Nama : Kesha Allam Ariqoh

NIM : 310700022410013

Prodi : Sains Data

1. a) untuk mengimplementasikan system pengelolaan data karyawan yang mencakup menambahkan data, menampilkan, mencari, dan menghapus data karyawan, struktur data yang paling sesuai adalah Binary Search Tree (BST). Alasaqnnya adalah pencarian yang cepat, pengurutan otomatis dikarenakan data karyawan otomatis terurut berdasarkan ID, penambahan dan penghapusan yang efisien

   b) **Algoritma Menambahkan Data Karyawan:**

```
def insert(root, id_karyawan, nama, jabatan):
    if not root:
        return Node(id_karyawan, nama, jabatan)
    elif id_karyawan < root.id:
        root.left = insert(root.left, id_karyawan, nama, jabatan)
    else:
        root.right = insert(root.right, id_karyawan, nama, jabatan)

    # Update height dan balance
    root.height = 1 + max(get_height(root.left), get_height(root.right))
    balance = get_balance(root)

    # Rotasi jika tidak seimbang
    if balance > 1 and id_karyawan < root.left.id:
        return rotate_right(root)
    if balance < -1 and id_karyawan > root.right.id:
        return rotate_left(root)
```

```
    if balance > 1 and id_karyawan > root.left.id:
        root.left = rotate_left(root.left)
        return rotate_right(root)
    if balance < -1 and id_karyawan < root.right.id:
        root.right = rotate_right(root.right)
        return rotate_left(root)
    return root
```

**Algoritma Menghapus Data Karyawan:**

```
def delete(root, id_karyawan):
    if not root:
        return root
    elif id_karyawan < root.id:
        root.left = delete(root.left, id_karyawan)
    elif id_karyawan > root.id:
        root.right = delete(root.right, id_karyawan)
    else:
        if not root.left:
            return root.right
        elif not root.right:
            return root.left
        temp = get_min_value_node(root.right)
        root.id = temp.id
        root.nama = temp.nama
        root.jabatan = temp.jabatan
        root.right = delete(root.right, temp.id)
```

```
        root.height = 1 + max(get_height(root.left), get_height(root.right))

        balance = get_balance(root)

        if balance > 1 and get_balance(root.left) >= 0:

            return rotate_right(root)

        if balance > 1 and get_balance(root.left) < 0:

            root.left = rotate_left(root.left)

            return rotate_right(root)

        if balance < -1 and get_balance(root.right) <= 0:

            return rotate_left(root)

        if balance < -1 and get_balance(root.right) > 0:

            root.right = rotate_right(root.right)

            return rotate_left(root)

        return root
```

c) - Menambah data karyawan:  Kompleksitas waktu: O(h), di mana h adalah tinggi pohon. Pada pohon seimbang, h = O(log n), sehingga kompleksitas rata-rata adalah O(log n). Namun, pada pohon yang tidak seimbang, h bisa mencapai O(n).

- Menampilkan semua data karyawan: Kompleksitas waktu: O(n), karena kita perlu melakukan traversal in-order untuk mengunjungi setiap node dalam pohon.

- Menghapus data karyawan: Kompleksitas waktu: O(h). Sama seperti pencarian, pada pohon seimbang adalah O(log n), tetapi pada pohon yang tidak seimbang bisa mencapai O(n).

2. a) Algoritma untuk Verifikasi Kode Sandi

```
def verifikasi_kode(kode):
    if len(kode) != 4 or not kode.isdigit():
        return False  # Kode harus 4 digit angka

    d1 = int(kode[0])
    d2 = int(kode[1])
    d3 = int(kode[2])
    d4 = int(kode[3])
```

```
    # Syarat 1: digit kedua tergantung digit pertama
    if d1 % 2 == 0:  # d1 genap
        if d2 <= d1:
            return False
    else:  # d1 ganjil
        if d2 >= d1:
            return False

    # Syarat 2: digit ketiga = d1 + d2
    if d3 != (d1 + d2):
        return False

    # Syarat 3: digit keempat = kebalikan dari digit kedua
    if d4 != (9 - d2):
        return False

    return True
```

b) Jumlah Kombinasi Kode Sandi yang mungkin adalah 20. Caranya:

```
def hitung_kombinasi():
    count = 0
    for d1 in range(10):
        if d1 % 2 == 0:
            d2_range = range(d1 + 1, 10)
        else:
            d2_range = range(0, d1)

        for d2 in d2_range:
            d3 = d1 + d2
            d4 = 9 - d2
            if 0 <= d3 <= 9:
                count += 1
    return count
```

c) Algoritma efisien dan waktu konstan karena ukuran input terbatas (4 digit dari 0–9), jumlah kombinasi sangat kecil, dan waktu proses tidak tumbuh mengikuti input. Ini membuat algoritma cepat, stabil, dan ringan dijalankan. Untuk komplesitas waktunya adalah O(1)