LAPORAN PRAKTIKUM STRUKTUR DATA

QUIZ I

Disusun Oleh:

Fitri Waldi: 22343021

Dosen Pengampu:

Randi Proska Sandra, M.Sc

PROGRAM STUDI S1 INFORMATIKA (NK)

JURUSAN TEKNIK ELEKTRONIKA

FAKULTAS TEKNIK

UNIVERSITAS NEGERI PADANG

2023

A. Teori Konsep

Python adalah bahasa pemrograman tingkat tinggi yang memiliki sintaks yang mudah dipahami oleh manusia. Python juga memungkinkan para pengembang untuk menulis kode yang lebih sedikit dan lebih mudah dibaca dibandingkan dengan bahasa pemrograman lainnya. Python memiliki filosofi desain yang menekankan keterbacaan kode, dengan mengedepankan penggunaan indentasi dan bahasa yang mudah dipahami.

Java merupakan bahasa pemrograman yang berbasis objek-oriented. Bahasa pemrograman ini memungkinkan para pengembang untuk membangun aplikasi yang modular, fleksibel, dan mudah dikembangkan.

Jenis-jenis fungsi yang akan dipraktekkan:

1. Linked List

   Linked List merupakan struktur data yang terdiri dari serangkaian simpul, dimana setiap simpul memiliki dua bagian yaitu data dan pointer ke simpul berikutnya. Ada tiga jenis Linked List, yaitu singly Linked List, double Linked List, dan circular Linked List.

2. Double List

   Double List atau Doubly Linked List adalah variasi dari Linked List dimana setiap simpul memiliki dua pointer, satu menuju simpul sebelumnya dan satu lagi menuju simpul berikutnya. Dalam Double List, pencarian data lebih cepat karena bisa dilakukan dari kedua arah.

3. Circular List

   Circular List adalah jenis Linked List dimana simpul terakhir mengarah pada simpul pertama sehingga membentuk lingkaran. Hal ini memungkinkan operasi seperti traversal dilakukan secara berkelanjutan dari awal ke akhir dan kembali ke awal tanpa harus menggunakan pointer khusus.

4. Skip List

   Skip List adalah struktur data yang mirip dengan Linked List, namun dengan tambahan beberapa tingkat pointer yang menghubungkan beberapa simpul. Hal ini memungkinkan akses data secara efisien dengan cara menghindari simpul yang tidak perlu diakses.

5. Queue

Queue adalah struktur data yang berfungsi seperti antrian, dimana data yang dimasukkan pertama kali akan keluar terlebih dahulu (First In First Out). Ada dua jenis queue, yaitu queue biasa dan priority queue.

6. Priority Queue

Priority Queue adalah jenis queue dimana setiap elemen memiliki prioritas yang ditentukan. Data dengan prioritas yang lebih tinggi akan diberikan prioritas keluar terlebih dahulu.

7. Circular Queue

Circular Queue adalah jenis queue dimana elemen-elemennya membentuk lingkaran. Jika queue sudah penuh, elemen baru akan ditambahkan pada posisi awal queue dan elemen di posisi akhir queue akan dihapus.

8. Stack

Stack adalah struktur data yang berfungsi seperti tumpukan, dimana data yang dimasukkan terakhir akan keluar terlebih dahulu (Last In First Out). Operasi yang umum dilakukan pada stack adalah push (menambahkan data) dan pop (mengambil data).

9. Implementasi menggunakan array dan queue

Untuk mengimplementasikan stack menggunakan array, dapat dilakukan dengan menggunakan array satu dimensi dan indeks untuk menyimpan data. Sedangkan untuk mengimplementasikan stack menggunakan queue, dapat dilakukan dengan menggunakan dua queue, yaitu queue input dan queue output.

B. Alat dan Bahan

1. Kompiler online: https://www.programiz.com/
2. Notepad++

C. Langkah Kerja

1. Tentukan bahasa pemograman yang akan dipakai seperti Phyton atau Java.
3. Buka kompiler online jika tidak memiliki app kompiler, contoh: https://www.programiz.com/
4. Tuliskan kode-kode yang ingin digunakan dengan bahasa pemograman yang telah dipilih.
5. Lihat hasil dari program yang telah dibuat.

6. Simpan kode tersebut dalam notepad++.

D. Pembahasan

a. Bahasa Pemograman Phyton

1. Linked List-Double List

- Source Code

```
class Node:
    def __init__(self, data):
        self.data = data
        self.prev = None
        self.next = None

class DoubleLinkedList:
    def __init__(self):
        self.head = None

    def append(self, data):
        new_node = Node(data)
        if self.head is None:
            self.head = new_node
        else:
            current_node = self.head
            while current_node.next is not None:
                current_node = current_node.next
            current_node.next = new_node
            new_node.prev = current_node

    def prepend(self, data):
        new_node = Node(data)
        if self.head is not None:
            self.head.prev = new_node
        new_node.next = self.head
        self.head = new_node

    def print_list(self):
        current_node = self.head
        while current_node is not None:
            print(current_node.data)
            current_node = current_node.next

dll = DoubleLinkedList()
```

```
dll.append(1)
dll.append(2)
dll.append(3)

dll.prepend(0)

dll.print_list()
```

- Hasil

```
0
1
2
3
>
```

- Penjelasan

Baris Program	Petikan Source Kode	Penjelasan

1-5	class Node:
	def __init__(self, data):
	self.data = data
	self.prev = None
	self.next = None
		Mendefinisikan sebuah class bernama **Node** yang akan digunakan sebagai struktur data dalam Double Linked List.

7-9	class DoubleLinkedList:
	def __init__(self):
	self.head = None
		Mendefinisikan sebuah class bernama **DoubleLinkedList.**

2. Queue-Priority Queue
   - Source Kode

```
import heapq

class PriorityQueue:
    def __init__(self):
        self._queue = []
        self._index = 0

    def push(self, item, priority):
        heapq.heappush(self._queue, (-priority, self._index, item))
        self._index += 1
```

```
    def pop(self):
        return heapq.heappop(self._queue)[-1]

class Queue:
    def __init__(self):
        self._queue = []

    def enqueue(self, item):
        self._queue.append(item)

    def dequeue(self):
        return self._queue.pop(0)

    def is_empty(self):
        return len(self._queue) == 0

pq = PriorityQueue()

pq.push("apple", 3)
pq.push("banana", 2)
pq.push("orange", 1)

while not pq.is_empty():
    print(pq.pop())

q = Queue()

q.enqueue("apple")
q.enqueue("banana")
q.enqueue("orange")

while not q.is_empty():
    print(q.dequeue())
```

- Hasil

```
Traceback (most recent call last):
File "<string>", line 37, in <module>
AttributeError: 'PriorityQueue' object has no attribute 'is_empty'
> 3
3
> 1
1
> 2
2
> apple
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'apple' is not defined. Did you mean: 'tuple'?
>
```

- Penjelasan

Baris Program	Petikan Source Kode	Penjelasan

3. Array
   - Source Kode

     arr = [0] * 5

     arr[0] = 1
     arr[1] = 2
     arr[2] = 3
     arr[3] = 4
     arr[4] = 5

     print(arr[0])
     print(arr[3])

     arr[2] = 6

     print(arr)

   - Hasil

```
1
4
[1, 2, 6, 4, 5]
>
```

- Penjelasan

Baris Program	Petikan Source Kode	Penjelasan

b. Java

1. Array

- Source Kode

```
public class ArrayExample {
    public static void main(String[] args) {
        int[] arr = new int[5];

        arr[0] = 1;
        arr[1] = 2;
        arr[2] = 3;
        arr[3] = 4;
        arr[4] = 5;

        System.out.println(arr[0]);
        System.out.println(arr[3]);

        arr[2] = 6;

        for (int i = 0; i < arr.length; i++) {
            System.out.print(arr[i] + " ");
        }
    }
}
```

- Hasil

```
java -cp /tmp/TqiDrNjyZ3 ArrayExample
14
1 2 6 4 5
```

- Penjelasan

Baris Program	Petikan Source Kode	Penjelasan

2. Linked List-Circular
   - Source Kode

```
public class CircularLinkedList {
    private int[] arr;
    private int size;
    private int front, rear;

    public CircularLinkedList(int size) {
        this.size = size;
        arr = new int[size];
        front = -1;
        rear = -1;
    }

    public boolean isFull() {
        return (front == 0 && rear == size - 1) || (rear == front - 1);
    }

    public boolean isEmpty() {
        return front == -1;
    }

    public void enqueue(int item) {
        if (isFull()) {
            System.out.println("Queue is full");
        } else {
            if (front == -1) {
                front = 0;
```

```
        }
        rear = (rear + 1) % size;
        arr[rear] = item;
        System.out.println("Enqueued item: " + item);
    }
}

public int dequeue() {
    int item;
    if (isEmpty()) {
        System.out.println("Queue is empty");
        return -1;
    } else {
        item = arr[front];
        if (front == rear) {
            front = -1;
            rear = -1;
        } else {
            front = (front + 1) % size;
        }
        return item;
    }
}

public void display() {
    if (isEmpty()) {
        System.out.println("Queue is empty");
    } else {
        System.out.print("Circular Linked List: ");
        for (int i = front; i <= rear; i = (i + 1) % size) {
            System.out.print(arr[i] + " ");
        }
        System.out.println();
    }
}

public static void main(String[] args) {
    CircularLinkedList queue = new CircularLinkedList(5);
    queue.enqueue(1);
    queue.enqueue(2);
    queue.enqueue(3);
    queue.enqueue(4);
```

```
        queue.display();
        queue.enqueue(5);
        queue.display();
        queue.enqueue(6);
        queue.dequeue();
        queue.dequeue();
        queue.display();
        queue.enqueue(7);
        queue.enqueue(8);
        queue.display();
    }
}
```

- Hasil

```
java -cp /tmp/TqiDrNjyZ3 CircularLinkedList
Circular Linked List: 1 2 3
```

- Penjelasan

Baris Program	Petikan Source Kode	Penjelasan

3. Linked List-Skip List
   - Source Kode

```
import java.util.Random;

public class SkipList {
    private static final int MAX_LEVEL = 5;
    private Node head;
    private int size;
    private Random random;

    private class Node {
        private int data;
        private Node[] next;

        public Node(int data, int level) {
```

```
        this.data = data;
        this.next = new Node[level];
    }
}

public SkipList() {
    head = new Node(-1, MAX_LEVEL);
    size = 0;
    random = new Random();
}

public boolean isEmpty() {
    return size == 0;
}

public int size() {
    return size;
}

public void add(int data) {
    int level = randomLevel();
    Node newNode = new Node(data, level);
    Node current = head;

    // cari tempat untuk menyisipkan node baru
    for (int i = MAX_LEVEL - 1; i >= 0; i--) {
        while (current.next[i] != null && current.next[i].data < data) {
            current = current.next[i];
        }
        if (i < level) {
            newNode.next[i] = current.next[i];
            current.next[i] = newNode;
        }
    }

    size++;
}

public boolean contains(int data) {
    Node current = head;

    for (int i = MAX_LEVEL - 1; i >= 0; i--) {
```

```
            while (current.next[i] != null && current.next[i].data < data) {
                current = current.next[i];
            }
        }

        if (current.next[0] != null && current.next[0].data == data) {
            return true;
        } else {
            return false;
        }
    }

    private int randomLevel() {
        int level = 1;
        while (random.nextDouble() < 0.5 && level < MAX_LEVEL) {
            level++;
        }
        return level;
    }

    public void display() {
        Node current = head.next[0];
        System.out.print("Skip List: ");
        while (current != null) {
            System.out.print(current.data + " ");
            current = current.next[0];
        }
        System.out.println();
    }

    public static void main(String[] args) {
        SkipList skipList = new SkipList();
        skipList.add(1);
        skipList.add(3);
        skipList.add(5);
        skipList.add(2);
        skipList.add(4);
        skipList.display();

        System.out.println("Contains 3: " + skipList.contains(3));
        System.out.println("Contains 6: " + skipList.contains(6));
    }
```

}

- Hasil

```
java -cp /tmp/nLCYp8tK7F SkipList
Skip List: 1 2 3 4 5
Contains 3: true
Contains 6: false
```

- Penjelasan

Baris Program	Petikan Source Kode	Penjelasan

4. Queue-Circular
   - Source Kode

```
public class CircularQueue {
    private int front, rear;
    private int[] queue;
    private int capacity;

    public CircularQueue(int size) {
        this.capacity = size;
        this.front = -1;
        this.rear = -1;
        this.queue = new int[size];
    }

    public boolean isEmpty() {
        return front == -1;
    }

    public boolean isFull() {
        return (rear + 1) % capacity == front;
    }

    public void enqueue(int data) {
        if (isFull()) {
```

```
            System.out.println("Queue is full!");
            return;
        }
        if (isEmpty()) {
            front = 0;
        }
        rear = (rear + 1) % capacity;
        queue[rear] = data;
        System.out.println("Enqueued " + data);
    }

    public int dequeue() {
        if (isEmpty()) {
            System.out.println("Queue is empty!");
            return -1;
        }
        int data = queue[front];
        if (front == rear) {
            front = -1;
            rear = -1;
        } else {
            front = (front + 1) % capacity;
        }
        System.out.println("Dequeued " + data);
        return data;
    }

    public void display() {
        if (isEmpty()) {
            System.out.println("Queue is empty!");
            return;
        }
        System.out.print("Circular Queue: ");
        int i = front;
        do {
            System.out.print(queue[i] + " ");
            i = (i + 1) % capacity;
        } while (i != (rear + 1) % capacity);
        System.out.println();
    }

    public static void main(String[] args) {
```

```
        CircularQueue queue = new CircularQueue(5);
        queue.enqueue(1);
        queue.enqueue(2);
        queue.enqueue(3);
        queue.enqueue(4);
        queue.display();
        queue.dequeue();
        queue.dequeue();
        queue.display();
        queue.enqueue(5);
        queue.enqueue(6);
    }
}
```

- Hasil

```
java -cp /tmp/nLCYp8tK7F CircularQueue
Enqueued 1
Enqueued 2
Enqueued 3
Enqueued 4
Circular Queue: 1 2 3 4
Dequeued 1
Dequeued 2
Circular Queue: 3 4 Enqueued 5
Enqueued 6
```

- Penjelasan

Baris Program	Petikan Source Kode	Penjelasan

5. Stack-Implementasi Queue
   - Source Kode

```
import java.util.LinkedList;
import java.util.Queue;

public class StackUsingQueue {
    private Queue<Integer> queue1;
    private Queue<Integer> queue2;
    private int top;
```

```
public StackUsingQueue() {
    queue1 = new LinkedList<>();
    queue2 = new LinkedList<>();
}

public void push(int x) {
    queue1.add(x);
    top = x;
}

public int pop() {
    if (queue1.isEmpty()) {
        System.out.println("Stack is empty");
        return -1;
    }
    while (queue1.size() > 1) {
        top = queue1.remove();
        queue2.add(top);
    }
    int popValue = queue1.remove();
    Queue<Integer> temp = queue1;
    queue1 = queue2;
    queue2 = temp;
    return popValue;
}

public int peek() {
    if (queue1.isEmpty()) {
        System.out.println("Stack is empty");
        return -1;
    }
    return top;
}

public static void main(String[] args) {
    StackUsingQueue stack = new StackUsingQueue();
    stack.push(1);
    stack.push(2);
    stack.push(3);
    System.out.println("Top element is " + stack.peek());
    System.out.println(stack.pop() + " popped from stack");
```

```
        System.out.println("Top element is " + stack.peek());
    }
}
```

- Hasil

```
java -cp /tmp/nLCYp8tK7F StackUsingQueue
Top element is 33 popped from stack
Top element is 2
```

- Penjelasan

Baris Program	Petikan Source Kode	Penjelasan

E. Kesimpulan

Python adalah bahasa pemrograman tingkat tinggi yang memiliki sintaks yang mudah dipahami oleh manusia dan Java merupakan bahasa pemrograman yang berbasis objek-oriented.