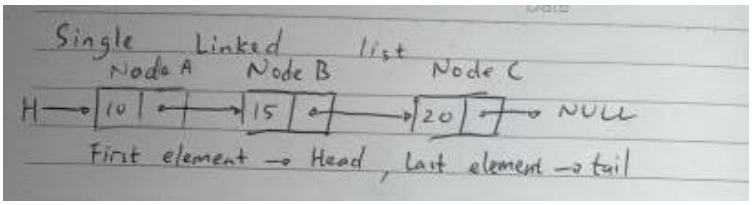
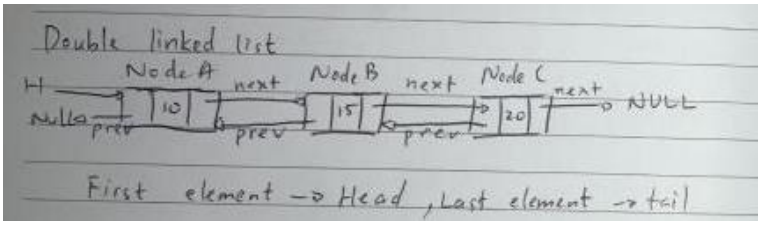
Linked List

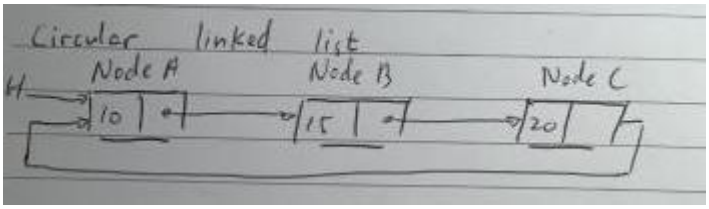
1. Explain single, double, and circular linked list in a graphical view! (.pdf, .png)
    a. Single linked list, each node has a data and only a next pointer, the head and tail aren't connected.



    b. Double linked list, each node has a data and two pointers, previous pointer and next pointer, this makes us go forward or backward, the head and tail are also not connected.



    c. Circular linked list is similar to a linked list, but the head and tail are connected, making a loop-like appearance.



2. What are the main differences between Linked List and Array? (.pdf)

Linked list	**Array**
• More dynamic and flexible(Memory can be expanded and contracted based on the number of elements) • Can contain non-primitive data types like structure, and the elements inside the linked list are linked to each other. • Must go from the head to the element we want.(Akses secara linear) • Memory utilization is more efficient compared to array.	• Have a fixed size • Only contains similar data (data types) • Can be accessed directly from the index • Because of having a fixed size, array's memory utiliziation is inefficient.

3. Explain Floyd's algorithm and implementation including its pseudocode! (.pdf)

The Floyd Warshall's algorithm is used to find the minimum cost/distance of each pairs, in the pseudocode below, i->j.

```
START
   for temp := 0 to size, do
      for i := 0 to size, do
         for j := 0 to size, do
            if cost[i,temp] + cost[temp,j] < cost[i,j], then
               cost[i,j] = cost[i,temp] + cost[temp,j]
         ENDFOR
      ENDFOR
   ENDFOR
   display minimal cost matrix
END
```

i and j is used to represent the vertices in a node, i represents the node we are in and j represents the destination of the node, temp is used as a temporary place, i -> temp ->, the if condition is used to find the minimal cost to go to j from i, if the cost of (i->temp->j) is smaller then i->j, then the cost array will be changed to the cost of (i->temp->j).

Stack and Queue

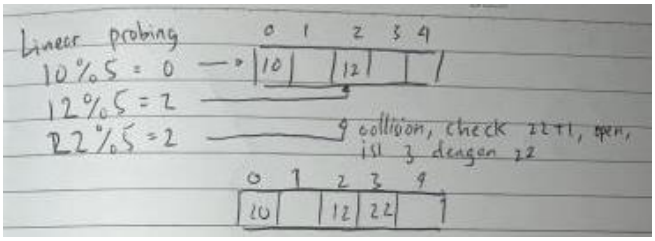
1. What are the main differences between Stack and Queue? (.pdf)

Stack	**Queue**
• Stack uses the principle of Last In First Out, which means that the last object added will be the first one to be removed, the program will most likely to use pushHead(inserting) and popHead(deletion) or pushTail(inserting) and popTail(deletion).	• Queue uses the principle of Last In Last Out, which means that the last object added will be the last object to be removed, the program will most likely use pushHead(inserting) and popTail(deletion) or pushTail(inserting) and popHead(deletion)

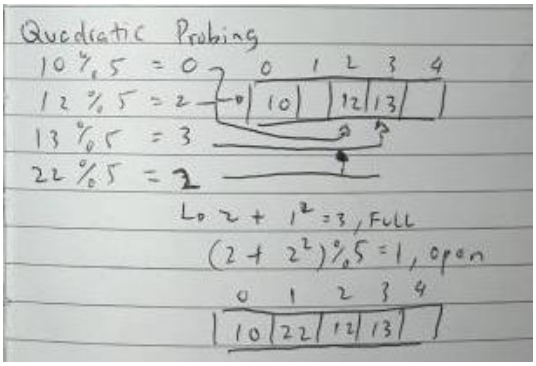
2. Explain prefix, infix, and postfix notation and its implementation using stack! (.pdf)
    a. prefix, prefix means the operator is behind the operands, for example -> operator operand operand, + 5 2
    b. infix, infix means that the operator is between the two operands, for example -> operand operator operand, 5 + 2
    c. postfix, postfix means that the operator is in front of the two operands, for example -> operand operand operator, 5 2 +

Hashing and Hash Tables

1. Explain what is a hashtable, hash function, and collision! (.pdf)
    a. hash table is a data structure which stores data in an array, and each data has an unique index, based on hash value. With this, we can access the data very fast only by knowing the unique data's index.
    b. hash function is a function which is used to calculate hash value.
    c. collision is what will happen if 2 data have the same hash value and the solution is with collision handling.
2. Explain 2 methods for collision handling and simulate the process in a graphical view! (.pdf, .png)
    a. linear probing is a method where we traverse the hash table by one until we find an empty spot then fill the spot with the new data.

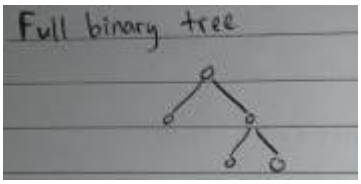


b. quadratic probing is similar with linear probing but instead of moving by one, quadratic probing moves by quadratic, for example the spot x is filled then, we'll check x + 1*1, if it's still full then we'll check x + 2*2 and so on until an open slot can be found.

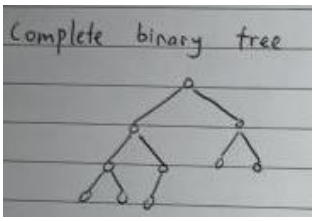


Binary Seach Tree

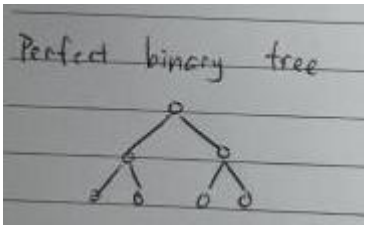
1. Explain 5 types of Binary Tree and draw each of them! (.pdf)
   a. Full binary tree the tree should only have either no children or two children.



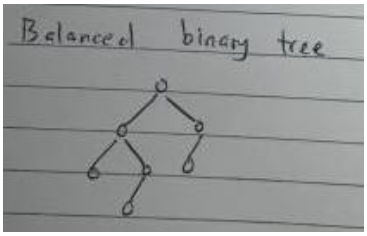
   b. Complete binary tree, each level must be filled with nodes, and at the last level, the node should be at the left side.



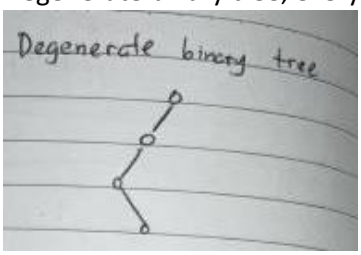
   c. Perfect binary tree every node have two children.



   d. Balanced binary tree, the left and right subtrees can only vary by one at most.



e. Degenerate binary tree, every node only has a single child, like a linked list



2. Simulate and explain clearly step by step the process of insertion: 24, 18, 55! (.pdf, .png)
    a. 24
        i. check if 24 > 27, 24 is smaller so traverse left
        ii. check if 24 > 14, 24 is bigger so traverse right
        iii. check if there's a child, (no child so) put 24 as 19's child
    b. 18
        i. check if 18>27, 18 is smaller so traverse left
        ii. check if 18 > 14, 18 is bigger so traverse left
        iii. check if there's a child, (no child so) put 18 as 19's child
    c. 55
        i. check if 55>27, 55 is bigger so traverse right
        ii. check if 55>35, 55 is bigger so traverse right
        iii. check if there's a child, (no child so) put 55 as 42's child.
3. Simulate and explain clearly step by step the process of deletion: 27, 35, 42! (.pdf, .png)
    a. 27
        i. search for what is deleted, (root), so
        ii. traverse to find the biggest number on the right subtree,
        iii. traverse to the right child of the root(14), then traverse to the left until the lowest level(31).
        iv. change the root to 31(also pop 27)
    b. 35
        i. search for what is deleted, (has more than 1 child), so
        ii. traverse to the right once to find the substitute(42)
        iii. change 35 with 42 (also pop 35)
    c. 42.
        i. search for what is deleted(leaf), so
        ii. traverse to the leaf,
        iii. pop the leaf 42.