LAPORAN TUGAS ALGORITMA DAN STRUKTUR DATA
JOBSHEET 10

`

NAMA : RADITYA RIEFKI
KELAS : TI 1E
ABSEN : 23

2. Praktikum 1

2.1 Percobaan 1 : Operasi Dasar Queue

Kode Program

```
package jobsheet11;

public class queue {
    int [] data;
    int front;
    int rear;
    int size;
    int max;

    public queue(int n){
        max = n;
        data = new int[max];
        size = 0;
        front = rear = -1;
    }

    public boolean IsEmpty(){
        if (size == 0) {
            return true;
        }else{
            return false;
        }
    }

    public boolean IsFull(){
        if (size == max) {
            return true;
        }else{
            return false;
        }
    }

    public void peek(){
        if (!IsEmpty()) {
            System.out.println("Elemen terdepan: " + data[front]);
        }else
        System.out.println("Queue masih kosong");
    }

    public void print(){
        if (IsEmpty()) {
            System.out.println("Queue masih kosong");
        }else{
            int i = front;
            while (i != rear) {
                System.out.println(data[i] + " ");
```

```
                i = (i + 1) % max;
            }
            System.out.println(data[i] + " ");
            System.out.println("Jumlah elemen = " + size);
        }
    }

    public void clear(){
        if (!IsEmpty()) {
            front = rear = -1;
            size = 0;
            System.out.println("Queue berhasil dikosongkan");
        }else{
            System.out.println("Queue masih kosong");
        }
    }
    public void enqueue(int dt){
        if (IsFull()) {
            System.out.println("Queue sudah penuh");
        }else{
            if (IsEmpty()) {
                front = rear = 0;
            }else{
                if (rear == max -1) {
                    rear = 0;
                }else{
                    rear++;
                }
            }
            data[rear] = dt;
            size++;
        }
    }
    public int dequeue(){
        int dt = 0;
        if (IsEmpty()) {
            System.out.println("Queue masih kosong");
        }else{
            dt = data[front];
            size--;
            if (IsEmpty()) {
                front = rear = -1;
            }else{
                if (front == max -1) {
                    front = 0;
                }else{
                    front++;
                }
            }
        }
        return dt;
    }
}
```

Kode Program Main

```
package jobsheet11;

import java.util.Scanner;

public class QueueMain {
    public static void menu(){
        System.out.println("Masukkan operasi yang diinginkan:");
        System.out.println("1. Enqueue");
        System.out.println("2. Dequeue");
        System.out.println("3. Print");
        System.out.println("4. Peek");
        System.out.println("5. Clear");
        System.out.println("--------------------------");
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Masukkan kapasitas queue: ");
        int n = sc.nextInt();

        queue q = new queue(n);

        System.out.print("Pilih menu: ");
        int pilih;

        do{
            menu();
            pilih = sc.nextInt();
            switch (pilih) {
                case 1:
                    System.out.print("Masukkan data baru: ");
                    int dataMasuk = sc.nextInt();
                    q.enqueue(dataMasuk);
                    break;
                case 2:
                    int dataKeluar = q.dequeue();
                    if (dataKeluar != 0) {
                    System.out.println("Data yang dikeluarkan: " +
dataKeluar);
                    break;
                }
                case 3:
                    q.print();
                    break;
                case 4:
                    q.peek();
                    break;
                case 5:
                    q.clear();
                    break;
            }
        }while (pilih == 1 || pilih == 2 || pilih == 3 || pilih == 4 ||
pilih == 5);
```

```
        }
    }
```

OUTPUT

```
Masukkan kapasitas queue: 4
Pilih menu: Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
--------------------------
1
Masukkan data baru: 15
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
--------------------------
1
Masukkan data baru: 31
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
--------------------------
4
Elemen terdepan: 15
Masukkan operasi yang diinginkan:
1. Enqueue
2. Dequeue
3. Print
4. Peek
5. Clear
--------------------------
PS D:\CollegeFile\SMT 2\ALSD>
```

2.1.3. Pertanyaan

1. Pada konstruktor, mengapa nilai awal atribut front dan rear bernilai -1, sementara atribut size bernilai 0?

Front dan rear di beri nilai -1 agar menandakan queue kosong (belum ada elemen terdepan) size diberi nilai 0 karena queue masih kosong dan tidak ada elemen

2. Pada method Enqueue, jelaskan maksud dan kegunaan dari potongan kode berikut!

```
if (rear == max - 1) {
    rear = 0;
```

Mengecek apakah pointer rear yang menunjukkan posisi terakhir dalam antrian sudah mencapai indeks terakhir array. Jika rear sudah di ujung array, maka pointer rear di-reset ke indeks 0 (awal array).

3. Pada method Dequeue, jelaskan maksud dan kegunaan dari potongan kode berikut!

```
if (front == max - 1) {
    front = 0;
```

Mengecek apakah pointer front (yang menunjukkan posisi elemen terdepan dalam antrian) sudah mencapai indeks terakhir array max - 1. Jika true maka front diinisiasi 0

4. Pada method print, mengapa pada proses perulangan variabel i tidak dimulai dari 0 (int i=0), melainkan int i=front?

Karena queue tidak selalu dimulai dari 0 tetapi dimulai dari front

5. Perhatikan kembali method print, jelaskan maksud dari potongan kode berikut!

```
i = (i + 1) % max;
```

memastikan pergerakan melingkar jika front atau rear melewati batas array.

6. Tunjukkan potongan kode program yang merupakan queue overflow!

```
public void enqueue(int dt){
    if (IsFull()) {
        System.out.println("Queue sudah penuh");
    }else{
        if (IsEmpty()) {
            front = rear = 0;
        }else{
            if (rear == max -1) {
                rear = 0;
            }else{
                rear++;
            }
        }
        data[rear] = dt;
        size++;
```

```
    }
  }
```

```
queue q = new queue(3);
q.enqueue(10);  // Data: [10, null, null], size=1
q.enqueue(20);  // Data: [10, 20, null], size=2
q.enqueue(30);  // Data: [10, 20, 30], size=3 (penuh)
q.enqueue(40);
```

7. Pada saat terjadi queue overflow dan queue underflow, program tersebut tetap dapat berjalan dan hanya menampilkan teks informasi. Lakukan modifikasi program sehingga pada saat terjadi queue overflow dan queue underflow, program dihentikan!

```
public void enqueue(int dt){
    if (IsFull()) {
        System.out.println(x:"Queue sudah penuh");
        System.exit(status:1);
```

```
public int dequeue(){
    int dt = 0;
    if (IsEmpty()) {
        System.out.println(x:"Queue masih kosong");
        System.exit(status:1);
```

2.2. Percobaan 2 : Antrian Layanan Akademik

Kode Program Mahasiswa

```
package jobsheet11.P2Jobsheet10;

public class Mahasiswa {
   String nim, nama, prodi, kelas;


public Mahasiswa (String nim, String nama, String prodi, String kelas){
   this.nim = nim;
   this.nama = nama;
   this.prodi = prodi;
   this.kelas = kelas;
   }
   public void tampilkanData(){
      System.out.println(nim + " - " + nama + " - " + prodi + " - " + kelas );
   }
}
```

Kode Program AntrianLayanan

```
package jobsheet11.P2Jobsheet10;

public class AntrianLayanan {
    Mahasiswa [] data;
    int front;
    int rear;
    int size;
    int max;

    public AntrianLayanan(int max){
        this.max = max;
        this.data = new Mahasiswa[max];
        this.front = 0;
        this.rear = -1;
        this.size = 0;
    }



    public boolean IsEmpty(){
        if (size == 0) {
            return true;
        }else{
            return false;
        }
    }

    public boolean IsFull(){
        if (size == max) {
```

```
            return true;
        }else{
            return false;
        }
    }

    public void peek(){
        if (!IsEmpty()) {
            System.out.println("Elemen terdepan: " + data[front]);
        }else
        System.out.println("Queue masih kosong");
    }

    public void print(){
        if (IsEmpty()) {
            System.out.println("Queue masih kosong");
        }else{
            int i = front;
            while (i != rear) {
                System.out.println(data[i] + " ");
                i = (i + 1) % max;
            }
            System.out.println(data[i] + " ");
            System.out.println("Jumlah elemen = " + size);
        }
    }

    public void clear(){
        if (!IsEmpty()) {
            front = rear = -1;
            size = 0;
            System.out.println("Queue berhasil dikosongkan");
        }else{
            System.out.println("Queue masih kosong");
        }
    }
    public void tambahAntrian(Mahasiswa mhs){
        if (IsFull()) {
            System.out.println("Antrian penuh, tidak dapat menambah
mahasiswa.");
            return;
        }
        rear = (rear + 1) % max;
        data[rear] = mhs;
        size++;
        System.out.println(mhs.nama + " Berhasil masuk ke antrian");
    }

    public Mahasiswa layaniMahasiswa(){
        if (IsEmpty()) {
            System.out.println("Antri kosong");
            return null;
        }
        Mahasiswa mhs = data[front];
```

```
        front = (front+1) % max;
        size-- ;
        return mhs;
    }

    public void lihatTerdepan(){
        if (IsEmpty()) {
            System.out.println("Antrian kosong");
        }else{
            System.out.print("Mahasiswa terdepan: ");
            System.out.println("NIM - NAMA - PRODI - KELAS");
            data[front].tampilkanData();
        }
    }

    public void tampilkanSemua(){
        if (IsEmpty()) {
            System.out.println("Antrian kosong.");
            return;
        }
        System.out.println("Daftar Mahasiswa dalam Antrian: ");
        System.out.println("NIM - NAMA - PRODI - KELAS");
        for (int i = 0; i < size; i++) {
            int index = (front + i) % max;
            System.out.print((i + 1) + ". ");
            data[index].tampilkanData();
        }
    }
    public int getJumlahAntrian(){
        return size;
    }
}
```

Kode Program LayananAkademikSIAKAD

```
package jobsheet11.P2Jobsheet10;

import java.util.Scanner;

public class LayananAkademiSIAKAD {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        AntrianLayanan antrian = new AntrianLayanan(5);
        int pilihan;

        do{
            System.out.println("\n=== Menu Antrian Layanan Akademik ===");
            System.out.println("1. Tambah Mahasiswa ke Antrian");
            System.out.println("2. Layani Mahasiswa");
            System.out.println("3. Lihat Mahasiswa Terdepan");
            System.out.println("4. Lihat Antrian");
            System.out.println("5. Jumlah Mahasiswa dalam Antrian");
            System.out.println("0. Keluar");
            System.out.print("Pilih menu: ");
            pilihan = sc.nextInt(); sc.nextLine();

            switch (pilihan) {
                case 1:
                    System.out.print("NIM   : ");
                    String nim = sc.nextLine();
                    System.out.print("Nama  : ");
                    String nama = sc.nextLine();
                    System.out.print("Prodi : ");
                    String prodi =sc.nextLine();
                    System.out.print("Kelas : ");
                    String kelas = sc.nextLine();
                    Mahasiswa mhs = new Mahasiswa(nim, nama, prodi, kelas);
                    antrian.tambahAntrian(mhs);
                    break;
                case 2:
                    Mahasiswa dilayani = antrian.layaniMahasiswa();
                    if (dilayani != null) {
                        System.out.print("Melayani Mahasiswa: ");
                        dilayani.tampilkanData();
                    }
                    break;
                case 3:
                    antrian.lihatTerdepan();
                    break;
                case 4:
                    antrian.tampilkanSemua();
                    break;
                case 5:
                    System.out.println("Jumlah dalam antrian: " +
antrian.getJumlahAntrian());
                    break;
                case 0:
```

```
                    System.out.println("Terima Kasih");
                    break;
                default:
                    System.out.println("Pilihan tidak valid.");
            }
        }while (pilihan !=0);
        sc.close();
    }
}
```

OUTPUT

```
=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Antrian
5. Jumlah Mahasiswa dalam Antrian
0. Keluar
Pilih menu: 1
NIM   : 123
Nama  : Aldi
Prodi : TI
Kelas : 1A
Aldi Berhasil masuk ke antrian

=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Antrian
5. Jumlah Mahasiswa dalam Antrian
0. Keluar
Pilih menu: 1
NIM   : 124
Nama  : Bobi
Prodi : TI
Kelas : 1G
Bobi Berhasil masuk ke antrian

=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Antrian
5. Jumlah Mahasiswa dalam Antrian
0. Keluar
Pilih menu: 4
Daftar Mahasiswa dalam Antrian:
NIM - NAMA - PRODI - KELAS
1. 123 - Aldi - TI - 1A
2. 124 - Bobi - TI - 1G
```

```
=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Antrian
5. Jumlah Mahasiswa dalam Antrian
0. Keluar
Pilih menu: 2
Melayani Mahasiswa: 123 - Aldi - TI - 1A

=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Antrian
5. Jumlah Mahasiswa dalam Antrian
0. Keluar
Pilih menu: 4
Daftar Mahasiswa dalam Antrian:
NIM - NAMA - PRODI - KELAS
1. 124 - Bobi - TI - 1G

=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Antrian
5. Jumlah Mahasiswa dalam Antrian
0. Keluar
Pilih menu: 5
Jumlah dalam antrian: 1

=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Antrian
5. Jumlah Mahasiswa dalam Antrian
0. Keluar
Pilih menu: 0
Terima Kasih
PS D:\CollegeFile\SMT 2\ALSD>
```

2.2.3 Pertanyaan

Lakukan modifikasi program dengan menambahkan method baru bernama LihatAkhir pada class AntrianLayanan yang digunakan untuk mengecek antrian yang berada di posisi belakang. Tambahkan pula daftar menu 6. Cek Antrian paling belakang pada class LayananAkademikSIAKAD sehingga method LihatAkhir dapat dipanggil!

Penambahan Methode LihatAkhir

```
public void LihatAkhir(){
      if (IsEmpty()) {
         System.out.println("Antrian Kosong");
      }else{
         System.out.print("Mahasiswa terdepan: ");
         System.out.println("NIM - NAMA - PRODI - KELAS");
         data[rear].tampilkanData();

      }
   }
```

```
case 6:
    antrian.LihatAkhir();
System.out.println(x:"6. Cek Antrian paling belakang ");
```

Output

```
=== Menu Antrian Layanan Akademik ===
1. Tambah Mahasiswa ke Antrian
2. Layani Mahasiswa
3. Lihat Mahasiswa Terdepan
4. Lihat Antrian
5. Jumlah Mahasiswa dalam Antrian
6. Cek Antrian paling belakang
0. Keluar
Pilih menu: 6
Mahasiswa terdepan: NIM - NAMA - PRODI - KELAS
2 - b - r - 2
```

2.3 Tugas

==========

| QueueKRS |

==========

| queue: Mahasiswa23[]|

| maxAntrian: int |

| front: int |

| rear: int |

| size: int |

| jumlahProses: int |

==========

| isEmpty(): boolean |

| isFull(): boolean |

| kosongkan(): void |

| tambahMahasiswa(m: Mahasiswa): void |

| panggilProsesKRS(): void |

| dequeue(): Mahasiswa |

| tampilkanSemua(): void |

| tampilkanDepan(): void |

| tampilkanDuaTerdepan(): void |

| tampilkanAkhir(): void |

| cetakJumlah(): void |

| cetakJumlahProses(): void |

| cetakBelumProses(): void |

==========

Kode program mahasiswa23

```
package jobsheet11;

public class Mahasiswa23 {
    public String nama;
    public String nim;
```

```
   public String prodi;
   public String kelas;

   public Mahasiswa23(String nama, String nim, String prodi, String kelas) {
      this.nama = nama;
      this.nim = nim;
      this.prodi = prodi;
      this.kelas = kelas;
   }
}
```

Kode Program queue KRS

```
package jobsheet11;
public class QueueKRS {
    private int maxAntrian = 10;
    private Mahasiswa23[] queue = new Mahasiswa23[maxAntrian];
    private int front = 0;
    private int rear = -1;
    private int size = 0;
    private int jumlahProses = 0;

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

    public boolean isFull() {
        return size == maxAntrian;
    }

    public void kosongkan() {
        front = 0;
        rear = -1;
        size = 0;
        System.out.println("Antrian telah dikosongkan.");
    }

    public void tambahMahasiswa(Mahasiswa23 mhs) {
        if (isFull()) {
            System.out.println("Antrian penuh, tidak bisa menambahkan
mahasiswa.");
        } else {
            rear = (rear + 1) % maxAntrian;
            queue[rear] = mhs;
            size++;
            System.out.println(mhs.nama + " ditambahkan ke antrian.");
        }
    }

    public void panggilProsesKRS() {
        if (size >= 2) {
```

```
            Mahasiswa23 m1 = dequeue();
            Mahasiswa23 m2 = dequeue();
            jumlahProses += 2;
            System.out.println("Memproses KRS untuk:");
            tampilkanMahasiswa(m1);
            tampilkanMahasiswa(m2);
        } else {
            System.out.println("Antrian tidak cukup untuk memproses 2
mahasiswa.");
        }
    }

    public Mahasiswa23 dequeue() {
        if (isEmpty()) return null;
        Mahasiswa23 mhs = queue[front];
        front = (front + 1) % maxAntrian;
        size--;
        return mhs;
    }

    public void tampilkanSemua() {
        if (isEmpty()) {
            System.out.println("Antrian kosong.");
        } else {
            System.out.println("Daftar antrian:");
            for (int i = 0; i < size; i++) {
                int index = (front + i) % maxAntrian;
                System.out.print((i + 1) + ". ");
                tampilkanMahasiswa(queue[index]);
            }
        }
    }

    public void tampilkanDepan() {
        if (!isEmpty()) {
            System.out.println("Antrian terdepan:");
            tampilkanMahasiswa(queue[front]);
        } else {
            System.out.println("Antrian kosong.");
        }
    }

    public void tampilkanDuaTerdepan() {
        if (size >= 2) {
            System.out.println("1. ");
            tampilkanMahasiswa(queue[front]);
            System.out.println("2. ");
            tampilkanMahasiswa(queue[(front + 1) % maxAntrian]);
        } else if (size == 1) {
            System.out.println("1. ");
            tampilkanMahasiswa(queue[front]);
            System.out.println("2. Tidak ada");
        } else {
            System.out.println("Antrian kosong.");
```

```
        }
    }

    public void tampilkanAkhir() {
        if (!isEmpty()) {
            System.out.println("Antrian terakhir:");
            tampilkanMahasiswa(queue[rear]);
        } else {
            System.out.println("Antrian kosong.");
        }
    }

    public void cetakJumlah() {
        System.out.println("Jumlah mahasiswa dalam antrian: " + size);
    }

    public void cetakJumlahProses() {
        System.out.println("Jumlah mahasiswa yang sudah melakukan KRS: " +
jumlahProses);
    }

    public void cetakBelumProses() {
        System.out.println("Jumlah mahasiswa yang belum melakukan KRS: " +
size);
    }

    private void tampilkanMahasiswa(Mahasiswa23 m) {
        if (m != null) {
            System.out.println("Nama : " + m.nama);
            System.out.println("NIM  : " + m.nim);
            System.out.println("Prodi: " + m.prodi);
            System.out.println("Kelas: " + m.kelas);
        }
    }
}
```

Kode Program Main

```
package jobsheet11;

import java.util.Scanner;

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

```
        QueueKRS antrian = new QueueKRS();
        Scanner scanner = new Scanner(System.in);
        int pilihan;

        do {
            System.out.println("\n=== MENU ANTRIAN KRS ===");
            System.out.println("1. Cek antrian kosong");
            System.out.println("2. Cek antrian penuh");
            System.out.println("3. Kosongkan antrian");
            System.out.println("4. Tambah mahasiswa ke antrian");
            System.out.println("5. Panggil 2 mahasiswa untuk proses KRS");
            System.out.println("6. Tampilkan semua antrian");
            System.out.println("7. Tampilkan 2 antrian terdepan");
            System.out.println("8. Tampilkan antrian terakhir");
            System.out.println("9. Cetak jumlah antrian");
            System.out.println("10. Cetak jumlah yang sudah proses KRS");
            System.out.println("11. Cetak jumlah yang belum proses KRS");
            System.out.println("0. Keluar");
            System.out.print("Pilih menu: ");
            pilihan = scanner.nextInt();
            scanner.nextLine();

            switch (pilihan) {
                case 1 :
                    System.out.println(antrian.isEmpty() ? "Antrian kosong."
: "Antrian tidak kosong.");
                    break;
                case 2 :
                    System.out.println(antrian.isFull() ? "Antrian penuh." :
"Antrian belum penuh.");
                    break;
                case 3 :
                    antrian.kosongkan();
                    break;
                case 4 :
                 {
                    System.out.print("Nama: ");
                    String nama = scanner.nextLine();
                    System.out.print("NIM: ");
                    String nim = scanner.nextLine();
                    System.out.print("Prodi: ");
                    String prodi = scanner.nextLine();
                    System.out.print("Kelas: ");
                    String kelas = scanner.nextLine();
                    Mahasiswa23 mhs = new Mahasiswa23(nama, nim, prodi,
kelas);
                    antrian.tambahMahasiswa(mhs);
                    break;
                }
                case 5 :
                    antrian.panggilProsesKRS();
                    break;
                case 6 :
                    antrian.tampilkanSemua();
```

```
                    break;
                case 7 :
                    antrian.tampilkanDuaTerdepan();
                    break;
                case 8 :
                    antrian.tampilkanAkhir();
                    break;
                case 9 :
                    antrian.cetakJumlah();
                    break;
                case 10 :
                    antrian.cetakJumlahProses();
                    break;
                case 11 :
                    antrian.cetakBelumProses();
                    break;
                case 0 :
                    System.out.println("Terima kasih!");
                    break;
                default :
                System.out.println("Pilihan tidak valid.");
            }

        } while (pilihan != 0);

    }
}
```

OUTPUT

```
1. Cek antrian kosong
2. Cek antrian penuh
3. Kosongkan antrian
4. Tambah mahasiswa ke antrian
5. Panggil 2 mahasiswa untuk proses KRS
6. Tampilkan semua antrian
7. Tampilkan 2 antrian terdepan
8. Tampilkan antrian terakhir
9. Cetak jumlah antrian
10. Cetak jumlah yang sudah proses KRS
11. Cetak jumlah yang belum proses KRS
0. Keluar
Pilih menu: 4
Nama: boy
NIM: 1
Prodi: TI
Kelas: 1E
boy ditambahkan ke antrian.
```

```
=== MENU ANTRIAN KRS ===
1. Cek antrian kosong
2. Cek antrian penuh
3. Kosongkan antrian
4. Tambah mahasiswa ke antrian
5. Panggil 2 mahasiswa untuk proses KRS
6. Tampilkan semua antrian
7. Tampilkan 2 antrian terdepan
8. Tampilkan antrian terakhir
9. Cetak jumlah antrian
10. Cetak jumlah yang sudah proses KRS
11. Cetak jumlah yang belum proses KRS
0. Keluar
Pilih menu: 5
Memproses KRS untuk:
Nama : boy
NIM  : 1
Prodi: TI
Kelas: 1E
Nama : Poe
NIM  : 2
Prodi: TI
Kelas: 1E
```

```
=== MENU ANTRIAN KRS ===
1. Cek antrian kosong
2. Cek antrian penuh
3. Kosongkan antrian
4. Tambah mahasiswa ke antrian
5. Panggil 2 mahasiswa untuk proses KRS
6. Tampilkan semua antrian
7. Tampilkan 2 antrian terdepan
8. Tampilkan antrian terakhir
9. Cetak jumlah antrian
10. Cetak jumlah yang sudah proses KRS
11. Cetak jumlah yang belum proses KRS
0. Keluar
Pilih menu: 7
1.
Nama : buye
NIM  : 3
Prodi: SIB
Kelas: 1E
2.
Nama : Bute
NIM  : 4
Prodi: SIB
Kelas: 1E
```

```
=== MENU ANTRIAN KRS ===
1. Cek antrian kosong
2. Cek antrian penuh
3. Kosongkan antrian
4. Tambah mahasiswa ke antrian
5. Panggil 2 mahasiswa untuk proses KRS
6. Tampilkan semua antrian
7. Tampilkan 2 antrian terdepan
8. Tampilkan antrian terakhir
9. Cetak jumlah antrian
10. Cetak jumlah yang sudah proses KRS
11. Cetak jumlah yang belum proses KRS
0. Keluar
Pilih menu: 8
Antrian terakhir:
Nama : Bute
NIM  : 4
Prodi: SIB
Kelas: 1E
```

```
=== MENU ANTRIAN KRS ===
1. Cek antrian kosong
2. Cek antrian penuh
3. Kosongkan antrian
4. Tambah mahasiswa ke antrian
5. Panggil 2 mahasiswa untuk proses KRS
6. Tampilkan semua antrian
7. Tampilkan 2 antrian terdepan
8. Tampilkan antrian terakhir
9. Cetak jumlah antrian
10. Cetak jumlah yang sudah proses KRS
11. Cetak jumlah yang belum proses KRS
0. Keluar
Pilih menu: 9
Jumlah mahasiswa dalam antrian: 2

=== MENU ANTRIAN KRS ===
1. Cek antrian kosong
2. Cek antrian penuh
3. Kosongkan antrian
4. Tambah mahasiswa ke antrian
5. Panggil 2 mahasiswa untuk proses KRS
6. Tampilkan semua antrian
7. Tampilkan 2 antrian terdepan
8. Tampilkan antrian terakhir
9. Cetak jumlah antrian
10. Cetak jumlah yang sudah proses KRS
11. Cetak jumlah yang belum proses KRS
0. Keluar
Pilih menu: 10
Jumlah mahasiswa yang sudah melakukan KRS: 2
```