QUESTION 1

(a) Analyze the code below (which computes the product of the first n entries of an integer array) and answer the questions that follow. [5]

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

1. Is there anything wrong with the code segment?
2. Justify your answer.

(b) Within the body of a method in Java, the keyword **this** is automatically defined as a reference to the instance upon which the method was called. Discuss two reasons why the **this** reference could be needed in a method body. [4]

(c) How would you go about **adding** an element **after** a given node in a **doubly linked list**? You may use pseudo code or diagrams to support your answer. [6]

Total: 15

QUESTION 2

(a) Consider the following function and using **primitive counting**, express the runtime of this function in Big-Oh notation, along with a justification for your answer. [5]

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

(b) Provide Java source that represents an **iterative** version of the following function. [5]

```
1  public static int factorial(int n) throws IllegalArgumentException {
2     if (n < 0)
3        throw new IllegalArgumentException( );
4     else if (n == 0)
5        return 1;
6     else
7        return n * factorial(n-1);
8  }
```

(c) Discuss the **Sequence ADT** together with an example of where it can be applied. [5]

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]

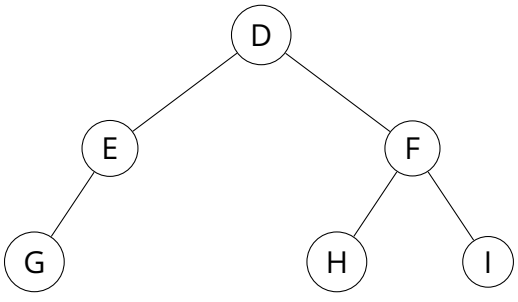
```
1  public interface List<T> {
2  public Node<T> addAfter(Node<T> elem, T item);
3  public Node<T> addFirst(T item);
4  public Node<T> addLast(T item);
5  public T remove(Node<T> elem);
6  public Node<T> search(T elem);
7  public Node<T> first();
8  public boolean isEmpty();
9  public Integer size();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

(b) Discuss how a Priority Queue can be used to sort a set of comparable elements. Be sure to include the two possible implementations together with the performance of the methods for these implementations. [5]

Total: 15

QUESTION 4

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



Provide the answers to the following:

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

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

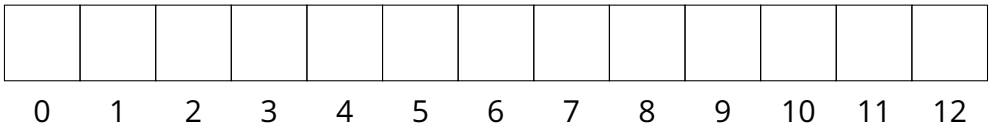
(3, 42, 22, 7, 9, 55, 32, 13, 27, 9, 8, 20, 11, 21, 37)

Total: 15

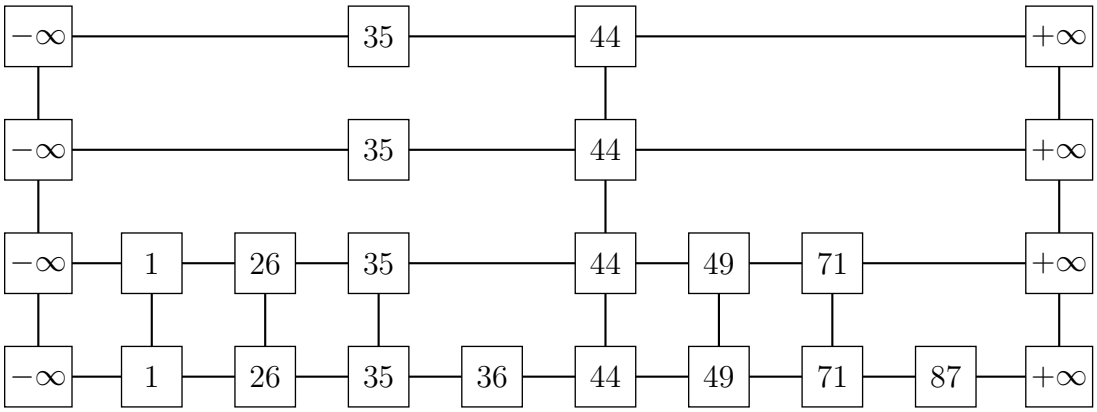
QUESTION 5

(a) How does the performance of a **list-based map** compare to that of a **hash-table map** implementation that uses separate chaining to handle collisions? For each implementation, give the performance of each method - *get*, *put*, and *remove* - in Big Oh notation. [6]

(b) Given a hash function $h(x) = xmod6$ for a hash table that uses **linear probing**, redraw the hash table below and **insert** the keys 6, 81, 87, 96, 38, 2, 21, 59 in this order (**note** - you only need to draw the final state). [8]



(c) Analyse the skip list below and illustrate using diagrams how you would **insert** an entry with a key of 64 and 3 heads coin flips. [6]



Total: 20

QUESTION 6

The South African Police Service has approached you to implement an efficient suspect lookup system that can also retrieve family members of suspects. They envision each citizen in the system having a vector representation of their right thumb fingerprint, citizen attributes and citizen behaviour. Your system must efficiently handle frequent searches at scale of suspects against the citizens in the Graph.

Upon analysing this problem, your team determines that the best data structure to use for this problem is Graph ADT. Discuss how you would **structure the nodes and edges** (3 marks), along with the **algorithms** (3 marks) that could be used to solve the given problem best. You must include a discussion of a **scenario** (3 marks) that concretely demonstrates its use with the aid of diagrams, followed by a discussion of **advantages, disadvantages** (3 marks), and **runtime efficiencies** (3 marks) of the implementation.

Total: 15

QUESTION 7

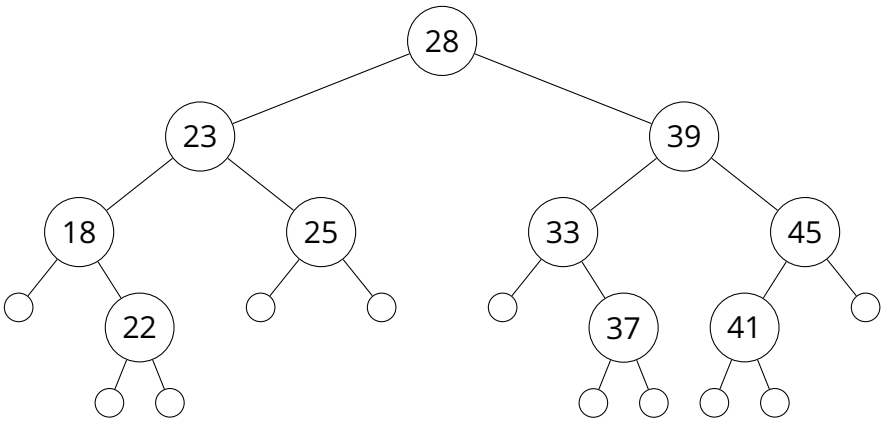
Consider the AVL tree provided below. Draw the AVL tree state after each of the following operations. If the tree is rebalanced, draw the state before and after it is balanced. If there is a duplicate key, the inorder predecessor should be used. Removal operations should follow from the tree that resulted from the insertion operations (i.e. removals take place after all the inserts have been completed).

1. Insert nodes that contain the following keys: (inserted one-by-one, in the given order, using an inorder predecessor duplicate strategy)

   3, 12, 23

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

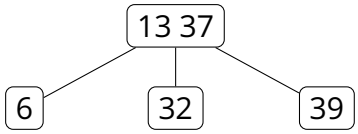
   25, 23, 28, 33, 12, 37



Total: 15

QUESTION 8

(a) Consider the (2,4) tree provided below: [5]



Draw the Red-Black Tree representation of the above (2,4) tree.

(b) Assuming the (2,4) tree is in the above state (the leaf nodes are not shown. However, they are assumed to exist), draw the (2,4) tree state after each of the following operations. If the tree is rebalanced, draw the state before and after it is balanced. [10]

1. Insert nodes that contain the following keys: (inserted one-by-one, in the given order, using an inorder predecessor duplicate strategy)

   2, 46, 22, 41, 47

2. Delete nodes that contain the following keys: (removed one-by-one, in the given order, using an inorder predecessor removal strategy)

   2, 47, 6

Total: 15

QUESTION 9

Consider the 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 is balanced. If there is a duplicate key, the inorder predecessor should be used. Removal operations should follow from the tree that resulted from the insertion operations.

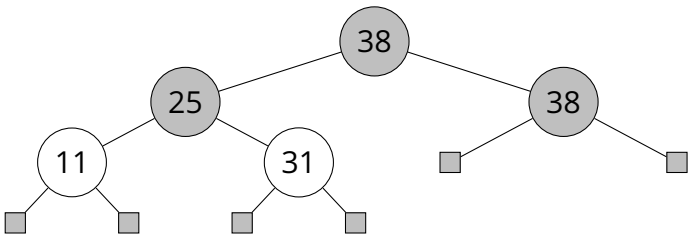
1. Insert nodes that contain the following keys: (inserted one-by-one, in the given order)

   18, 30, 31, 28

2. Delete nodes that contain the following keys: (removed one-by-one, in the given order, using an inorder predecessor removal strategy)

   25, 38, 38

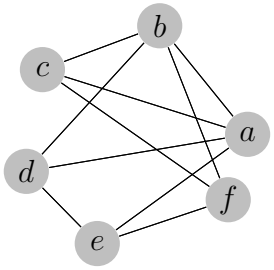
The Red-Black tree is in the current state:



Total: 15

QUESTION 10

(a) Analyse the undirected graph representation below and answer the question that follows: [6]



What would be the **adjacency matrix representation** of the above graph? Assume the vertix list is as follows $\{a, b, c, d, e, f\}$

(b) Using the above graph, show what would be the first four vertices visited if a **Breadth First Search (BFS)** is performed, starting at **a**. You may use a figure to support your answer. [4]

Total: 10

— End of paper —