FACULTY OF SCIENCE

ACADEMY OF COMPUTER SCIENCE AND SOFTWARE ENGINEERING

MODULE	CSC03A3/CSC3A10 COMPUTER SCIENCE 3A
CAMPUS	AUCKLAND PARK CAMPUS (APK)
ASSESSMENT	JULY 2020 **MEMO**

DATE: 2020-07	**SESSION:** 08:00 - 11:00
ASSESOR(S):	PROF D.T. VAN DER HAAR MR R. MALULEKA
EXTERNAL MODERATOR:	PROF J. GELDENHUYS (SUN)
DURATION: 180 MINUTES	**MARKS:** 150

Please read the following instructions carefully:

1. This is a time restricted open book assessment. Answer **all** the questions in a text processor or on paper, which is scanned and submitted.
2. Write *cleanly* and *legibly* on any handwritten parts (if applicable).
3. This paper consists of 26 pages.
4. Ensure that your submission to **Eve** is *complete* and done *before* the cut-off time.

QUESTION 1

(a) Analyze the code below (which computes the sum of the first n entries of an integer array) and answer the questions that follow. (10)

```
1  public int arrSum(int() A, n){
2  int prod;
3  for(int i=0; i<n; i++)
4    prod += A(i)
5  return prod;
6    }
7  }
```

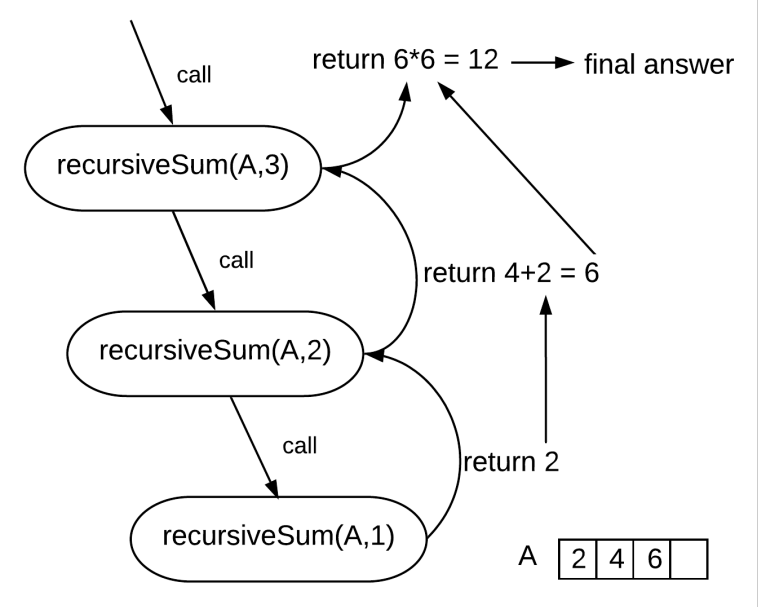
1. Will the given function return the sum of the array entries? Justify your answer.
2. Give a recursive version of the given function.
3. Draw a recursion trace for your new recursive function with $A = \{2, 4, 6\}$ and $n = 3$.

Solution:

1. Yes, int standard default value is 0 (2 mark)
2. 2 marks for base case, 2 marks for recursive case (4 marks)

```
1 public int recursiveSum(int() A, n){
2 if n == 1
3   return A(0);
4 else
5   return A(n) + arrProd(A, n-1);
6 }
```

3. 3 marks for each return value, 1 marks for correct overall graphic



(4 marks)

(b) Using pseudocode, describe how would you go about **adding** an element **after** a given node in a **doubly linked list**? You may use diagrams to support your answer. (5)

Solution:

1 mark each, max 5:

1. Create a new node called storing new element *newNode*
2. Set *newNode*'s next to be the given *node*'s next
3. Set *newNode*'s prev to be the given *node*
4. Update the given *node*'s next node's prev to *newNode*
5. Update the given *node*'s next to the *newNode*
6. Increase the size by 1 (may be excluded)

Total: 15

QUESTION 2

(a) Which kind of growth best characterizes each of these functions? (Redraw the table in your answer sheet, and put an **X** in the appropriate cells) (6)

	Constant	Logarithmic	Exponential	Polynomial
e^n				
2^{5n}				
$(n+5)^3$				
$\log 4$				
$\log n^2$				
183				

Solution:

1. e^n - Exponential
2. 2^{5n} - Exponential
3. $(n+5)^3$ - Polynomial
4. $\log 4$ - Constant
5. $\log n^2$ - Logarithmic
6. 183 - Constant

(b) Consider the following function and, using **primitive counting**, express (5)

the runtime of this function in Big-Oh notation, along with a justification for your answer.

```
1  public int[] prefSum(int[] X, int n){
2    int[] PartSum = new int[n];
3    for(int i = 0; i< n;i++){
4      PartSum[i] = 0;
5      for(int j = 0; j<=i ;j++){
6        PartSum[i] += X[j];
7        }
8      }
9    return PartSum;
10   }
```

Solution:

```
1
2    n+2 // array allocation is n, 1 for variable declaration, 1
         for assignment
3    3n+2
4    2n
5    (3N+2)(n) = 3Nn+2n
6    4(N)
7
8    1
```

$N = 1, 2, ...n - 1$. We take the max i.e. set $N = n - 1$.
Therefore, $T(n) = 8n + 5 + (3n(n-1) + 4(n-1)) = 3n^2 + 9n + 1$.
2 marks for counting primitive ops. 1 mark for $3n^2 + 9n + 1$. The total operation is proportional to $O(n^2)$. (2 marks)

(c) Discuss the **Positional List ADT**, along with the **benefits** and **limitations** of using it. (4)

Solution:

1. Referring to a place in a list without an index.
2. Can use the Position.
3. If we are using a linked list, then we could use the node as the position for an element in the list.
4. Using an index in a linked list means we have to iterate through all the elements, counting as we go (linear time).
5. Having a node means we can perform O(1) insertions and removals, as the node acts as the position of an element in the list.

Total: 15

QUESTION 3

(a) Consider the following List Interface and write a class *Queue* that makes use of the List Interface and the Adapter design pattern to realize a *Queue ADT*. **Note: You do not need to implement the List methods.** (10)

```
1   public interface List<T> {
2     public Node<T> addAfter(Node<T> elem, T item);
3     public Node<T> addFirst(T item);
4     public Node<T> addLast(T item);
5     public T remove(Node<T> elem);
6     public Node<T> search(T elem);
7     public Node<T> first();
8     public boolean isEmpty();
9     public Integer size();
10  }
```

Solution:

Exceptions could be included but are not essential

```
1  //2 marks for formatting
2  public class Queue<T> {
3     private List<T> que; //2 marks
4
5     public Queue() {
6        que = new List<T>(); //1 mark
7     }
8
9     //2 marks
10    public void enqueue(T item) {
11       que.addLast(item);
12    }
13
14    //2 marks
15    public T dequeue() {
16       Node<T> elem = que.first();
17       return que.remove(elem);
18    }
19
20    //1 marks
21    public T front(T item) {
22       return que.first();
23    }
24 }
```

(b) Discuss how a Priority Queue can be used to sort a set of comparable elements. Including the two possible implementations and their performance. (5)

Solution:

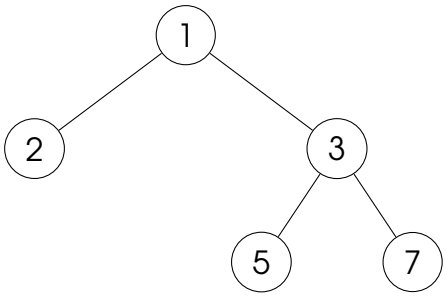
1. Insert unsorted elements in PQ, then remove in order by calling removeMin() repeatedly. (2)

2. A priority queue can be used to sort elements by way of an unsorted list or a sorted list. (1)
3. Unsorted: Insert takes O(1), RemoveMin and min take O(n) time - sorting takes quadratic time (1)
4. Sorted: Insert takes O(n), RemoveMin and min take O(1) time - sorting takes quadratic time (1)

Total: 15

QUESTION 4

(a) Consider the tree below, and answer the questions that follow: (5)



Provide the output if the following traversals are followed:

1. What is the **height** of the tree?
2. What is the **depth** of node with element 2?
3. Is the tree a **proper binary** tree?
4. List the elements in the order of a **inorder traversal** of the tree.

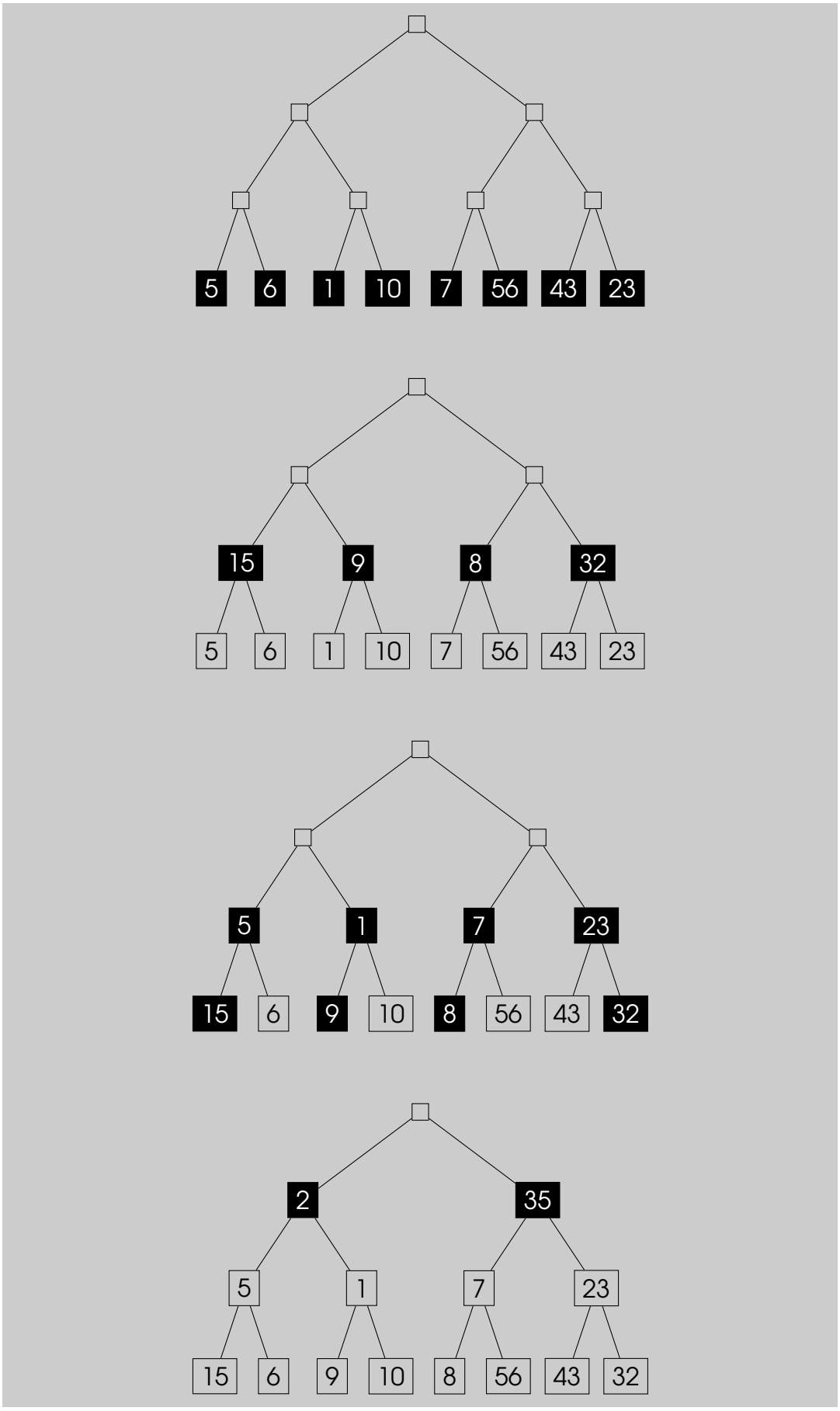
Solution:

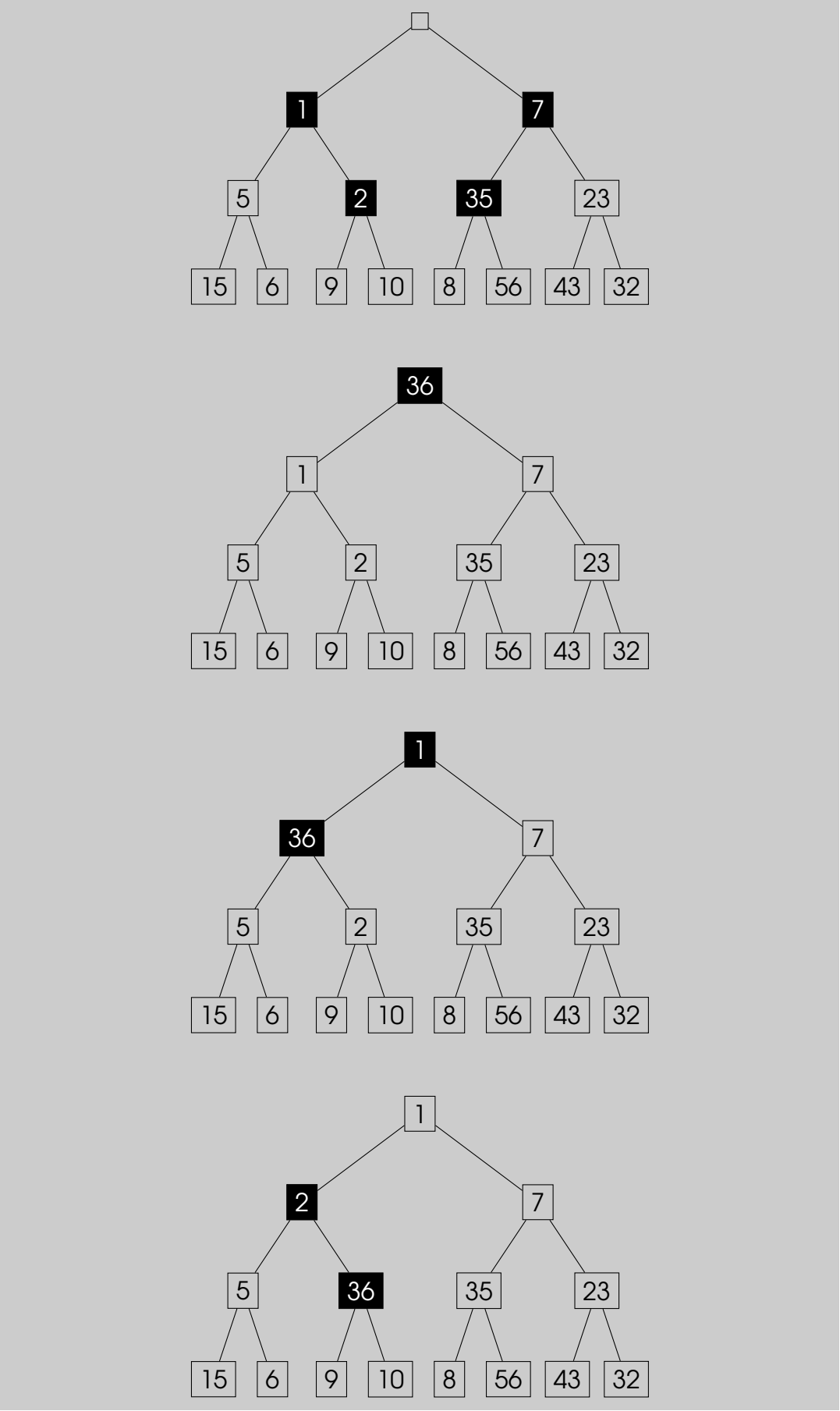
1. 2 (1 mark)
2. 1 (1 mark)
3. Yes (1 mark)
4. 2 1 5 3 7 (2 mark)

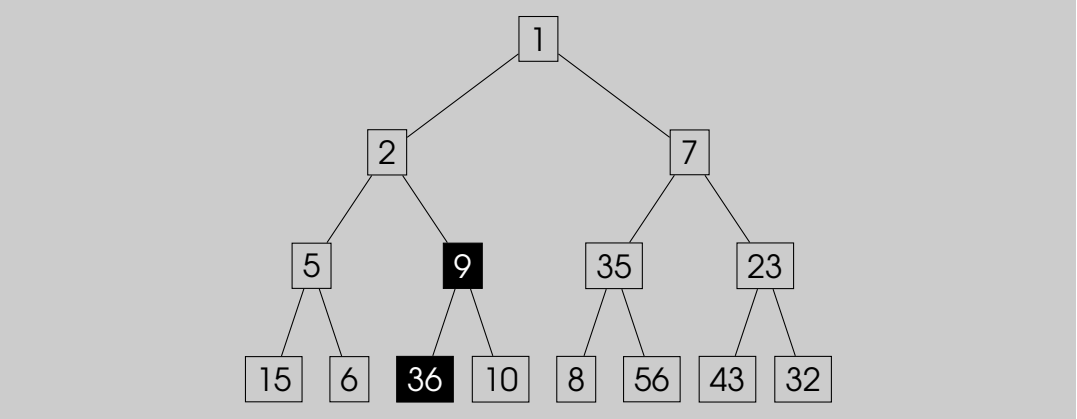
(b) Illustrate the execution of the **bottom-up construction of a heap** on the following sequence. You only need to provide a graphical representation of the heap at each stage in the construction, including any intermediate operations. (10)

(5, 6, 1, 10, 7, 56, 43, 23, 15, 9, 8, 32, 2, 35, 36)

Solution:


5 6 1 10 7 56 43 23
15 9 8 32
5 6 1 10 7 56 43 23
5 1 7 23
15 6 9 10 8 56 43 32
2 35
5 1 7 23
15 6 9 10 8 56 43 32

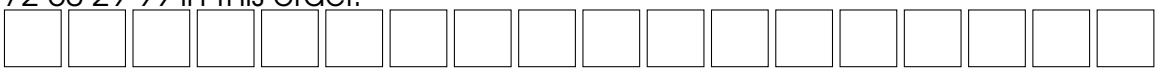

1
7
5
2
35
23
15
6
9
10
8
56
43
32
36
1
7
5
2
35
23
15
6
9
10
8
56
43
32
1
36
7
5
2
35
23
15
6
9
10
8
56
43
32
1
2
7
5
36
35
23
15
6
9
10
8
56
43
32



Total: 15

QUESTION 5

(a) Given a hash function $h(x) = xmod17$ for a hash table that uses **linear probing**, redraw the hash table below and **insert** the keys 84 46 47 22 72 60 29 99 in this order. (8)



0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Solution:

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

(b) What is the **load factor** for the above hash table **after** all the entries have been inserted? (2)

Solution:

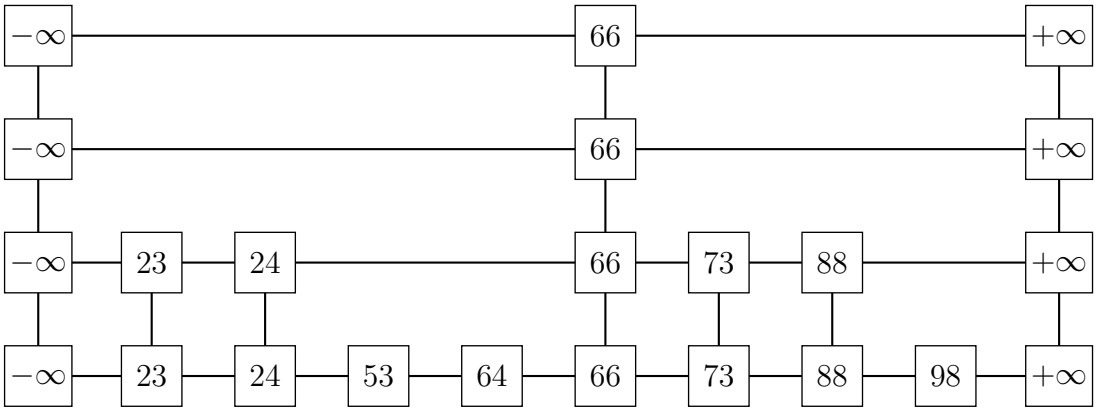
$8/17 = 0.47$

(c) Provide Java or pseudo source code for the **remove** method (that removes a key-pair e of type $Entry < K, V >$ from List S) in a **List-Based Dictionary**. (4)

Solution:

```
1 Algorithm remove(e):
2 Input: An entry e
3 Output: The removed entry e or null if e was not in D
4 {We don't assume here that e stores its location in S}
5 For each entry p in S.positions do
6   if p.element() = e then
7       S.remove(p)
8   return e
9 return null   {there is no entry e in D}
```

(d) Analyse the skip list below and illustrate using diagrams how you would **insert** an entry with a key of 76 and 2 heads coin flips. (6)



Solution:
Any diagram that:

1. facilitates a skip search from 66 in S3,
2. a drop down to S1,
3. a probe to 73.
4. Followed by a drop to the S0 list
5. an insert of the 76 with a height of 2 (all the way to S1).

Total: 20

QUESTION 6

The World Health Organisation (WHO) is responsible for collating any health-related information captured from around the world and performing analytics relevant to different health disease outbreaks. Discuss **three (3)** data structures that would be the most efficient way to implement this information system, along with its **worst case** run times of its **key** functions, **advantages** and **disadvantages**, and clearly indicate the reasons for your choice.

Solution:

Marks will be awarded based on the framework below (5 marks per option).

- The student will probably discuss a tree-based structure such as a Graph, AVL Tree, RB Tree or 24 Tree as the basis of storing an index
- In some cases the student may discuss a structure such as a hash table to store large amount of data indexed by a key and then searched based on that key.
- For the structures that the students chooses to discuss the advantages, disadvantages and efficiencies should be mentioned.

Total: 15

QUESTION 7

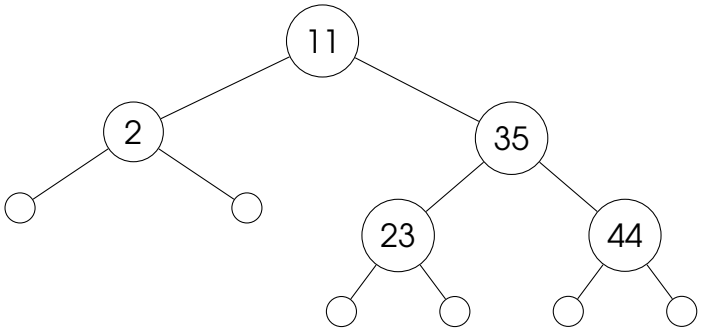
Consider the following AVL tree provided below. Draw the AVL tree state after each of the following operations. If the tree is rebalanced draw the state before and after it being balanced. Removal operations should follow from the tree that resulted from the insertion operations.

1. Insert nodes that contain the following keys: (inserted one-by-one, in the given order)

   9, 8, 40, 35, 43, 18, 3

2. Delete nodes that contain the following keys: (removed one-by-one, in the given order)

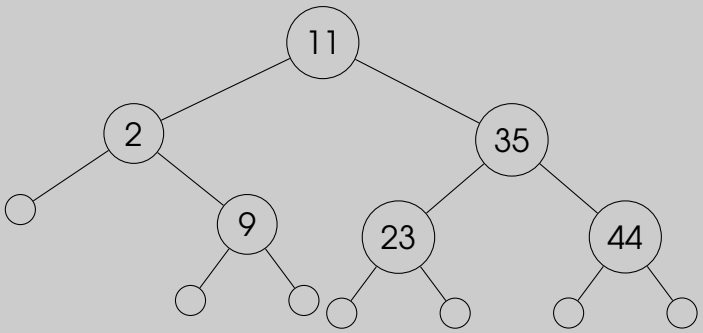
   8, 3



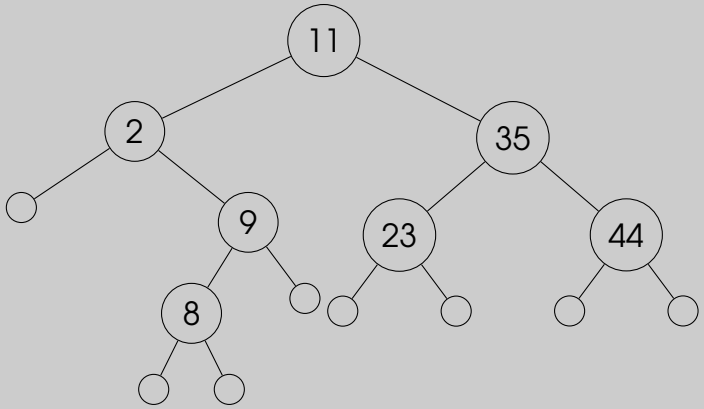
Solution:

Insert 9

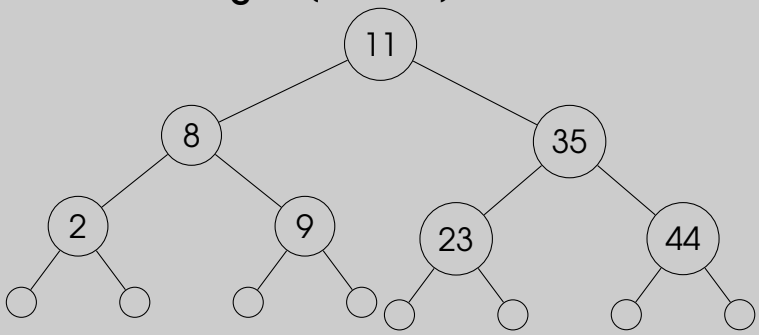
(1 mark):



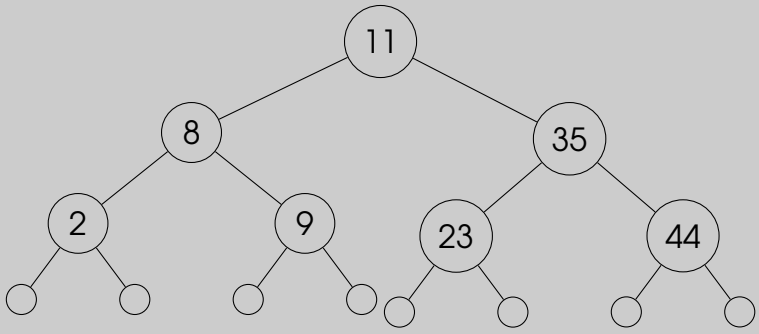
Insert 8



<<**Rebalancing**>> **(2 marks):**

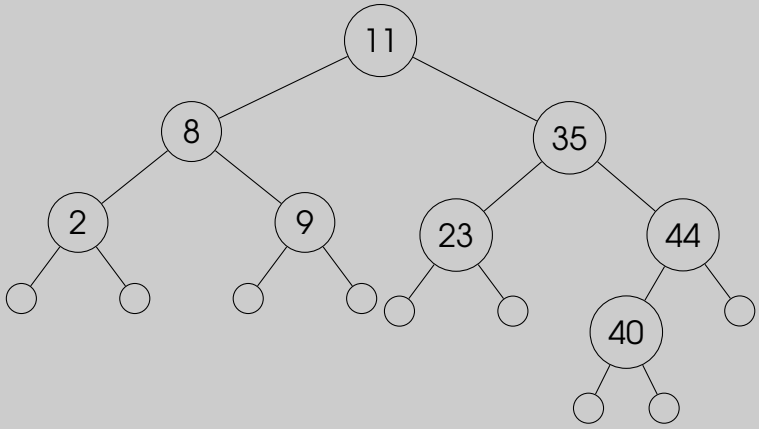


(1 mark):



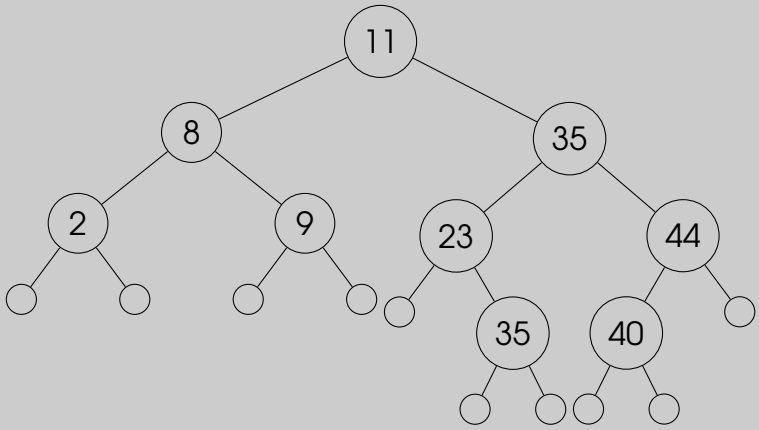
Insert 40

(1 mark):

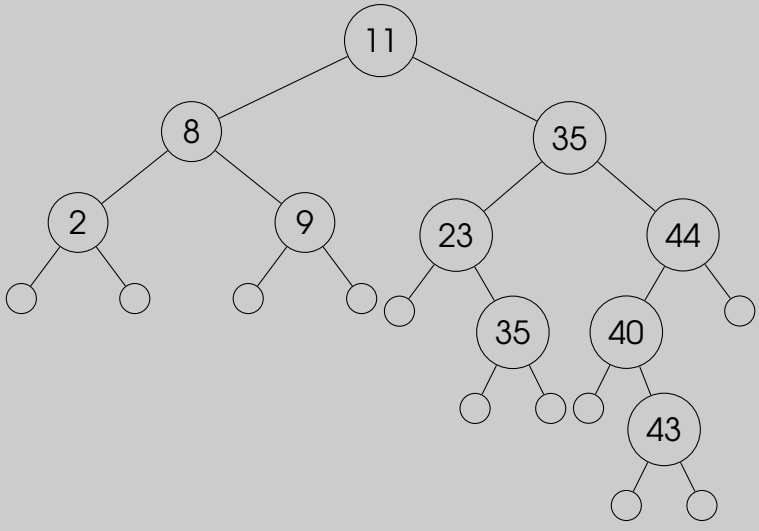


Insert 35

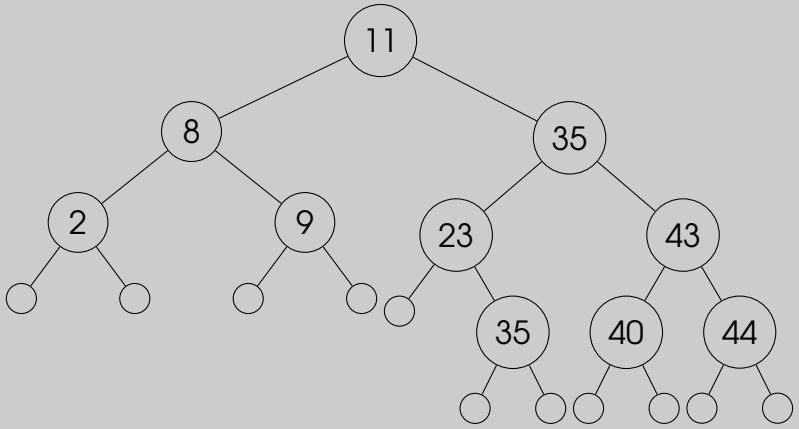
(1 mark):



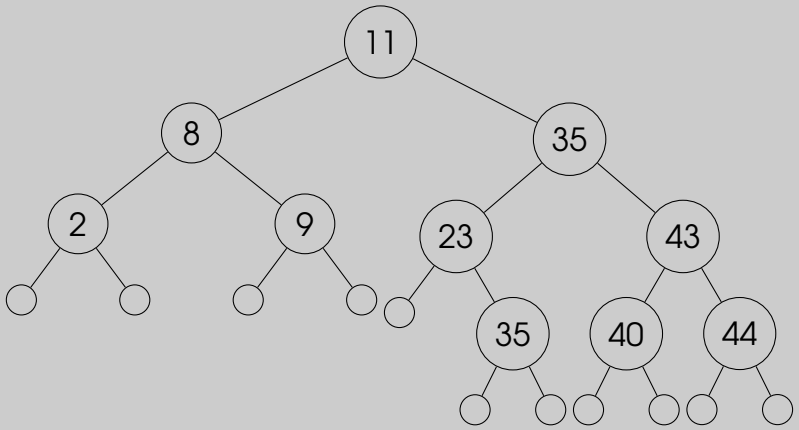
Insert 43



<<**Rebalancing**>> **(2 marks):**

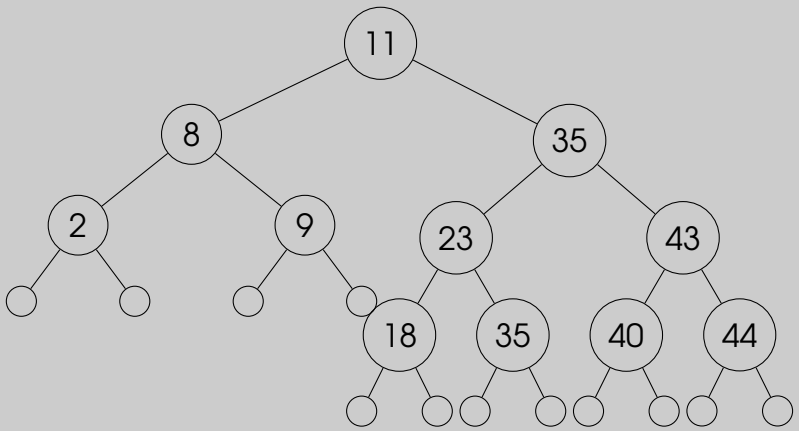


(1 mark):



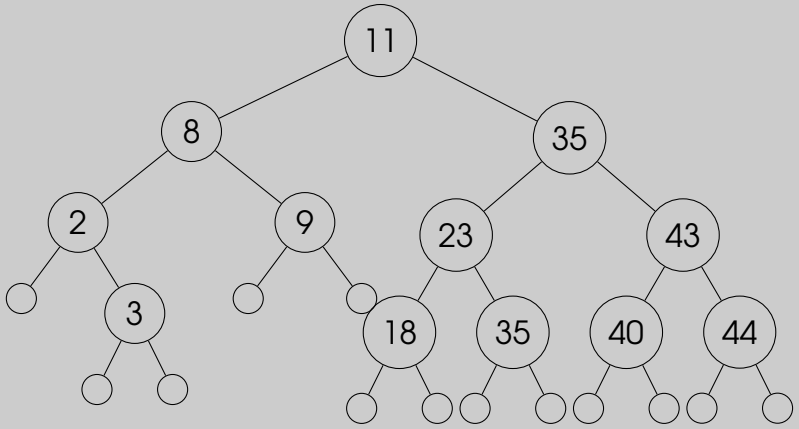
Insert 18

(1 mark):

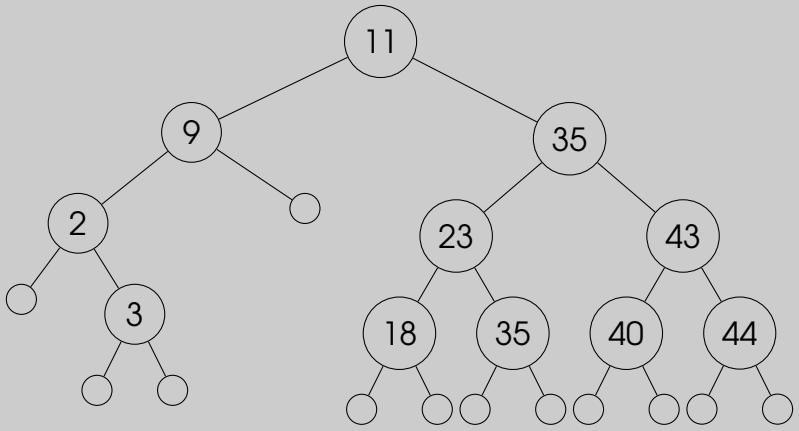


Insert 3

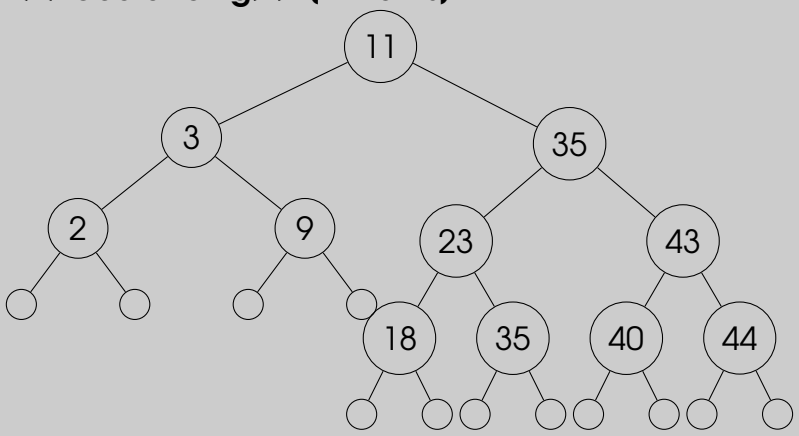
(1 mark):



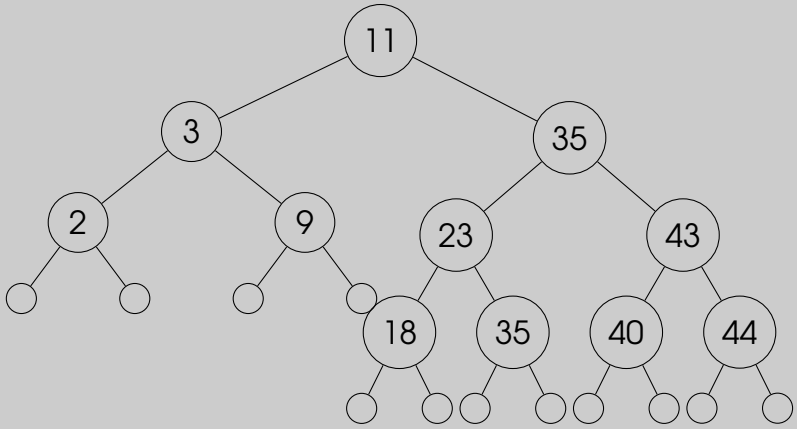
Remove 8



<<**Rebalancing**>> **(2 marks):**

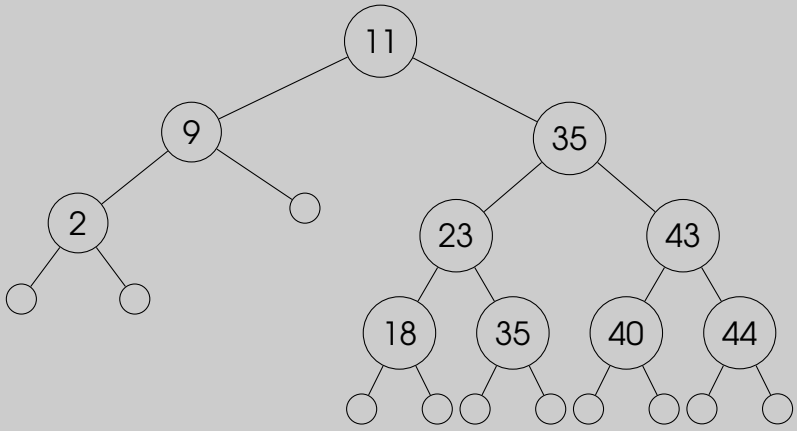


(1 mark):



Remove 3

(1 mark):



Total: 15 marks

Total: 15

QUESTION 8

Consider the following Red-Black tree provided below. Draw the Red-Black tree state after each of the following operations. If the tree is rebalanced draw the state before and after it being balanced. Removal operations should follow from the tree that resulted from the insertion operations.Removal operations should follow from the tree that resulted from the insertion operations.

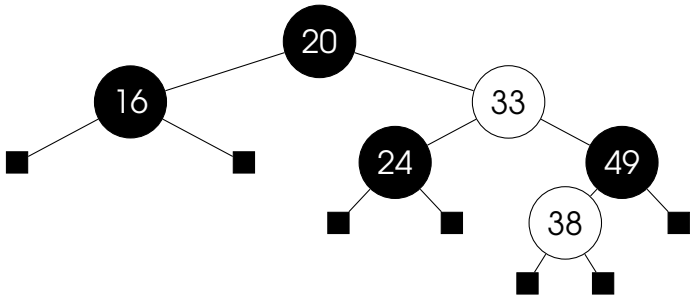
1. Insert nodes that contain the following keys: (inserted one-by-one, in the given order)

   28, 22, 29, 13, 20, 0

2. Delete nodes that contain the following keys: (removed one-by-one, in the given order)

   33, 28, 49, 29, 24, 22

The Red-Black tree is in the current state:

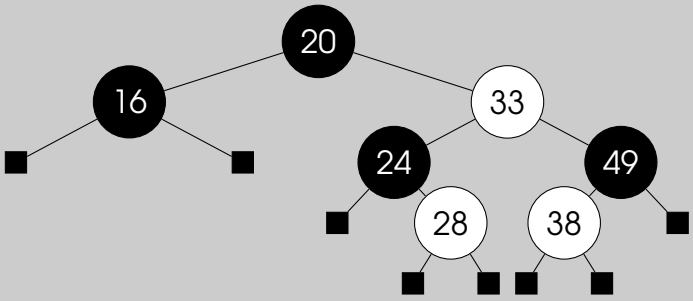


Solution:

The following is a guide to the allocated marks, the student might combine operations into a single step (in which case the marks are granted). In the even of a mistake for a particular step, marks will be deducted for that step, and then partial marks will be awarded for the steps following the mistake (based on the correctness of the step relative to the mistake). If the number of mistakes results in a completely incorrect representation no marks will be awarded

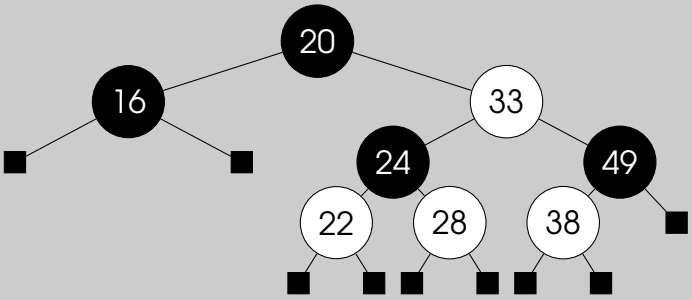
Insert 28

(1 mark):



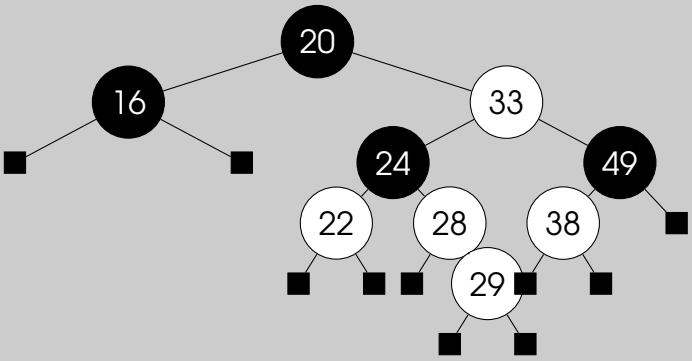
Insert 22

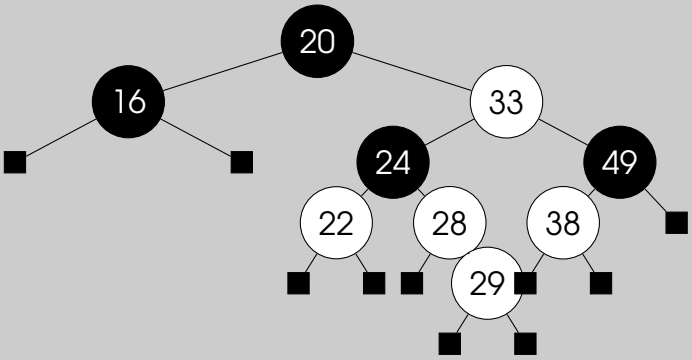
(1 mark):

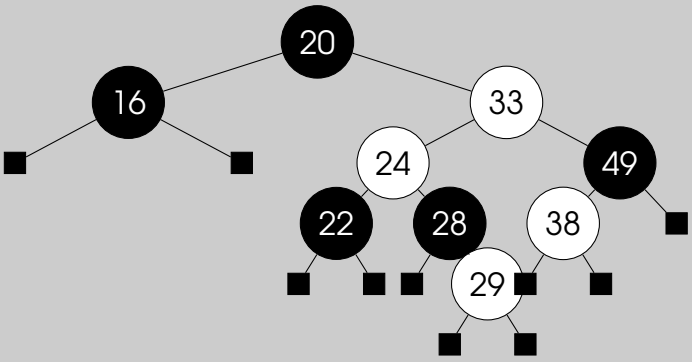


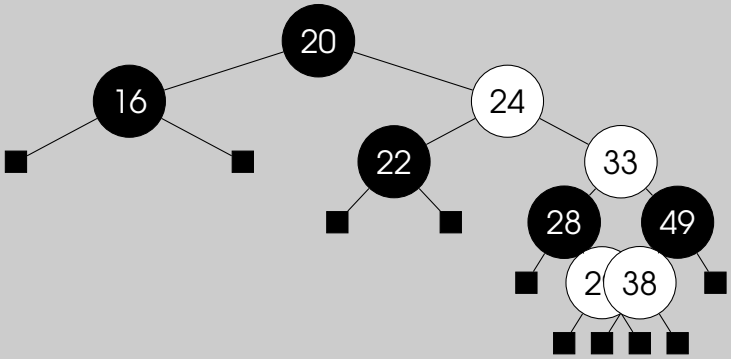
Insert 29

(1 mark):

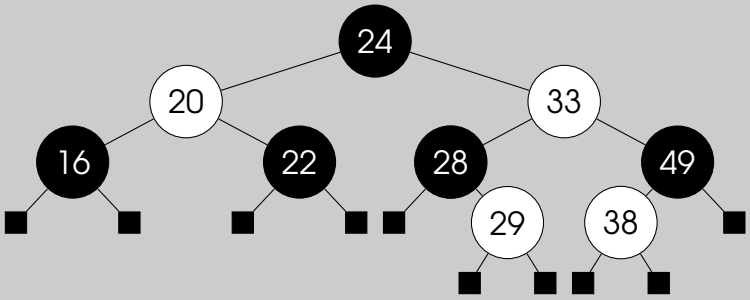




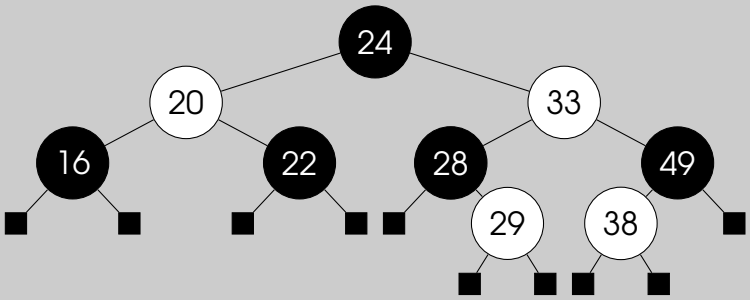




<<**Restructure**>> **(2 marks):**

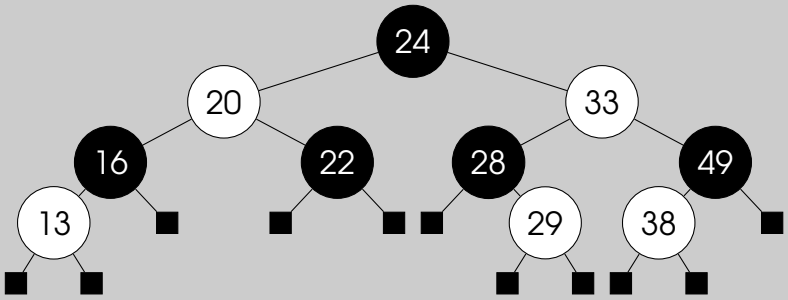


<<**Recolor**>> **(2 marks):**



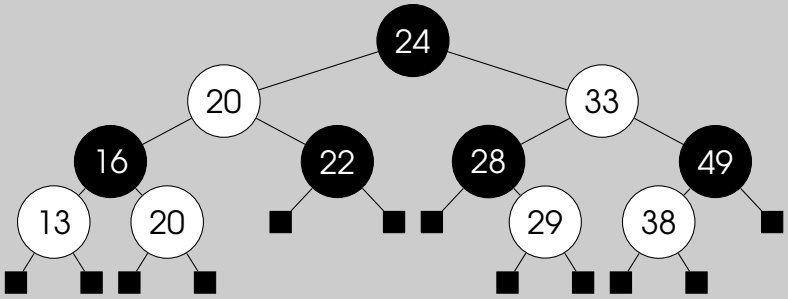
Insert 13

(1 mark):



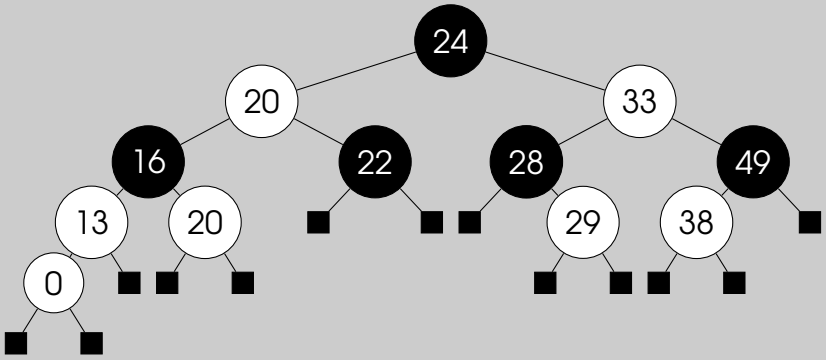
Insert 20

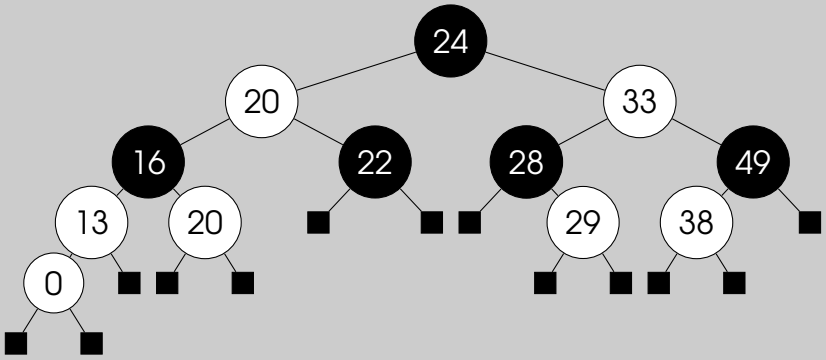
(1 mark):

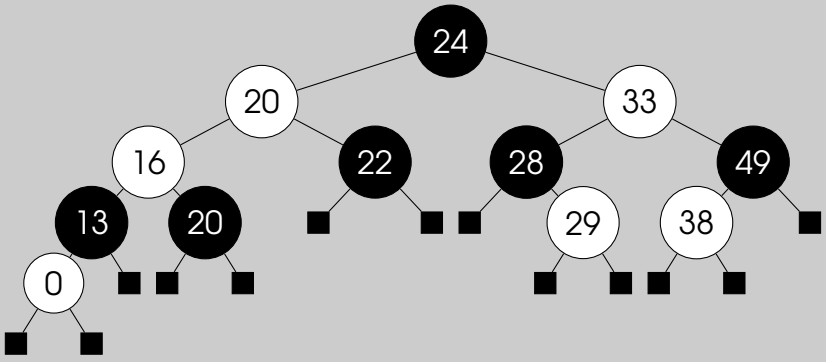


Insert 0

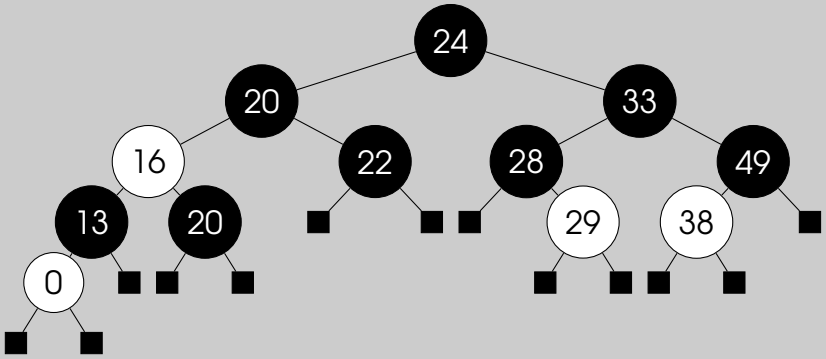
(1 mark):



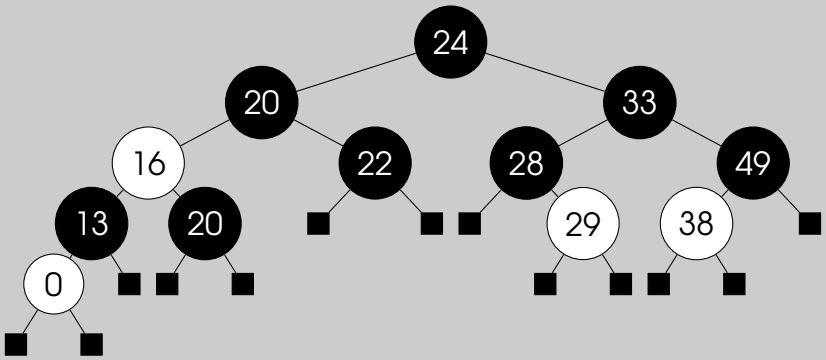




<<**Recolor**>> **(2 marks):**

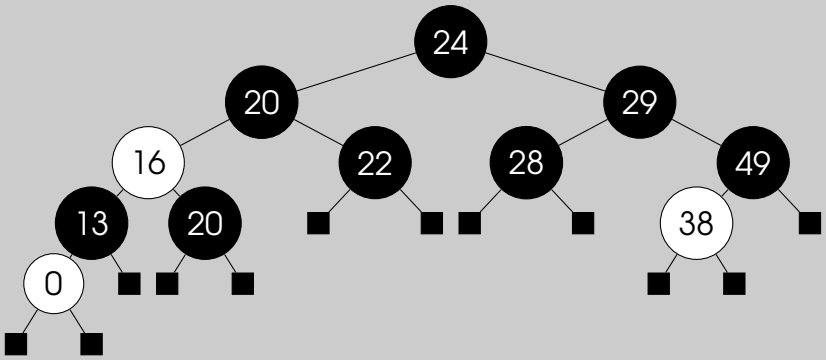


<<**Recolor**>> **(2 marks):**



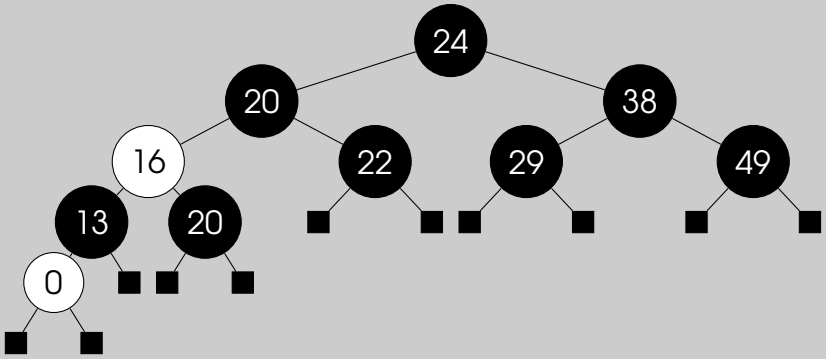
Remove 33

(1 mark):



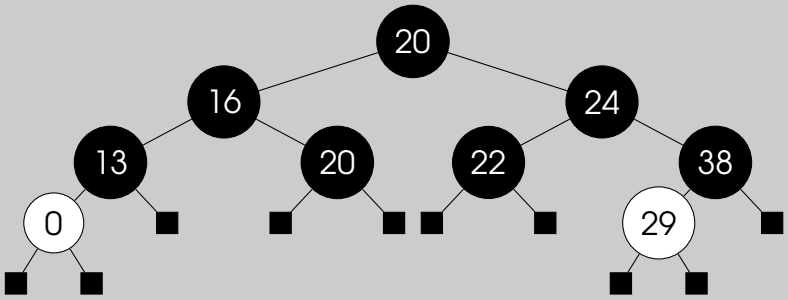
Remove 28

(1 mark):



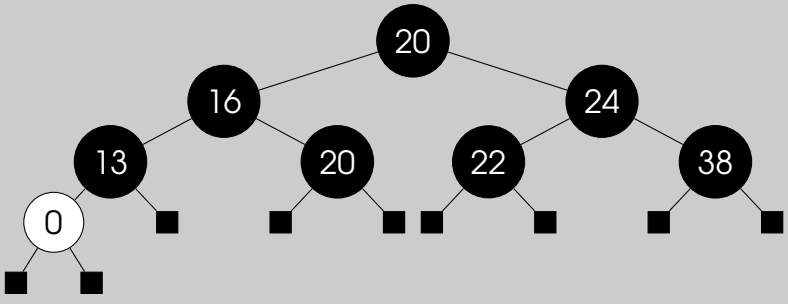
Remove 49

(1 mark):



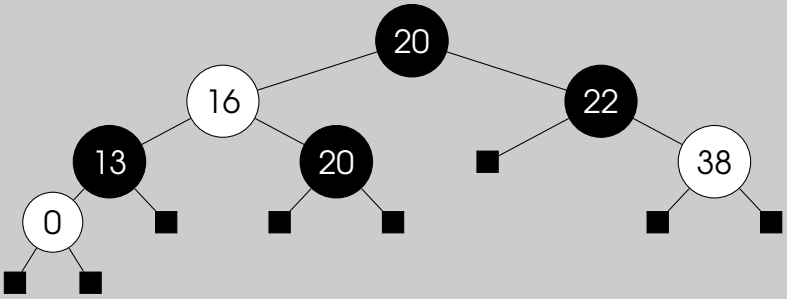
Remove 29

(1 mark):



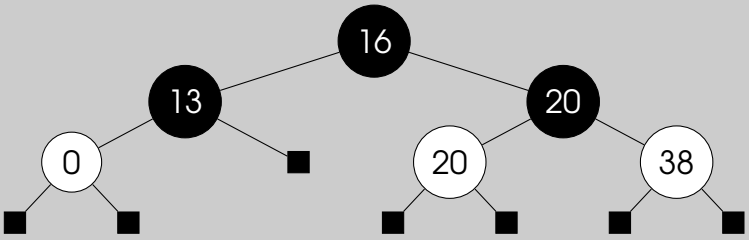
Remove 24

(1 mark):



Remove 22

(1 mark):

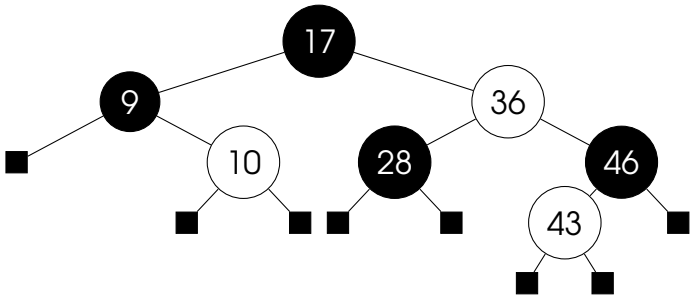


Total: 20 marks

Total: 20

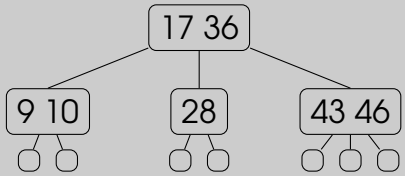
QUESTION 9

(a) Given the Red-Black tree below: (5)

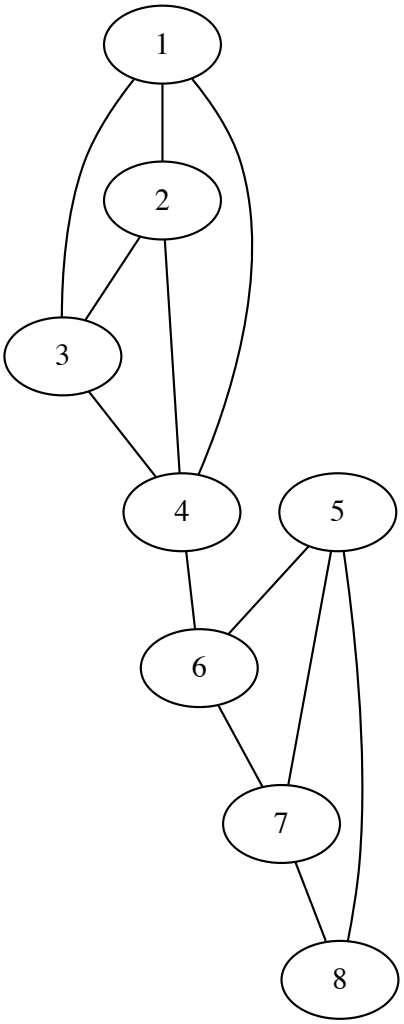


Provide a **(2,4) tree** representation that is equivalent to the above Red-Black Tree.

Solution:



(b) Analyse the undirected graph representation below and answer the question that follows: (10)


1
2
3
4
5
6
7
8

Show how the vertices will be visited if a **Depth First Search (DFS)** is performed, starting at **1**, along with whether you think a DFS or breadth first search (BFS) will reach vertix **8** faster. You may use a figure to support your answer.

Solution:

```
() - means already visited (back edge)
[1 -> 3,4,6,7,8]
[1 -> 2,(3),(4),(6),(7),(8)]
[2 -> (1),(3),(4)]
[4 -> (3),(2),(1),6]
[6 -> (4),5,7]
[5 -> (6),(7),8]
[7 -> (6),(5),(8)]
[8 -> (7),(5)]
DFS is much faster in this graph [2 marks]
```

(c) Provide a **proof** (by contradiction) for the following theorem: (5)
A digraph admits a topological ordering if and only if it is a Directed Acyclic Graph

Solution:

Assume that G is not acyclic I.e., there is a cycle C in G Let the edges in the cycle C be:

(v_{i0}, v_{i1}) (v_{i1}, v_{i2}) ... (v_{ik}, v_{i0}) According to the given fact that G has a topological ordering, we have from the edges in the cycle C that: v_{i0} ¡ v_{i1} ¡ v_{i2} ... ¡ v_{ik} ¡ v_{i0} This inequality is impossible — contradiction Therefore, the assumption that "G is not acyclic" is false:

G is acyclic

Total: 20

— End of paper —