakilerden hangisini barındırmaz ?**

a)Bağlı listeler +1(linked lists)

b)Yığıt ve kuyruk

c)İkili ağaçlar

d)Dizilerek

e)Veri Tipleri

10. **Aşağıdakilerden hangisi tam sayı formatlarından değildir?**

a)Doğal İkili Kod

b)1'e Tümleyen

c) 3'e Tümleyen

d)2'ye Tümleyen

e)8’e Tümleyen

11.Aşağıdakilerden hangisi ağaç türlerinden biri değildir?

a)İkili Arama
b)Veri Tabanı+1(database)
c)Kodlama
d)Sözlük
e)Kümeleme

12.Aşağıdakilerden hangisi stack fonksiyonlarından biri değildir?

a)Push
b)Reset
c)Top
d)Data +1
e)Pop

13.Genellikle dizi indisi hangi karakterler arasına yazılır ?

a)( ) ? +1

b)[ ]

c){ }

d){ ( ) }

e){--}

14.Hangisini dizi olarak tanımlayamayız ?

a)Yapı

b)Metin

c)Karakter

d)Döngü+1(loop)

e)Sayısal veri

15.Hangisi bir veri listesi değildir?

a)Bağlı

b)Çift Bağlı

c)Grafiksel +1(Grafically)

d)Yığın yığın

e)Doğrusal

16.Aşağıdakilerden hangisi algoritmalar tarafından işlenen en temel elemanlardan biri değildir?

a)Sayısal b)Metinsel c)Ses d)Return+1 e)Resim

17.Aşağıdaki açıklamalardan hangisi yanlıştır ?

a) Düğüm (node) : Ağacın her bir elemanına düğüm adı verilir.
b) Kök (root) : En üst seviyedeki tek düğümdür.
c) Çocuk (child) : Bir düğümün sol ve sağ bağı aracılığı ile bağlandığı düğümler o düğümün çocuklarıdır.
d) Aile (parent) : Bir düğüm, sağ ve sol bağları ile bağlandığı düğümlerin ailesidir.

e) Yaprak (leaf) : Bir düğümün kök düğümden olan uzaklığıdır.+1Leaf (leaf): The distance of a node from the root node.

18.Hangisi ağaç türlerinden değildir?

a)İkili Arama Ağacı (Binary Search Tree)

b) Kodlama Ağacı (Coding Tree)+1

c)Sözlük Ağacı(Dictionary Tree)

d)Kümeleme Ağacı (Heap Tree)

e)Toplama ağacı (Topstatin Tree)

19.Nesne ekleme ve çıkarmaların en üstten(top) yapıldığı veri yapısına ne ad verilir?

a)ağaç

b) infix

c)dizi(array)

d)aile

e)yığıt(stack)

20.LIFO(Last İn First Out) liste mantığı yapısı nesne ekleme ve çıkarmaların olduğu hangi veri yapısında kullanılır?

a)düğüm b) ağaç **c)stack(yığın)**
d)etkili arama e)sondan başlayarak dolaşma

21.Bir yığın dinamik olarak hangi biçimde tanımlanır?

a)dizi b)ikili ağaç c)kodlama ağacı d)bağlı liste+1(linked list)? e)sıralı dolaşma

22.Aşağıdakilerden hangisi yığın mantığının bilgisayarlarda kullanım şekli değildir?

a)Rekürsif olarak tanımlanan bir fonksiyon çalışırken hafıza kullanımı bu yöntem ile ele alınır.

b)(,,,*,+,-,) ayraçlarının C/C++ derleyicisinin kontrollerinde.

c)postfix→infix dönüştürmelerinde.

d)Web browserlarındaki back butonu(önceki sayfaya) uygulamasında.

e) kod hatası düzeltmede.

23.Hangisi ağaç yapısının avantajlarından değildir?

a)ikili arama ağacı

b)esneklik

c)etkili arama

d)doğal temsil

e)Uygulamayı daha kolay uygulanabilir hale dönüştürmek

24.Aşağıdaki verilen temel kavramların açıklaması yanlıştır?

a) Kök: En üst seviyedeki tek düğümdür.

b)Düğüm: Ağacın her bir elemanına düğüm adı verilir.

c) Yaprak: Bir düğüm sağ ve sol bağlarına ile bağladığına yaprak denir? +1

d)Kardeş: Aynı aileye sahip düğümlerdir.

25.Asağıdaki verilen yığıtlara verilen örneklerdendir?

a)El feneri b)Sıralar c)Bozuk para+1 d)Ayakkabılık e)Sıra

26.Aşağıdaki bilgilerden hangisi doğru verilmiştir?

a)Özyineleme: Yinelge (özyineleme), en genel anlamıyla bir yapının (kendi kendine) yinelenmesidi.

b)Kuyruk : İlk giren, ilk çıkar (FIFO - First In First Out) özelliğine sahiptir+2

The operations of a queue make it a first-in-first-out (FIFO) data structure.

c)Yığıt : İlk giren, ilk çıkar (FIFO - First In First Out) özelliğine sahiptir

d)Rekürsif : Bir çözümü gerçekleştirmek için yarar.

e)Ağac :Bir kök işaretçisi sonlu sayıda düğümleri ve onları birbirine bağlayandır. ++++++bu!

27.İkili Ağaç Sistemi aşağıdakilerden hangisi ile eş anlamlıdır?

Child Binary Tree+1

Two Tree Parent d)Third Tree

28.Ağaç veri yapısında çocuk ne anlama gelir?

A)Ağacın her bir elemanını birer çocuktur.

B)Bir düğümün bir alt düğümünde olan bağlar çocuklarıdır.+1

C)Bir düğümün sol ve sağ bağı aracılığı ile bağlandığı düğümler o düğümün çocuklarıdır.

D)Şemanın en üstündeki düğüm çocuktur.

“Programlamayla ilgili tüm projeler, projenin amacını belirten şartların tanımlanmasıyla başlar. Gereksinmeler, programcıya verilen girdiler ve bu girdiler sonucu üretilmesi gereken çıktıların ne olması gerektiği sorusuyla tanımlanır.”

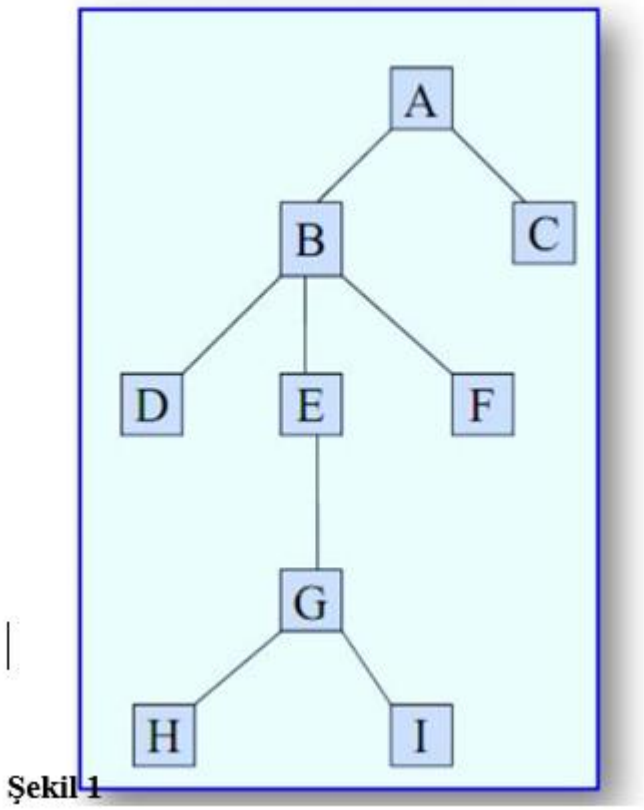
29.Yukarıdaki metinde aşağıdakilerden hangisi tanımlanmaktadır?

A) Design

B)Analysis

C)Root

D)Requirements



Şekil 1

Aşağıdaki 10 soruyu yukarıdaki şekil1'e göre yapınız.

30.Ağacın düğüm sayısı kaçtır?

a)9 b)10 c)11 d)12 e)8

31.Ağacın Yüksekliği kaçtır?

a)3 b)4 c)5 d)6 e)7

32.Kök düğüm hangisidir?

a)A b)B c)C d)D e)E

33.Aşağıdakilerden hangisi yapraklardan biri değildir?

a)C b)D c)I d)E e)H

34.Ağacın düzey sayısı kaçtır?

a)3 b)4 c)5 d)6 e)7

35.H'nin ataları hangileridir?

a)ABC

b)EBA

c)DEF

d)FAE

e)ABF

36.B'nin torunları hangileridir?

a)GHI

b)ABC

c)BCD

d)GHE

e)GHA

37.E'nin kardeşleri hangileridir?

a)AF

b)BF

c)CF

d)DF

e)EF

38.Sağ alt ağaç hangisidir?

a)H b)I c)F d)C e)Yok

39.Sol alt ağaç hangisidir?

a)A b)B c)C d)D e)E

40.Kuyruk ekleme işlemine ne ad verilmektedir?

a)Enqueue +1 b)Dequeue c)Add d)Delete e)Update

41.“Verilerin düzenlenme biçimini belirleyen yapı taşlarıdır.” cümlesi aşağıdakilerden hangisinin tanımıdır?

a)Algoritma +1 b)Veri c)Veri Yapılar d)Karakter e)Dizi

42.Bir problemi çözmek için, bir alt programın kendi kendini çağırmasını sağlayarak, tekrarlı işlerin çözülmesine ne ad verilir?

a)Özyineleme+1 b)Kuyruk

c)Yığıt d)Ağaç e)Dizi

43.Aşağıdakilerden hangisi özyinemeli fonksiyona örnek olarak verilebilir?

a)Faktöriyel+1

b)Ekrana “hello world” yazdırma

c)2 ve 3’ün toplanması

d)Günler dizisi oluşturma

e)2 sayının farkını hesaplama

44.Bir ağaç yapısında en üst seviyedeki düğüme ne ad verilir?

a)Parent b)Leaf c)Node d)Root +1 e)Path

45.Bir düğümün aşağıya doğru üzerinden geçilmesi gereken düğümlerin listesine ne ad verilir?

a)Parent b)Leaf c)Node d)Root e)Path+1

46.Herhangi bir çocuğu bulunmayan düğüme ne ad verilir?

a)Parent b)Leaf+1 c)Node d)Root e)Path

47.Ağacın her bir alanına verilen isim .....dur.

a)Parent b)Leaf c)Node +1 d)Root e)Path

48.Aşağıdakilerden hangisi ikili ağaç üzerinde dolaşma çeşitlerinden biri değildir?

a)Kökten başlayarak dolaşma

b)Sıralı dolaşma

c)Sondan başlayarak dolaşma

d)Rasgele dolaşma

49.Kuyruklar hakkında verilen aşağıdaki bilgilerden hangisi yanlıştır?

A) Kuyruklar, eleman eklemelerin baştan ve eleman çıkarmaların sondan yapıldığı veri yapılarıdır.+1

B)Eleman ekleneceği zaman kuyruğun sonuna eklenir.

C)Bir eleman çıkarılacağı zaman kuyrukta bulunan ilk eleman çıkarılır.

D)Gerçek yaşamda da bankalarda kullanım örneği verilebilir.

E)Kuyruklara FIFO listeleri denilmektedir.

GEÇMİŞ YILLARDA ÇIKAN TEST SORULARIDIR.

1.Inorder traversal of binary seach tree will produce

a) unsorted
b) none of the above
c) reverse of input
ç) sorted list

2. Which of the following data structure is linear data structure?

a)Trees b) Graphs **c) Linked List** d)None of above

3. A queue is a…….

a) **FIFO** b) LIFO c)Ordered d)Linear Tree

4. Which data structure is needed to convert infix notatiton to postfix notation?

a)Branch b)Queue c)Tree **d)STACK**

5. A full binary tree with n leaves contains

a)n^2 **b)2n-1** c)longn nodes d)nlogn nodes

6.Stack is work on the principal of

a) FIFO b) **LIFO** c)ZORO d) LILO

7. Under which condition circular queue is FULL is ent is not used

a) Front=-1 **b) Front=(rear+1)%maxsize** c)Front=(front+1)%maxsize

d) Rear=(rear+1)%maxsize

8. Which is not application of stack ?

a) Reversal of string b) Evaluation of arithmetic operation

c) **Real operating system** d) Recursion

9.Any node is the path from the root to the node is called

a) Successor node **b)Ancestor Node** c)Internal node d)None of

10. What is the result of the following operation? TOP(Push(S,X))

a) **X** b) NULL c) S d) None of above

11. Which of the following operations is performed more efficiently by doubly linked list than by singly linked list?

a) **Deleting a node whose location in given**
b) Search of an unsorted list for a given item
c) Inserting a new node to the front of the list
ç) Traversing the list in the forward direction

12. What is the output of the following code?

```
int value[5]={7,9,11,13,15};

stack s;

initialize(&s);

for (int i=0;i<5;i++)

push  (&s,values[i]);

int n=30;

for (int i=0;i<3;i++)
```

n+=pop(&s);

for (int i=0;i<2;i++)

n=pop(&s);

printf("%d",n);

a)30 **b)53** c)29 d)none of above

13. The POSTFIX form of (A+B)*(C*D-E)*F/G is

a)AB+CD*E-FG/** b)AB+CD*E-F**G/ **c)AB+CD*E-*F*G/**

14. Which one of the following is not in the queue data structure?

a)Front **b)TOP** c)Rear d)Counter

15. Where is the smallest element a binary tree?

a)Root **b)Anywhere** c)Leftmost d)Righmost

16. An AVL tree is

a)a balanced binary tree

b)a balanced binary search tree

c)an unbalanced tree

d)a tree with at most 3 children

17. The number of edges from the root to the deepest node is called…….. of the tree.

a)Depth **b)Height** c)Path d)None

18. What is the output of the following function?

```
void function(struct node *head){
if(head==NULL)
        return;
function(head->next);
printf("%d",head->data);
}
```

a) prints the last node value in the list
b) prints all node values in the list in increasing order
c) prints all node values in the list in reverse order
ç) prints all node values in the list from the beginning

19. Let the array [6;7;15;_____22] be a binary min heap. Indices start with

1. **Which number should be in_____.**

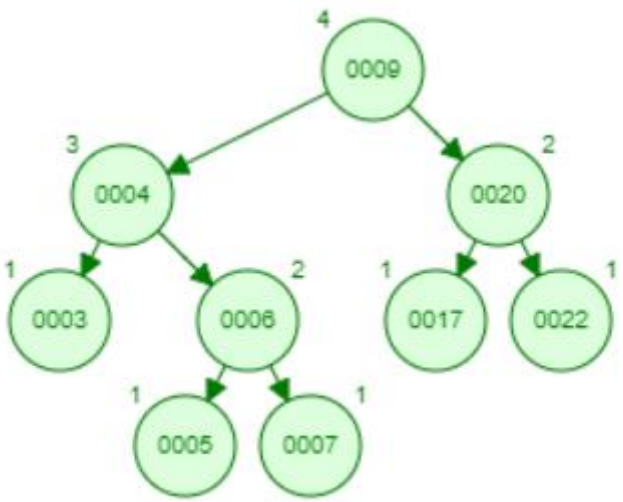
a)8 b)7 **c)15** d) None of above

20. Which of the following is true

A subtree of the root of an AVL tree is always itself an AVL tree.

21.Which of the following is true

Recursive function calls use stack



22. First violation from the bottom if the following tree is AVL

a)6 b)4 **c)17** d) 9None of above

23. What is the smallest and largest number of nodes in a binary heap of height 7?

a)min:256 max:511

b)min:128max:255

c)min:128max:256

d)min:127max255

24. The number of edges from the root to the node is called _____ of the tree.

a) Height

b) Depth

c) Length

d) None of the mentioned

25. What is the time complexity for finding the height of the binary tree?

a) h = O(loglogn)

b) h = O(nlogn)

c) h = O(n)

d) h = O(log n)

26. Which of the following is not an advantage of trees?

a) Hierarchical structure

b) Faster search

c) Router algorithms

d) Undo/Redo operations in a notepad

27. In a full binary tree if there are L leaves, then total number of nodes N are?

a) N = 2L

b) N = L + 1

c) N = L – 1

d) N = 2L – 1

28. Priority-Queue is implemented as a Max-Heap. Initially, it has 5 elements.

The level-order traversal of the heap is given below: 10, 8, 5, 3, 2 Two new elements ”1‘ and ”7‘ are inserted in the heap in that order.

The level-order traversal of the heap after the insertion of the elements is

A)10,8,7,5,3,2,1

B)10,8,7,2,3,1,5

C)10,8,7,1,2,3,5

D)10,8,7,3,2,1,5

29. Which of the following is not a type of data structure?

A) Primitive data structure

B) Simple data structure

C) Linear - nonlinear data structure

D) Complicated data structure

30. What is an AVL tree?

a) a tree which is balanced and is a height balanced tree

b) a tree which is unbalanced and is a height balanced tree

c) a tree with three children

d) a tree with at most 3 children

31. Why we need to a binary tree which is height balanced?

a) to avoid formation of skew trees

b) to save memory

c) to attain faster memory access

d) to simplify storing

32. Which of the following is the basic data structures?

A) Union

B) Stack

C) Tree

D) Integer

33.Percolate up and down are used for

A) AVL trees

B) B-trees

C) Circular queue

D) Binary heaps

34.Which of the following data structure is used by recursive functions ?

A) Queue

B) Graph

C) Stack

D) Boolean

35.Which of the following can not be done with linked lists?

A) Binary heaps

B) Stacks

C) Queues

D) Trees

36.A node with key 8 has a left child with key 10. Which of the following objects could this node be found in?

A) Binary Search Tree

B) Max Heap

C) Min Heap

D) Two of the above

37.Where is the smallest element a binary search tree?

A) Root

B) Leftmost

C) Rightmost

D) It can be anywhere

38. Which one of the following is not in the queue data structure?

A) Front

B) Top

C) Rear

D) Counter

39. Why keep the first element index in the queues?

A) Because the elements are removed from the first index

B) For fun

C) To use memory more efficiently

D) Because the first element is so important

40.Which one of the following do not have right and left child concept?

(Aşağıdakilerden hangisinin sağ ve sol çocuk kavramı yoktur?)

A) BST

B) Double Linked List

C) Binary Heap

D) AVL Tree

41. Which of the following data structure is linear type?

A) Strings

B) Lists

C) Queues

D) All of above

42. The condition Top=-1 indicates that

a) Stack is empty

b) Stack is full

c) Stack has only one element

d) None of these

43. Which ones right?

i.)Recursive function calls use queue

ii)A tree is a graph that may have cycle

iii)The length of a path is the number of edges on that path

iv)Every node in a tree has at most 2 children

A)i and ii B) only ii **C)only iii** D)i,ii,iii

44. 60-70-65-80-72-100-90-82-91-75

Bir binary Heap'imiz olsun.Min heap'e göre " 71" eklersek yeni heap'imiz nasıl olur?

A)60-70-65-80-72-100-90-82-91-75-71

B)60-70-71-65-80-72-100-90-82-91-75

C)71-60-70-65-80-72-100-90-82-91-75

D)60-70-65-80-71-100-90-82-91-75-72

45. What is the smallest and largest number of nodes in a heap of height 8? What is the height of a heap with 600 nodes ?

A-) min: 256 max:511 height=9

B-) min: 255 max:511 height=8

C-) min: 256 max:512 height=9

D-) min: 255 max:512 height=9

E-) min: 256 max:511 height=8

---

46. Ağaçlarda bağlı bir liste olduğuna göre klasik bağlı listelerden farkı nedir?

Ağaçlar, klasik listeler gibi doğrusal yapıda değillerdir.

47. 60-70-65-80-72-100-90-82-91-75 Bir Binary Heap olsun. Aşağıdakilerden hangisi 100'ün parent dır?

A)60 B)91 **C)65** D)80

48. İkili ağaçlarla ilgili olarak hangisi yada hangileri doğrudur?

I.Bir ikili ağaçta, k seviyesindeki maksimum düğüm sayısı 2^k 'dır.

II.Bağlı listelerden farkı nonlinear(dogrusal olmayan) olmasıdır.

III.İki düğüm arasında birden fazla kenar olamaz.

A)II ve III B) Yalnız III **C)I,II ve III** D)I ve II E)Yalnız II

49.Aşağıdakilerden hangisi veya hangileri priority queue lerde kullanılır?

i)Printers

ii)CPU

iii)Recursion

A)Yalnız i B.)Yalnız iii **C)i ve ii** D)i,ii,iii

50. n düğümlü bir BST ağacında maximum yükseklik kaçtır?

A)n B)2n C)n2 **D)n-1**

51. 1 milyon düğümü olan bir AVL ağacında yükseklik kaçtır?

A)10 **B)20** C)100 D)1000 e)999999

Yanıt:B (20) Bu yükseklik değeri tam olarak söylenemez ama bu civardadır denilebilir.

20 den az olamaz. 100 de olamaz (HK)

52.

GEÇMİŞ YILLARDA ÇIKAN TRUE/FALSE SORULAR

1)Linked List stacks and queues are example of dynamic data structure. **TRUE**

2)Creating and maintaining dynamic data structure requires dynamic memory allocation the ability for a program to obtain more memory…….. TRUE

3)A linked list is a linear collection of self referential structure called nodes connected by pointer links hence the term linked list. TRUE

4)Array are dynamic so the length of an array can increase or descrase as necessary…….FALSE

5)Linked list nodes are normally stored contiguously in memory.. FALSE

6)A stack is a linked list that can be acccesed from either end. FALSE

7)Push is used to place elements on the bottom of a stack and pop is used to remove elements from the top o a stack. FALSE

8)Queue nodes are removed only from the front of a queue and are inserted only at the back of queue. TRUE

9)A major advantage of array over linked list is that it takes fewer steps to insert a new element at the end of a long array…. TRUE

10)ADT stands for Abstract Data Tree. FALSE

11)A Stack also known as a FIFO buffer is a first in first our linear data structure. FALSE

12) Bir fonksiyon ya iteratiftir yada özyinelemelidir. TRUE

RECURSIVE FACTORİYEL FUNCTION

1. faktroriyel problemi
2.

```
int fact(int n) {
        if (n == 0)
        return 1;
        else
                return n*fact(n-1);
```

2. girilen n değerine kadar olan sayıların toplamını bulur.

```
int sum (int n) {
if (n==o)
        return 1;
else
        return n+sum(n-1)
```

3. fibonacci problemi

```
        int fibonac( int n) {

        if (n==0)
                return 0;
        else if (n==1)
                return 1;
        else
                return (fibonac(n-1)+fibonac(n-2));
```

```
}
```

TEK BAĞLI DOĞRUSAL LİSTELER

```
struct node {
        int data;
        struct node *next;
}
```

TEK BAĞLI DOĞRUSAL LİSTE OLUŞTUR VE ELEMAN EKLE

```
main() {

        struct node *head;
        head = (struct node *) malloc(sizeof(struct node));
        head->data=-1;
        head->next=NULL;

//listeye eleman ekleme
        head->next=(struct node*)malloc(sizeof(struct node));
        head->next->data=10;
        head->next->next=NULL;
```

TEK BAĞLI DOĞRUSAL LİSTE ELEMANLARINI SAYMAK

```
int count(struct node *head) {
int counter = 0;
while(head != NULL) { // head->next!=NULL koşulu olsaydı son düğüm sayılmazdı
counter++;
head = head -> next;
}
return counter;
}
```

Bu işlemi özyinelemeli yapmak istersek:

```
int count_recursive(struct node *head) {
if (head == NULL)
return 0;
return count_recursive(head->next) + 1;
}
```

TEK BAĞLI DOĞRUSAL LİSTE ARAMA İŞLEMİ

```
struct node* locate(struct node* head, int key) {

struct node* locate = NULL;

while(head != NULL)

if(head -> data != key)

head = head -> next;

else {

locate = head;

break;

}

return(locate);

}
```

Tek Bağlı Doğrusal Listelerde İki Listeyi Birleştirmek

list_1 ve list_2 adındaki iki listeyi birleştirmek için concatenate fonksiyonunu kullanabiliriz.

```
void concatenate(struct node*& list_1, node* list_2) { // parametrelere dikkat

if(list_l == NULL)

list_1 = list_2;

else

last(list_l) -> next = list_2; // last isimli fonksiyona çağrı yapılıyor

}
```

Tek Bağlı Doğrusal Listelerde Verilen Bir Değere Sahip Düğümü Silmek

```
struct node *remove(struct node *head, int key) {
if(head == NULL) {
printf("Listede eleman yok\n");
return;
}struct node *temp = head;
if(head -> data == key) { // ilk düğüm silinecek mi diye kontrol ediliyor.
head = head -> next; // head artık bir sonraki eleman.
free(temp);
}else if(temp -> next == NULL) { // Listede tek düğüm bulunabilir.
printf("Silmek istediginiz veri bulunmamaktadir.\n\n");
return head;
}else {
while(temp -> next -> data != key) {
if(temp -> next -> next == NULL) {
printf("Silmek istediginiz veri bulunmamaktadir.\n\n");
return head;
}temp = temp -> next;
}struct node *temp2 = temp -> next;
temp -> next = temp -> next -> next;
free(temp2);
}return head;
}
```

TERSTEN YAZDIRMAK

```
void print_reverse(struct node *head) {
struct node *head2 = NULL; // yeni listenin başını tutacak adres değişkeni
struct node *temp = head;
while(temp != NULL) {
head2 = addhead(head2, temp -> data);
temp = temp -> next;
}
print(head2);
}
```

GEÇMİŞ YILLARDA ÇIKAN KLASİK SINAV SORULARIDIR.

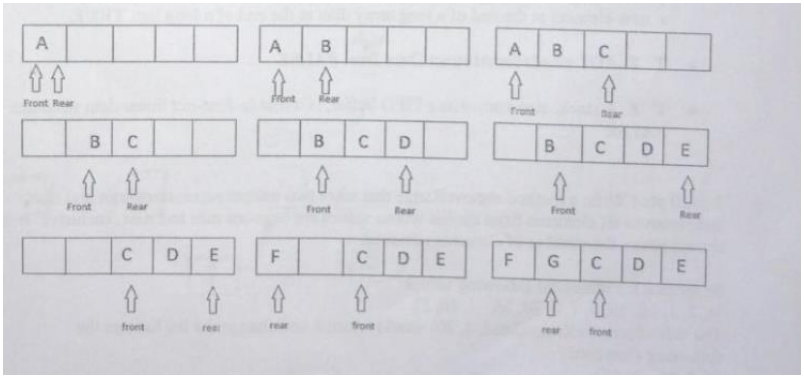
1. Write a method removeRange that takes two integer parameter min and max and removes all elements from the list whose values are between min and max inclusive. It should return the number of elements removed.

2. Write a function that makes a copy of a stack. You may declare and use any local variables. you may also use stack operations such as push,pop,top and isrempty etc.
3. Fibonacci numbers are sequence of numbers in which first two numbers are F0=0 F1=1 and other numbers are calculated as follows: Fn=Fn-1+Fn-2

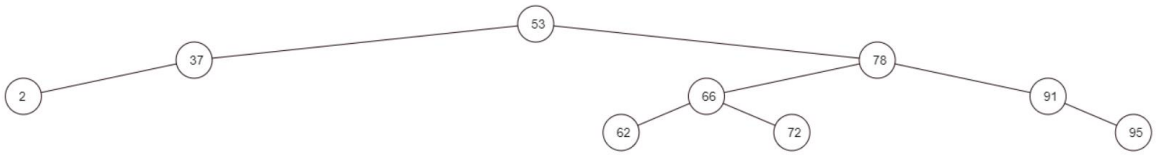
   Write a function which generates the first 10 fibonacci number using a doubly circular linked list.You may assume that the first 2 fibonacci numbers are given in the list as follows:

4. What is data structure?
5. Describe two real life applications of the queue and stack each.
6. What are advantages and disadvantage of a linked list and an array

7. Draw the curcilar queue data structure in array implementions for each step in the following sequence:



AVL ÖRnek:



Inorder: 2 37 53 62 66 72 78 91 95

Preorder: 53 37 2 78 66 62 72 91 95
Postorder: 2 37 62 72 66 95 91 78 53

KLASİK SORULARDAN SEÇMELER

1. Bir min heap i max heap e dönüştüren bir fonksiyon yazın

```
queue *convert(queue *q){
queue *maxheap;
initialize(&maxheap);
while (q->counter!=0){
int data=delete(&q);
insert2(&maxheap,data);
}
return maxheap;
}.
```

```
void insert2(queue *q,int key){
q->counter++;
if(q->counter<=QUEUE_SIZE){
int index=q->counter;
q->A[index]=key;
while(index!=1 && q->A[index/2] < q->A[index])
{
int temp=q->A[index];
q->A[index]=q->A[index/2];
q->A[index/2]=temp;
index=index/2;
}
queue *convert(queue *q){
queue *maxheap;
initialize(maxheap);
while (q->counter!=0){
int data=delete(&q);
insert2(maxheap,data);
}
return maxheap;
```

2. **Bir ikili arama ağacındaki verilerden tek olanları diğer bir BST ağacına kopyalayan copyOdd isimli fonksiyonu yazınız.**

```
struct node *copyOdd(struct node *root, struct node *root2){
if(root != NULL){
if(root -> data % 2 == 1)
root2 = insert(root2, root -> data);
root2 = copyOdd(root -> left, root2);
root2 = copyOdd(root -> right, root2);
}
return root2;
}
```

3. **Tanımlanan bir ikili arama ağacında sadece sol çocuğu olan düğümleri ekrana yazdıran fonksiyonu yazınız?**

```
void leftchild(BTREE *root)
{
if(root != NULL)
{
if(root->left != NULL && root->right == NULL)
```

```
printf(“%d”,root->data);
leftchild(root->left);
leftchild(root->right);
}
else
return;
}
```

4. **Düğüm sayısını bulan fonksiyonu yazınız?**

```
int size(btree *root)
{
if(root==NULL)
return 0;
return 1+size(root->left)+size(root->right);
}
```

5. **Verilen bir q1 kuyruğundaki tek sayi içeren elemanları yeni bir q2 kuyruğuna,çift sayı içeren elemanları yeni bir q3 kuyruğuna ekleyen bir fonksiyon yazınız.**

```
void ekle(queue *q1)
```

```
{
queue q2,q3;
initialize(&q2);
initialize(&q3);
while(!isEmpty(q1))
{
İnt x=dequeue(q1);
İf(x %2==1)
enqueue(&q2,x);
else
enqueue(&q3,x);
}
}
```

6. Fonksiyona parametre olarak gelen sayının tam bölenlerini tanımlanan "q" kuyruğuna sırasıyla ekleyen fonksiyonu yazınız

```
void bölen(int x,queue *q)
{
int i;
for(i=1;i<=x;i++)
{
if (x % i ==0)
```

```
{
enqueue(q,i);
}
}
```

7. Yaprakları kırılmış bir ağaçtım. datalarını toplayan fonksiyon yazınız

```
int sumLeaves(BTREE *root)
{
if(root==NULL)
return 0;
if(root->left ==root->right)
return root->data;
return sumLeaves(root->left) +sumLeaves(root->right);
}
```

7. 4-ary max heap is like a binary max heap, but instead of 2 children, nodes have 4 children. Find mapping.And insert function for max heap

Parent(i) = (i+2)/4 (aşağıya yuvarlama ile),

1.child = 4i-2,

2.child = 4i-1,

3.child = 4i,

4.child = 4i+1

İnt insertMax(int A[], int key){

counter++;

İf(counter<=QUEUE_SIZE){

int index = counter;

A[index] = key;

while(index != 1 && A[(index+2)/4]<A[index]){

swap(A[(index+2)/4],A[index]);

index = (index+2)/4;

}

}

}

8. **Tüm düğümlerin toplamını bulan bir fonksiyon yazınız?**

```
int sumAllNodes(BTREE *root,int sum)
{
İf(root!=NULL)
{
sum+=root->data;
sumAllNodes(root->left,sum);
sumAllNodes(root->right,sum);
}
else
return sum;
}
```

Fonksiyon cagırma: sumAllNodes(root,0);

9. **Tüm sağ çocuklar ile sol çocukların yerlerini birbirleri ile değiştiren fonksiyonu yazınız (inorder sıralandığında büyükten küçüğe doğru sıralanacak şekilde değiştiren fonksiyonu yazınız)**

Yanıt:

```
void mirror (BTREE root){
if(root==NULL)
return;
mirror(root->left);
mirror(root->right);
BTREE temp=root->left;
root->left=root->right;
root->right=temp;
}
```

---

10. boş olmayan bir kuyruk ve S bir yığın. Sadece S ve Q'yu kullanarak kuyruğu ters çeviren fonksiyonu yazınız.

Yanıt:

```
void cevir( )
{
while(!isEmpty(Q)) {
push( S, dequeue(Q));
while(!isEmpty(S)){
enqueue(Q,pop(S));
}
}
}
```

11. Write a function Addmax that adds the largest node value to all nodes (including itself) in a given linked list

```
struct node *addmax(struct node *head){

struct node * temp=head;

int max=head->data;

while(temp!=NULL){

if(temp->data>max)

{

max=temp->data;

}

temp=temp->next;

}

temp=head;

while(temp!=NULL)

{

temp->data+=max;

temp=temp->next;

}

return head;

}
```

12.Recursive Destroy Örnek

```
void destroy(struct Node* head)

{
```

```
    if (head == NULL)
        return;
    destroy(head->next);
    free(head);
}
Node* node = head;
Node* next;
while( node )
{
    next = node->next;
    delete node;
    node = next;
}
}
```

İkili Ağaca Veri Eklemek

```
/* İkili ağaca veri ekleyen fonksiyon */
BTREE *insert(BTREE *root, int data) {
// Fonksiyona gonderilen adresteki ağaca ekleme yapılacak
if(root != NULL) { // ağac boş değilse
if(data < root -> data) // eklenecek veri root'un data'sından kucuk veya eşitse
root -> left = insert(root -> left, data);
```

```
// eklenecek veri root'un data'sından buyukse
else
root -> right = insert(root -> right, data);
}e
lse // eğer ağac boş ise
root = new_node(data);
return root;
}
```

EXAMPLE

Ağaca 13 sayısının eklenecek olduğunu kabul edelim.

Fonksiyon insert(300, 13); kodu ile çağrılacaktır.

```
if(root != NULL) { // 300 adresinde bir ağac
var, yani NULL değil ve koşul doğru
if(data < root -> data) // 13 root'un data'sı
olan 15'ten kucuk ve koşul doğru
root -> left = insert(root -> left, data); //
yürütülecek satır
else
root -> right = insert(root -> right, data);
}
else
```

```
root = new_node(data);

return root;
```

13, kokun değeri olan 15'ten kucuk olduğu icin sol cocuk olan 8 verisinin bulunduğu 200 adresiyle tekrar cağrılıyor.

```
insert(200, 13);

if(root != NULL) { // 200 adresi NULL değil ve koşul doğru

if(data < root -> data) // 13, 8'den kucuk mu, değil ve koşul yanlış

root -> left = insert(root -> left, data);

else // 13, 8'den buyuk mu, evet ve koşul doğru

root -> right = insert(root -> right, data); // yürütülecek satır

}

else

root = new_node(data);

return root;
```

Bu defa 13, 8'den buyuk olduğundan fonksiyon 8'in sağ cocuğu olan 10 verisinin bulunduğu 400 adresiyle cağrılıyor.

```
insert(400, 13);

if(root != NULL) { // 400 adresi NULL değil ve koşul doğru

if(data < root -> data) // 13, 10'dan kucuk mu, değil ve koşul yanlış
```

```
root -> left = insert(root -> left, data);

else // 13, 10'dan buyuk mu, evet ve koşul doğru

root -> right = insert(root -> right, data); // yürütülecek satır

}

else

root = new_node(data);

return root;
```

Şimdi de fonksiyon `10`'un sağ cocuğu ile tekrar cağrılıyor. Fakat `root->right` (*sağ çocuk*) duğumu henuz olmadığı icin adres ile değil `NULL` ile fonksiyon cağrılacaktır.

```
insert(NULL, 13);

else // eğer ağac boş ise yani duğum yok ise

root = new_node(data); // duğum oluşturuluyor ve 13 ekleniyor

return root; // oluşturulan ve veri eklenen duğumun adresi geri donduruluyor
```

`400` adresindeki 10 datasını bulunduran duğumun sağ cocuğu olan right, `NULL` değere sahiptir.Yani herhangi bir duğumu gostermemektedir, cocukları yoktur. Fonksiyonun en sonunda geri dondurulen yeni duğumun adresi, `10`'un sağ cocuğunu

gosterecek olan ve `NULL` değere sahip `right` işaretcisine atanıyor. Sonra fonksiyon kendisinden onceki cağrıldığı

noktaya geri donuyor.

Bir Ağacın Düğümlerinin Sayısını Bulmak

```
/* İkili bir ağacın duğum sayılarını veren
fonksiyon */

int size(BTREE *root) {

if(root == NULL)

return 0;

else

return size(root -> left) + 1 + size(root ->
right);

}
```

EXAMPLE

İkili Arama Ağacından Bir Düğüm Silmek

1- Silinecek olan duğumun cocuğu yok ise,

2- Silinecek olan duğumun sadece bir cocuğu var ise,

3- Silinecek olan duğumun iki cocuğu var ise

1. İkili arama ağacından bir yaprağın silinmesi.

2. İkili arama ağacından 1 cocuğu olan duğumun silinmesi

Bu durumda duğumun çocuğunu ebeveynine bağlar ve istenen duğumu silebiliriz

3. İkili arama ağacından 2 cocuğu olan duğumun silinmesi

Silinecek olan düğüm eğer iki çocuğa sahip ise, silinen düğümün yerine o düğümün sağalt ağacındaki en küçük anahtar değeri getirilir. Daha sonra ise sağalt ağaçtan taşınan anahtar değeri silinir.

ÇALIŞMA SORULARI

1) Klavyden girilen bir cümlede her kelimedeki harflerin yerini rastgele şekilde değiştiren bir program yazınız.

Bir string giriniz; C'de port kontrolu

Çıktı: 'dCe tpro ulkontr

2) Kullanıcının girdiği karakterleri özel karakter, harf veya sayı şeklinde sınıflandıran program yazınız. Program -1 değeri girilene kadar bir döngü yardımıyla karakter sormaya devam etmelidir.

Bir karakter giriniz: r

Harf

Bir karakter giriniz : 3

Sayı

Bir karakter giriniz : @

Özel karakter

3) Klavyeden girilen sayıya kadar olan asal sayıları bulan bir recursive fonksiyon yazınız.

Bir sayı giriniz: 12

2 3 5 7 11

4) Kullanıcının girdiği stringi her satırda satır numarası kadar karakter olacak şekilde yazan programı yazınız.

Bir string giriniz: Computer engineering

1.satır: C

2.satır: om

3.satır: put

4.satır: er e

5) 10 elemanlı bir dizi içerisindeki değerlerden sayı değeri negatif olanları stack içine yerleştiren C kodunu yazınız.

6) Aşağıda struct yapısı verilen tek bağlı doğrusal listenin her bir düğümünde veri olarak tamsayı değerleri tutulmaktadır ve listede birden fazla düğüm bulunmaktadır. Prototipi **float ortalamaBul (struct dugum *)** ; olan öyle bir fonksiyon yazınızki bu fonksiyon listenin başlangıç adresini parametre olarak alsın ve liste içerisinde yer alan sayılardan 15 e tam bölünenlerin ortalamsını geriye döndürsün.

Struct dugum

{ int değer;

```
  struct dugum *sonraki;
};
```

7) (A+B)*C-(D+E)/F

Infıx formunu prefix ve postfix e dönüştürün.

QUESTIONS

Q1) Caesar encryption algorithm is easiest encryption algorithm in the history. In this encryption algorithm, either location of the characters in the string are exchanged or characters are changed to other characters. Traditionally in Caesar encryption algorithm, characters are changed to next 3 characters such as D letter is used for A letter. Another Example in Caesar encryption algorithm, instead of "karabuk" is used "**ndudexn**". According to above explanation, write full program in C programming language converting user entered string to Caesar encrypted string. Notes : ASCII codes between 65 and 90 for A-Z and between 97 and 122 for a-z

(Tarihteki ilk şifreleme tekniklerinden olan **Sezar Şifrelemesi** en basit şifreleme yollarından biridir. Şifrelenmesi istenen metindeki harfler yer değiştirilerek ya da başka harflerle değiştirilerek şifrelenir. Sezar Şifrelemesinde harfler 3 harf sonraki harfle değiştirilerek

şifrelenir.. Örnek olarak Sezar Şifrelemesinde " **karabuk** " yerine " **ndudexn** " kullanılır.

Girilen bir string ifadeyi Sezar Şifrelemesine göre çeviren tam programı c programlama dilinde yazınız.

NOT: A - Z ASCII Kodu 65 - 90 arası a - z ASCII Kodu 97 -122 arasıdır.)(30 P)

Q2) Write a function that expression in the postfix notation comes as parameter convert to infix notation and return address of expression in the infix notation.(Write yourself all function used in this program. Don't use any system function) (Parametre olarak gelen postfix notasyonundaki ifadeyi infix notasyonunda bir ifadeye çeviren ve bu ifadenin adresini döndüren bir fonksiyon yazınız? (Kullanacağınız bütün yapıyı kendiniz oluşturunuz. Hazır fonksiyon kullanmayınız)(35 P)

Q3)

```
typedef enum bool{true=1,false=0}bool;
struct Node{
    char c;
```

```
int main () {
struct Node *head;
…….
```

struct Node *next; struct Node *prior; };	}

According to above structure, you supposed to have doubly linked list that includes nodes. Write a function that takes address of the head as parameter and the function returns true if the sequence is the same from the beginning and from the end otherwise return false. For example, if doubly linked list includes k-a-v-a-k then the function returns true.

(Yukarıda verilen yapıda çift yönlü bir bağlı liste içerisinde elemanlar olduğunu varsayınız. Buna göre başlangıç adresini parametre olarak alan ve baştan ve tersten okunuşu aynı olan kelimeyi tutuyorsa true değilse false değeri döndüren bir fonksiyon yazınız. (Örneğin listede k-a-v-a-k varsa true değeri dönecektir.)(35 P)

ANSWERS

A1)

```
#include <stdio.h>
```

```c
#include <stdlib.h>

#define N 100

int main(void) {

	char word[N];

	char c;

	printf("String giriniz");

	fflush(stdout);

	scanf("%s", word);

	printf("\n");

	int i, boyut = strlen(word);


	for (i = 0; i < boyut; i++) {

	if (word[i] <= 90) {

		c = (word[i] + 3) % 90;

		printf("%c", (c < 65) ? c + 65 : c);

	} else {

		c = (word[i] + 3) % 122;

		printf("%c", (c < 97) ? c + 97 : c);

	}

	}

}
```

A2)

```c
#include <stdio.h>
```

```c
#include <stdlib.h>
#include <string.h>
char * toinfix(char *);
char *word[100];
int yer = 0;
void push(char *);
char *pop();
void push(char *c) {
	int uzunluk = strlen(c);
	word[yer] = malloc((uzunluk + 1) * sizeof(char));
	strcpy(word[yer], c);
	*(word[yer] + (uzunluk + 1)) = '\0';
	yer++;

}

char *pop() {
	if (yer == 0)
	printf("popppps");

	return word[--yer];
}
int main(void) {

	char *s = "ABC*DE/F-G*-H*+";
```

```
        printf("Postfix : %s\n", s);
        toinfix(s);
        int length = strlen(word[0]);
        printf("infix   : %s\n", word[0]);
        return EXIT_SUCCESS;
}
char *toinfix(char *s) {
        int i, j, boyut = strlen(s);
        int temp = 0;
        char *op1, *op2;
        char str[20];
        for (j = 0; j < 20; j++)
        str[j] = '\0';
        for (i = 0; i < boyut; i++) {
        char c = *(s + i);
        char *d = malloc(sizeof(char) * 6);
        d[0] = c;
        d[1] = '\0';
        d[2] = '(';
        d[3] = '\0';
        d[4] = ')';
        d[5] = '\0';
        if ((c == '*') || (c == '/') || (c == '+') || (c == '-')) {                op2 = pop();
                op1 = pop();
                temp = strlen(op2) + strlen(op1);
```

```
            strcat(str, d + 2);
            strcat(str, op1);
            strcat(str, d);
            strcat(str, op2);
            strcat(str, d + 4);
            str[temp + 3] = '\0';
            push(str);
            for (j = 0; j < 20; j++)
            str[j] = '\0';

        } else {
            push(d);
        }
        }
        return word[0];
}
```

A3)

```
bool findNode(node *head){

        node *tail=head;
        bool b=true;
        int i=0,elemanSayisi=0;
```

```
        while(tail->next!=NULL){
        tail=tail->next;
        elemanSayisi++;
        }
        while(i<elemanSayisi/2){
    if((head->karakter) != (tail->karakter)) b=false;
    head=head->next;
    tail=tail->prior;
     i++;
        }
          return b;
}
```

Above code is enough for question 3. Full code for doubly linked list.

```
#include <stdio.h>
#include <stdlib.h>
int id;
typedef enum bool{true=1,false=0} bool;
typedef struct node {

        char karakter;
        struct node *next;
        struct node *prior;
```

```
} node;
void writeLinkedListRecursively(node *);
node *insertNode(char, node *);
void writeLinkedList(node *);
void findlength(node *);
node *deleteNode(int, node *);
bool findNode(node *);

//*************************
bool findNode(node *head){

        node *tail=head;
        bool b=true;
        int i=0,elemanSayisi=0;
        while(tail->next!=NULL){
        tail=tail->next;
        elemanSayisi++;
        }

        while(i<elemanSayisi/2){
    if((head->karakter) != (tail->karakter)) b=false;
    head=head->next;
    tail=tail->prior;
      i++;
        }
```

```
            return b;
}
//*********************************
void findlength(node * head) {
        int length = 0;
        node *ptr = head;

        while (ptr != NULL) {
        ptr = ptr->next;
        length++;
        }
        printf("Length of Linked List is: %d\n", length);

}
node *insertNode(char c, node *head) {
        node *link,*temp=head;
        link = malloc(sizeof(node));
        if (head == NULL) {
        link->karakter = c;
        link->next = NULL;
        link->prior = NULL;
        head = link;
        return head;
        } else {
      while(temp->next!=NULL) temp=temp->next;
```

```
            link->karakter =c;
            link->next = NULL;
            temp->next=link;
        link->prior=temp;
            return head;
            }
}
void writeLinkedList(node *head) {
            node *ptr = head;

            while (ptr != NULL) {
            printf("karakter:%3c ", ptr->karakter);
            ptr = ptr->next;
            }
            printf("\n");
}

int main(void) {
            node *head = NULL;
            int secim, key, data;
            head = insertNode('k', head);
            head = insertNode('a', head);
            head = insertNode('v', head);
            head = insertNode('v', head);
            head = insertNode('k', head);
```

```
        writeLinkedList(head);
  printf("\n sonuc:%s",(findNode(head)==1)?"true":"false");
        return EXIT_SUCCESS;
}
```

Homework : İkili bir ağacın sol cocukları ile sağ cocuklarının yerlerini değiştiren mirror isimli fonksiyonu yazınız.