FINAL DATA STRUCTURES (2020-S1)

QUESTION 1: Choose the most appropriate data structures Queue

I. Adding orders in a restaurant by taking in customers' food requests. Double Linked List

II. Web server cache (it works by allowing users to add a new HTML page or go to a previous one or even going to the next).

III. Tracing a packet traversal through nodes in a network topology. Graph

IV. Counting the frequency of words in a word document (any new word is deemed as frequency of 1) and its frequency is incremented each time it is found. AVL

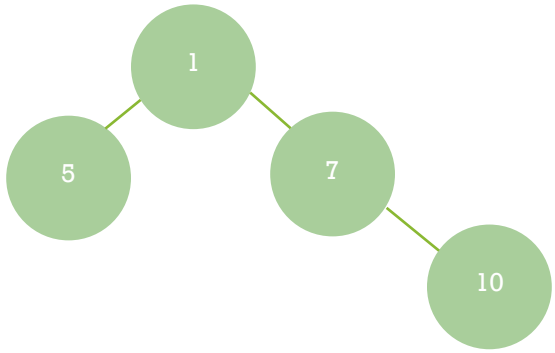
V. Taking patient requests for covid19 vaccines, a patient can be healthy or non-healthy, requests are accepted based on their health history. Priority Queue

VI. The span Si of the stock's price on a given day i is defined as the maximum number of consecutive days just before the given day, for which the price of the stock on the current day is less than or equal to its price on the given day. For example, if an array of 7 days prices is given as {100, 80, 60, 70, 60, 75, 85}, then the span values for corresponding 7 days are {1, 1, 1, 2, 1, 4, 6}. Double Linked List

QUESTION 2: Write a method as a user for splitting a linked list (list) and return a new list that has been split at a given value (n). For example, calling split(linked list<integer> list ,int n) ; list = {1,2,3,4,5,6,7,8,9} and n = 6 returns the list {6,7,8,9}. Also, if the list is empty or the value has not been found display an appropriate message and return an empty list (the original list MUST not change).

```
public static LinkedList <Integer> split(LinkedList<Integer> list, int n)
{
...
}
```

QUESTION 3: Write the recursive BT member method by the following signature, boolean isPathSum (int k), Which is a function that returns true if one of the paths has the sum value of k, if its empty it has no path sum. For example, let bt be the following



so by calling isPathSum(18)
1+7+10 = true
and isPathSum(22) = false

```
public boolean isPathSum(int k)
{
...
}

public boolean rps(BTNode<Integer> t, int k)
{
..
}
```

Write the same method in recursive but now as a user

```
public static boolean isPathSum(BT<Integer> bt, int k)
{
...
}

public static boolean rps(BT<Integer> bt, int k)
{
...
}
```

SOME THEORETICAL QUESTIONS:

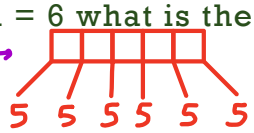
VII. What is the worst-case time complexity of retrieving a max value from a max-heap. O(1)

keys = parents

values/items = leaves

VIII. Given a b+ tree m = 6 what is the maximum number of keys at level 1 (root is at level 0).

level 0 →



level 1 → 5 5 5 5 5 5

(5x6) = 30

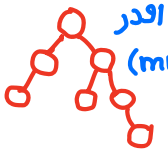
leaves can be 6 (6x6) = 36

IX. In an AVL tree what is the maximum number of nodes at level 2.

If root = 0  0 1 2 4 nodes! If root = 1  1 2 2 nodes!

X. In an AVL tree with 7 nodes what is the maximum depth.

Depth starts at 0, maximum depth = 3 (4 levels)



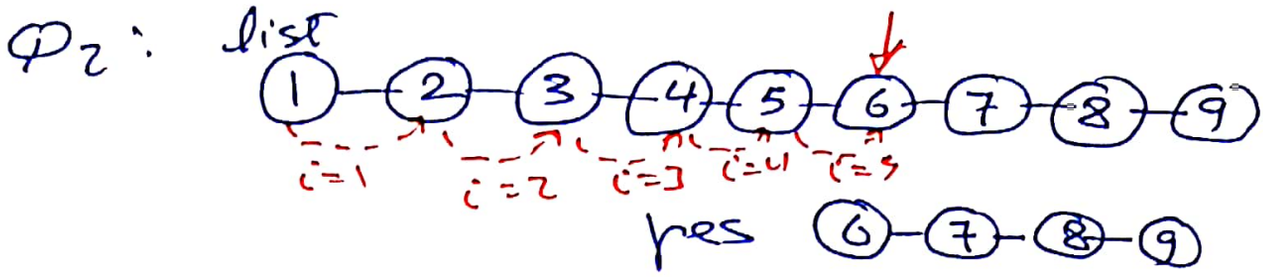
أحاول أطولها قدما اقدر (must be balanced)

XI. What is the best-case complexity for insertion in coalesced chaining or external (I don't remember the type of rehashing) O(1)

XII. DFS uses what data structure. stack

QUESTION 5,6,7,8: AVL, B+, HASH, HEAP AND GRAPHS ARE EXACTLY THE SAME AS THE PREVIOUS EXAMS.

Q2:




```
public List<Integer> split(LinkedList<Integer> list, int n)
{
  List res = new LinkedList<Integer>();
  if(list.empty())
    s.o.p("empty List");
    return res;
  list.findFirst();
  int i=1;
  for( i=1; i< n && !list.last; i++)
    {
      list.findNext();
    }
  if(i==n){
    while(!list.last())
      res.insert(list.retrieve());
      list.findNext();
    res.insert(list.retrieve());
    return res;
  }
  else{
    s.o.p("not Found");
    return res;
  } }
```

i=1 ✓
i=2 ✓
i=3 ✓
i=4 ✓
i=5 ✓

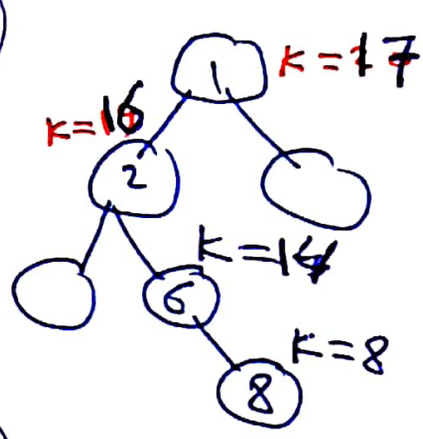
Question 3-a:-

```
public boolean ispathsum(int k){

    return ispathsum(root, k);
}
```

20

20

```
private boolean ispathsum(BTnode<Integer>p, int k)
{
    if(p == null)    return false;
    if(k - p.data == 0) return true;

    return ispathsum(p.left, k-p.data)
        || ispathsum(p.right, k-p.data);
}
```



ممكن يكتب بشكل آخر

```
if(k == p.data)    return true;
```