2.1.3 Pertanyaan

1. Sebutkan dan tunjukkan masing-masing komponen perulangan FOR pada kode program Percobaan 1!
2. Mengapa variabel **tertinggi** diinisialisasi 0 dan **terendah** diinisialisasi 100? Apa yang terjadi jika variabel tertinggi diinisialisasi 100 dan terendah diinisialisasi 0?
3. Jelaskan fungsi dan alur kerja dari potongan kode berikut!

```
if (nilai > tertinggi) {
    tertinggi = nilai;
}
if (nilai < terendah) {
    terendah = nilai;
}
```

4. Modifikasi kode program sehingga terdapat perhitungan untuk menentukan berapa mahasiswa yang lulus dan yang tidak lulus berdasarkan batas kelulusan (nilai minimal 60). Tampilkan jumlah mahasiswa lulus dan tidak lulus setelah menampilkan nilai tertinggi dan terendah!
5. **Commit dan push kode program ke Github**

Jawab

1.

```
for (int i =1;i <= 10;i++){
```

for (Statement 1 ; Statement 2 ; Statement 3)
Statement 1 akan dieksekusi satu kali sebelum mengeksekusi kode

Statement 2 akan mengecek apakah kondisi benar atau salah.Jika benar kode di bawah akan dieksekusi sedangkan jika salah program akan keluar dari loop

Statement 3 akan dieksekusi setelah mengeksekusi 1 kali putaran

2.Variabel tertinggi di inisialisasikan 0 karena kita akan mencari nilai tertinggi dan variabel terendah di inisialisasikan 100 karena kita akan mencari variabel terendah.Hal ini ditujukan supaya pada saat sesi pengecekan if dan else tidak akan terbalik dan value akan berubah mengikuti inputan dari pengguna

3,if (nilai > tertinggi) yang artinya apabila nilai dari “nilai” lebih tinggi dari variabel maka akan mengeksekusi kode di dalamnya.

tertinggi = nilai ; artinya *mereplace* atau memperbarui nilai dari suatu variabel

if (nilai < terendah) apabila nilai dari “nilai” dibawah dari value terendah, maka akan mengeksekusi kode di dalamnya.

terendah = nilai; memperbarui nilai dari terendah ke value yang telah di input oleh user.

2.2.3 Pertanyaan

1. Pada potongan kode berikut, tentukan maksud dan kegunaan dari sintaks berikut:

```
if (nilai < 0 || nilai > 100) {
    System.out.println(x:"Nilai tidak valid. Masukkan lagi nilai yang valid!");
    continue;
}
```

   a. **nilai < 0 || nilai > 100**
   b. **continue**
2. Mengapa sintaks **i++** dituliskan di akhir perulangan WHILE? Apa yang terjadi jika posisinya dituliskan di awal perulangan WHILE?
3. Apabila jumlah mahasiswa yang dimasukkan adalah 19, berapa kali perulangan WHILE akan berjalan?
4. Modifikasi kode program sehingga apabila terdapat mahasiswa yang mendapat nilai A, program menampilkan pesan tambahan "Bagus, pertahankan nilainya"!
5. **Commit dan push kode program ke Github**

1.untuk mengecek apakah nilai tidak kurang dari 0 dan lebih dari 100.apabila nilai kurang dari 0 dan lebih dari 100, maka program akan mengoutput “Nilai tidak valid. Masukan lagi nilai yang valid!”

2.fungsi dari i++ adalah untuk menambah value i sebanyak 1 setelah mengeksekusi seluruh kode program di atasnya dan akan dieksekusi paling akhir .Pada intinya nilai i akan naik 1 setelah 1 kali loop

3.18 kali

4.

```
if (nilai>80 && nilai<=100){
    System.out.println("Nilai Mahasiswa ke-"+(n+1)+" adalah A");
    System.out.println(x:"Bagus pertahankan Nilainya!");
```

2.3.3 Pertanyaan

1. Pada penggunaan DO-WHILE ini, apabila nama pelanggan yang dimasukkan pertama kali adalah “batal”, maka berapa kali perulangan dilakukan?
2. Sebutkan kondisi berhenti yang digunakan pada perulangan DO-WHILE tersebut!
3. Apa fungsi dari penggunaan nilai **true** pada kondisi DO-WHILE?
4. Mengapa perulangan DO-WHILE tersebut tetap berjalan meskipun tidak ada komponen inisialisasi dan update?

1.Transaksi akan dibatalkan.

```
Masukan nama pelanggan (ketik 'batal' untuk keluar) : batal
transaksi dibatalkan
semua transaksi selesai
PS C:\Users\saya\daspro-jobsheet7>
```

2.kondisi berhenti yang digunakan do while tersebut adalah apabila kita memasukan batal pada permintaan input

3.Fungsi dari True pada DO-WHILE adalah untuk selalu menganggap kondisi program benar dan loop akan selamanya dilakukan kecuali jika kita mengimplementasikan break; pada kode.

4.Karena do akan mengeksekusi kode terlebih dahulu dan akan mengecek kondisi setelah kita mengeksekusi kode (misalnya dengan mengganti while).