SKRIV NAVN - PÅ ALLE SIDER:

PROCESSING:

1. Skriv en processing-kode, der opretter en variabel "navn", der indeholder dit navn.

   ```
   String navn = “Anders”;
   ```

2. Skriv processing-kode, der opretter et array med 100.000 forskellige heltal.

   ```
   int[] list = new int[100000];
   for(int i = 0; i < 100000 ; i++){
           list[i] = i;
   }
   ```

JAVASCRIPT OG HTML:

3. Byg en knap med HTML og JavaScript, der ved tryk giver en visuel effekt, f.eks. en tekst, der ændrer sig eller lignende.

   ```
   <button onclick=”alert(‘hej’)”>Knap<button>

   eller

   <script>function myFunction(){alert("!!!");}</script>
    <button onclick="myFunction()">Click me</button>
   ```

OOP PRINCIPPER:

4. **2 point**
   Skriv kode til følgende klasser: "Fugle", "And", "Måge" og "Hjerne", der anvender nedarvning og komposition til at "forbinde" klasserne på en hensigtsmæssig måde.

   ```
   class Fugl{Hjerne h = new Hjerne(); }
   class And extends Fugl{}
   class Måge extends Fugl{}
   class Hjerne{}
   ```

ALGORITMER:

5. **2 point**
   Bestem worst-case og best-case kørselstids-funktionen af array-sammenligninger for denne (ikke så smarte) algoritme:

   ```
   boolean toEns = false;
   for ( int i = 0 ; i < list.length ; i++ ) {
           for( int k = 0 ; k < list.length ; k++) {
                   if( k != i && list[i] == list[k] ) {
                           toEns = true;
                           break;
   } } }
   ```

Best case er hvis alle elementer i arrayet er ens - i så fald kører det inderste loop kun en gang for hver gang det ydre kører. Bemærk at der kun "breakes" ud af det inderste loop og ikke det ydre. Hvis alle elementer er ens breakes der hver gang.
Dvs. tiden er N sammenligninger

Worst case hvis alle elementer er forskellige kører der yderste loop N gange og det inderste loop kører N gange, hver gang det yderste kører:
Dvs. N^2 sammenligninger

6. **2 point**
   Kan du lave en hurtigere algoritme, end den ovenfor, der finder to ens tal i et array?

   Ja - en hurtigere algoritme kan laves ved at breake ud af begge loop.
   Man kan desuden erstatte k=0 med k=i

7. **2 point**
   Anvend binær søgning til at søge efter tallet 4 i denne liste af tal. Hvor mange forsøg koster det før tallet er fundet? Og hvilke tal finder algoritmen i sin søgning?
   [1, 2, 4, 5, 10, 23, 33, 45, 90, 91, 100, 120, 130]
   Ved en binær-søgning der runder index op ved hvert gæt
   33, 5, 2, 4 … 4 forsøg
   Hvis den runder op:
   33, 4 … 2 forsøg

8. **2 point**
   Tegn et klassisk binært søgetræ ud fra følgende sekvens af tal:
   [10, 6, 16, 3, 9, 13, 19]

9. Lav sekvens af samme tal som i opgave 8, der giver det "dårligst" mulige søgetræ.
   19 , 16 , 13, 10, 9 , 6, 3

10. Vurder hvor hurtigt man kan søge i træerne fra opgave 8 og 9
    Den ene køre hastigheden N den anden log2(N)

I alt 15 point - opgaverne 4,5,6,7 og 8 giver 2 point