# Løsningsforslag til ukesoppgaver, gruppetime 7 - Rekursjon

### Oppgave 1a - Løsningsforslag:

```
//Her har vi brukt overloading. Det er fordi vi begynner med hode-noden (første-noden),
men fra hovedprogrammet trenger vi ikke å vite dette.

  public void skriv_rek_forst() {
      skriv_rek_forst(hode);
  }

  public void skriv_rek_forst(Node denne) {
      if (denne == null){
          return;
      }
      System.out.println(denne.data);
      skriv_rek_forst(denne.neste);
  }

  public void skriv_rek_sist() {
      skriv_rek_sist(hode);
  }

  public void skriv_rek_sist(Node denne) {
      if (denne == null){
          return;
      }
      skriv_rek_sist(denne.neste);
      System.out.println(denne.data);
  }
```

### Oppgave 1b - Løsningsforslag:

```
  public int antNoder(){
       return antNoder(hode, 0);
  }

  public int antNoder(Node denne, int ant){
       if (denne == null){
       return ant;
       }
       return antNoder(denne.neste, ant+1);
  }
```

### Oppgave 1c - Løsningsforslag:

```
class TestFifoListe {
  //Teste Fifo-listen:
    public static void main(String[] args) {
       FIFOListe<String> liste = new FIFOListe<>();
       liste.leggTil("Hei");
       liste.leggTil("paa");
       liste.leggTil("deg");
       liste.leggTil("!");
       liste.skrivUtListe();
```

```
        System.out.println("Iterator-utskrift:");
        for(String t: liste){
           System.out.println(t);
        }

        liste.skriv_rek_sist();
        System.out.println();
        liste.skriv_rek_forst();

        System.out.println("ANT NODER: " + liste.antNoder());

    }
}
```

## Oppgave 2a - Løsningsforslag

```
class KaninOreTeller {
  //Teste Fifo-listen:
    public static void main(String[] args) {
        System.out.println(tellKaninOrer(0)); //Forventet resultat 0
        System.out.println(tellKaninOrer(1)); //Forventet resultat 2
        System.out.println(tellKaninOrer(2)); //Forventet resultat 4
        System.out.println(tellKaninOrer(12)); //Forventet resultat 24
        System.out.println(tellKaninOrer(234)); //Forventet resultat 486

    }

    public static int tellKaninOrer(int antallKaniner){
       if(antallKaniner == 0){
          return 0;
       }
       return tellKaninOrer(antallKaniner - 1) + 2;
    }
}
```

## Oppgave 2b - Løsningsforslag

```
class KaninOreTeller {
    public static void main(String[] args) {
        System.out.println(tellKaninOrer(0)); //Forventet resultat 0
        System.out.println(tellKaninOrer(1)); //Forventet resultat 2
        System.out.println(tellKaninOrer(2)); //Forventet resultat 4
        System.out.println(tellKaninOrer(12)); //Forventet resultat 24
        System.out.println(tellKaninOrer(234)); //Forventet resultat 486

    }

    public static int tellKaninOrer(int antallKaniner){
       if(antallKaniner == 0){
          return 0;
       }

       if(antallKaniner % 2 == 0){
          return tellKaninOrer(antallKaniner - 1) + 2;
       }
       return tellKaninOrer(antallKaniner - 1) + 1;
    }
}
```

## Oppgave 3a - Løsningsforslag

```
class StringOprasjoner {
    public static void main(String[] args) {
       String str = "kattkukatt"
        System.out.println(antallSubStringer(str, "katt")); //Forventet resultat 2
        System.out.println(antallSubStringer(str, "ku")); //Forventet resultat 1
        System.out.println(antallSubStringer(str, "hund")); //Forventet resultat 0

    }

    public static int antallSubStringer(String str, String sub){
       if(str.length() < sub.length()){
          return 0;
       }
       if(str.substring(0, sub.length()).equals(sub)){
          return 1 + antallSubStringer(str.substring(1, str.length()), sub);
       }
       return antallSubStringer(str.substring(1, str.length()), sub);
    }
}
```

## Oppgave 3b - Løsningsforslag

```
class StringOprasjoner {
    public static void main(String[] args) {
        System.out.println(settInnStjerner("hello")); //Forventet resultat hel*lo
        System.out.println(settInnStjerner("helllo")); //Forventet resultat hel*l*lo
        System.out.println(settInnStjerner("helloo")); //Forventet resultat hel*lo*o
        System.out.println(settInnStjerner("")); //Forventet resultat
        System.out.println(settInnStjerner("trollmannen")); //Forventet resultat
trol*lman*nen

    }

    public static String settInnStjerner(String str){
       if(str.length() < 2){
          return str;
       }
       if(str.charAt(0) == str.charAt(1)){
          return str.charAt(0) + "*" + settInnStjerner(str.substring(1, str.length()));
       }
       return str.charAt(0) + settInnStjerner(str.substring(1, str.length()));
}
```

## Oppgave 4 - Løsningsforslag

```
public static int factorialRecursive(int n) {
    if (n == 1) return 1;
    return n * factorialRecursive(n - 1);
}
```

La oss gå gjennom hva som skjer i den rekursive metoden `factorialRecursive` fra det øyeblikket den blir kalt til den terminerer, med `n = 5`:

1. **Første kall**:
    - Du kaller `factorialRecursive(5)`.

2. **Rekursivt kall (n != 1)**:
    - Metoden beregner fakultetet ved å multiplisere `n` med resultatet av `factorialRecursive(n - 1)`.
    - `factorialRecursive(5)` kaller `factorialRecursive(4)`.
3. **Gjentatte rekursive kall**:
    - Dette fortsetter:
        - `factorialRecursive(4)` kaller `factorialRecursive(3)`.
        - `factorialRecursive(3)` kaller `factorialRecursive(2)`.
        - `factorialRecursive(2)` kaller `factorialRecursive(1)`.
4. **Basistilfellet (n == 1)**:
    - Når `n` blir 1, når vi basistilfellet.
    - `factorialRecursive(1)` returnerer 1.
5. **Tilbakeføring (retur)**:
    - Nå klatrer vi tilbake:
        - `factorialRecursive(2)` returnerer `2 * 1 = 2`.
        - `factorialRecursive(3)` returnerer `3 * 2 = 6`.
        - `factorialRecursive(4)` returnerer `4 * 6 = 24`.
        - `factorialRecursive(5)` returnerer `5 * 24 = 120`.
6. **Metoden terminerer**:
    - Vi er tilbake på det opprinnelige kallet (`factorialRecursive(5)`).
    - Metoden returnerer fakultetet til tallet 5, som er 120 (`5 * 4 * 3 * 2 * 1`).

Rekursjonen bryter problemet ned i mindre delproblemer, og bygger opp løsningen ved å kombinere resultatene fra disse delproblemene.