Exercise 1: repetition. *Array-rotation*
Givet et array af længde n, skal der foretages k rotationer, således at elementer går "mod uret", eksempelvis for n=5 og k = 2:

```
[1, 2, 3, 4, 5] -> [3, 4, 5, 1, 2]
```

Input: en linje der indeholder 2 tal, n og k. Hvor n er antallet af elementer i arrayet og k er antallet af rotationer der foretages i arrayet. Efterfølgende gives der n tal, som er elementer i arrayet.
Output: Tallene i deres nye roterede rækkefølge.

1. Hvordan kan dette løses med 2 for-loops?
2. Kan de flyttede indekser udledes uden brug af det ekstra for-loop?

End of Exercise 1

Exercise 2: Rekursion. *Løses uden computer.*
Betragt følgende metode:

```
public static void mysteryI(int n){
        if(n <= 1){
                System.out.print(n);
        } else {
                mysteryI(n/2);
                System.out.print(", "+n);
        }
}
```

Hvad bliver skrevet til System.out ved følgende funktionskald?

1. mysteryI(1);
2. mysteryI(2);
3. mysteryI(3);
4. mysteryI(4);
5. mysteryI(16);
6. mysteryI(30);
7. mysteryI(100);

Hvis der kaldes mysteryI(n), kan du da beskrive antallet af printede tal som funktion af n?

End of Exercise 2

Exercise 3: Rekursion. *Løses uden computer.*
Betragt følgende metode:

```
public static int mysteryII(int x, int y){
  if(x<y){
    return x;
  } else {
    return mysteryII(x-y, y);
  }
}
```

Hvilken værdi bliver returneret ved disse metodekald?

1. mysteryII(6,13);
2. mysteryI(14,10);
3. mysteryI(37,10);
4. mysteryI(8,2);
5. mysteryI(50,7);

Kan du beskrive med ord eller formler, hvad programmet udregner, når $x \geq 0$ og $y \geq 0$? Er programmet robust?

End of Exercise 3

Exercise 4: Rekursion *Fibonacci med Memoisering.*
På CodeJudge findes skelletet til endnu en måde at løse Fibonacciproblemet på: memoisering.
Ideen er at gemme resultaterne efterhånden som de bliver beregnet, så intet tal bliver beregnet flere gange.
PS: Husk også at løse `fibII`, hvis ikke du har gjort det endnu, og husk at checke at din løsning er rekursiv (`fibHelper` kalder sig selv) og hurtig.

End of Exercise 4

Exercise 5: Rekursiv datastruktur: *Hægtet liste.* På CodeJudge ligger skelettet til en simpel hægtet liste. Udfyld resten og få programmet til at virke.

End of Exercise 5

Exercise 6: Rekursiv datastruktur: *Binært søgetræ.* På CodeJudge ligger skelettet til et binært søgetræ. Udfyld resten og få programmet til at virke.

End of Exercise 6

Exercise 7: *Tekstanalyse.* Prøv at bruge hægtet liste og/eller binært søgetræ til at tælle antallet af unikke ord i en tekst.

End of Exercise 7