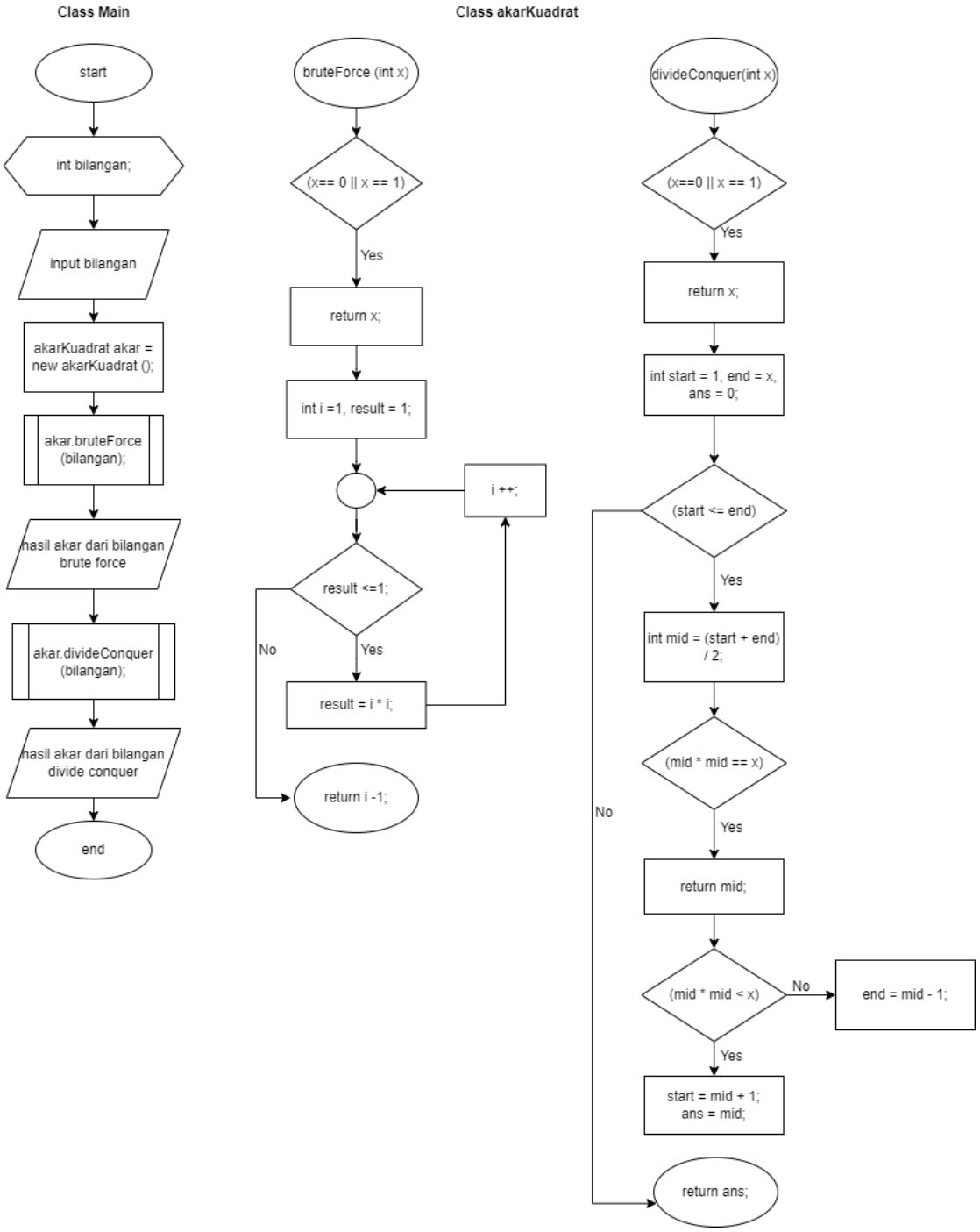
Nama : Deanissa Sherly Sabilla

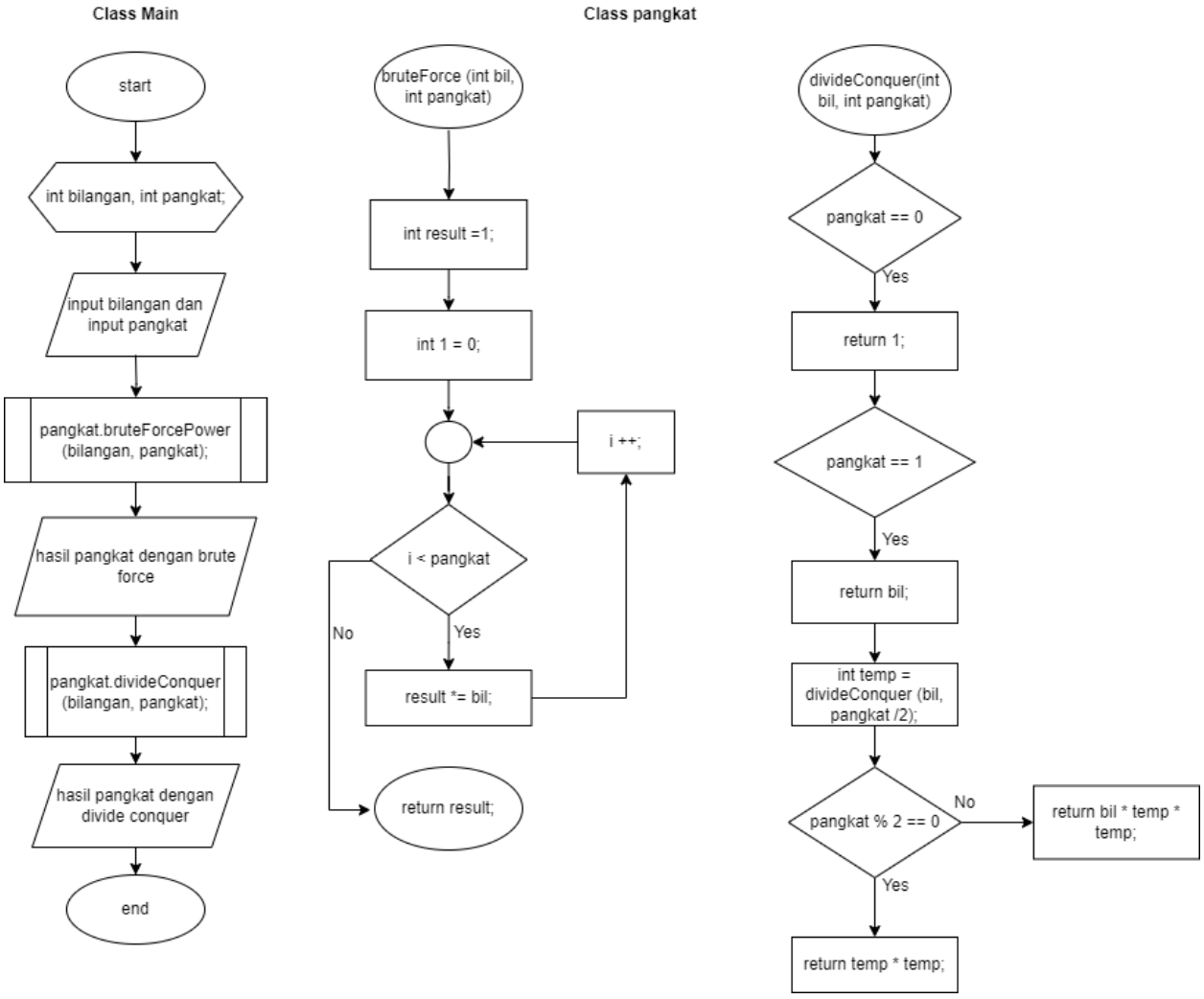
Kelas / Absen : SIB 1B / 06

-PERTEMUAN 5-

1. Buatlah flowchart untuk menghitung nilai akar dari suatu bilangan dengan algoritma Brute Force dan Divide Conquer! Jika bilangan tersebut bukan merupakan kuadrat sempurna, bulatkan angka ke bawah.



2. Buatlah flowchart untuk menghitung hasil pangkat dari inputan suatu bilangan dengan algoritma Brute Force dan Divide Conquer!



3. Tentukan notasi Big O yang sesuai dari kode program berikut!

```java
public int countVowels(char[] word){
    char[] vowels = {'a', 'i', 'u', 'e', 'o'};
    int count = 0;

    for (int i = 0; i < word.length; i++) {
        for (int j = 0; j < vowels.length; j++) {
            if (word[i] == vowels[j]) {
                count++;
            }
        }
    }

    return count;
}
```

- Notasi Big O dari kode program tersebut :
  1. Loop luar yang mengiterasi melalui setiap karakter dalam array `word`. Loop ini memiliki kompleksitas waktu O(n), di mana n adalah panjang array `word`.
  2. Loop dalam yang mengiterasi melalui setiap karakter dalam array `vowels`. Loop ini memiliki kompleksitas waktu tetap, yaitu O(1), karena ukuran array `vowels` selalu tetap.

  Karena loop dalam dikerjakan secara konstan untuk setiap iterasi loop luar, kompleksitas keseluruhan dari program ini adalah O(n * m), di mana n adalah panjang array `word` dan m adalah panjang array `vowels`.
  Jadi, notasi Big O untuk kode program tersebut adalah O(n).

4. Tentukan notasi Big O yang sesuai dari kode program berikut!

```java
public boolean checkItemInList(String item, String[] list) {
    for (int i = 0; i < list.length; i++) {
        if (list[i] == item) {
            return true;
        }
    }

    return false;
}
```

- Untuk menentukan notasi Big O dari kode program tersebut :
  1. Loop iterasi: Program memiliki satu loop yang mengiterasi melalui setiap elemen dalam array `list`. Jumlah iterasi tergantung pada panjang array `list`, yang dilambangkan sebagai `n`.

  2. Setiap iterasi melakukan satu operasi perbandingan antara elemen `list[i]` dengan `item`. Ini adalah operasi dengan kompleksitas waktu O(1), karena hanya membandingkan dua string.
  3. Jumlah operasi perbandingan dalam loop adalah sama dengan panjang array `list`, yaitu `n`.

  Jadi, notasi Big O dari kode program tersebut adalah O(n), di mana `n` adalah panjang array `list`.