Prøveeksamen til kurset “Grundlæggende Datalogi”

Morten Misfeldt, tlf: +45 xx xx xx xx, Ken Friis Larsen tlf: +45 xx xx xx xx

Prøven er uden hjælpemidler

Question 1: Datatyper (5 pts). Hvad skriver det her program til konsolen? Forklar hvorfor.

```
console.log("3"+"4");
console.log(3+4);
```

Question 2: Løkker (5 pts). Hvad skriver det her program til konsolen? Forklar hvorfor.

```
let sum = 0;
    for (let i = 1; i <= 7; i++) {
        sum += i;
        console.log("Sum after adding " + i + ": " + sum);
    }
```

Question 3: Funktioner og løkker (10 pts). Forklar hvad en funktion er, og vis hvordan en funktion skrives i JavaScript.

Skriv en funktion i JavaScript `howManyA(text)`, der tager en streng som input og returnerer antallet af "a“er. Forklar din strategi.

Question 4: Forklar og modificer programmet (10 pts). Forklar hvad følgende program gør (5 pts):

```
function greetUser() {
    let userName = prompt("Please enter your name:");
    if (userName) {
        alert("Hello, " + userName + "!");
    } else {
        alert("Hello, stranger!");
    }
}
```

Hvordan kan du udvide programmet, så det også spørger efter alder og bruger alderen til at give en mere personlig hilsen (5 pts)?

Question 5: HTML (10 pts). For hver af de 5 gange der står "what is this # x", skriv din forklaring på hvad der sker på dette sted i koden.

```
<!DOCTYPE html>
<html>
<head>
     <title>what is this # 1</title>
</head>
<body>
    <h1>what is this # 2</h1>

    <p>what is this # 3</p>

    <a href="https://www.example.com"> what is this # 4</a>

    <script>
      alert( "what is this # 5");
    </script>
</body>
</html>
```

Question 6: CSS (5 pts). Forklar hvad der menes med ”specificity” i CSS og skriv et kort eksempel på nogle CSS rules, der illustrerer princippet.

Question 7: HTML ellementer (5 pts). Forklar forskellen på semantiske og ikke-semantiske HTML-elementer (giv et eksempel på hver).

Question 8: Find fejlen (5 pts). Funktionen `multTable` nedenfor er beregnet til at implementere og returnere en multiplikationstabel, dvs. en $m \times n$ matrix, der ved række i og kolonne j indeholder værdien $(i+1) * (j+1)$. Koden indeholder dog en række fejl; find og forklar kort mindst **tre** af dem. (Fejlene kan identificeres selv uden at forstå hvad `multTable` gør).

```
function multTable(m, n) {
    let matrix = [];

    for(let i = 0; i < m; i = i+1) {
        let matrix = new Array(m);

        for(let j = 0; j < n; j=j+1) {
            let row = [];

            let ij = (i+1) * (j+1);

            push.row( ij );
            matrix{^i^} = row;
        }
    }

    return matrix;
}
```

Question 9: Hvad skriver programmet til konsolen? (3 pts)
Hint: `split` er en metode der virker på en streng ved at dele strengen i et array af ord, i forhold til det/de tegn der tages som argument. I dette tilfælde deler split i første omgang ved dobbelt bindestreg `"--"` og i anden omgang ved underscore `"_"`.

```
let str = "I_can_split--that_string_into--a_table_if_needed";

let arr_of_strs = str.split("--");

let table = arr_of_strs.map( str => str.split("_") );

console.log( table[2][1] + " " + table[1][0] + " " + table[0][2] + " " + table[1][1] );
```

Question 10(a): Implementer løkken (3 pts). Afslut implementeringen af funktionen nedenfor, så den returnerer indekset for det minimale element i et input-array, der indeholder positive heltal, og -1", hvis arrayet er tomt. Du skal implementere indholdet af løkken nedenfor; implementeringsplanen er givet som kommentarer direkte i koden.

```
function indexOfMinInArray( input_array ) {
    let min_ind = -1;
    let min_val = -1;

    for(let i=0; i<input_array.length; i=i+1) {
        // 1. get the current element of the input array




        // 2. if `min_val` is greater than the current element then:
        //      2.1 set `min_ind` to the value of the loop index
        //      2.2 set `min_val` to the value of the current element









    }
    return min_ind;
}
```

Question 10(b): Kompleksitet (3 pts). Angiv og forklar kort kompleksiteten af din implementering for et input-array af længde n. (Er den logaritmisk $O(\ log(n)\ )$, lineær $O(n)$, $O(\ n \cdot log(n))$, kvadratisk $O(n^2)$ eller eksponentiel $O(2^n)$?)

Question 11(a): max (3 pts). Implementer indmaden af funktionen `max` nedenfor, der skal at returnere det største af dens to argumenter `a` og `b`.

```
function max(a, b) {




}
```

Question 11(b): Hvad skrives til konsolen? (3 pts) Hvis vi antager, at funktionen `max` er korrekt implementeret til at returnere det største af dets to argumenter, hvad udskriver koden nedenfor så?

```
let x = [5, 1000000, 21, 45, 11, 75, 3];
let y = x.reduce(max, Infinity);
console.log(y);
```

Question 11(c): Map-Reduce-Filter (5 pts). Implementer — ved at bruge split, map, reduce og filter — funktionen nedenfor, hvis argument er en streng `str`. Funktionen skal

(1) split strengen i et array af strenge ved at bruge "."som separator.

(2) behold kun de strenge fra dette array, hvis længde er mindre end 20.

(3) transformerer hvert strengelement i arrayet opnået i trin (2) til et tal svarende til længden af strengelementet.

(4) beregn (og returner) det maksimale antal af arrayet opnået i trin (3)

```
function maxLengthLessTh20( str ) {
    // 1. create an array of strings by spliting `str` with "." as separator.
    let step1_array = ...

    // 2. create an array that constains only the string elements
    //      of `step1_array` whose length is less than 20.
    let step2_array = ...



    // 3. create an array by transforming each of the string elements of
    //      `step2_array` to its length, i.e., an array of numbers.
    let step3_array = ...



    // 4. compute the maximal element of the array `step3_array`
    let step4_max_num = ...



    return step4_max_num;
}
```

Question 12(a): Objekter (5 pts) Ved øvelserne har vi arbejdet (som opvarmning) med et array `Course` objekter. Nedenfor kan du finde definitionen af klassen `Course` (som har en `prettyPrint` metode nu), og koden til at oprette og indsætte tre kursusobjekter i et array af kurser.

```
class Course {
  name;     // a string
  students; // an array
            //   of strings
  constructor(nm, stds) {
    this.name = nm;
    this.students = stds;
  }
  prettyPrint() {
    let str =
      "The "+this.name +
      " course is taken by: " +
      this.students.toString();
    return str;
  }
}
```

```
let array_of_courses = [];
{
  let cbio =
        new Course("biology",   ["Anne", "Mette"]  );

  let clit =
        new Course("literature",["Anne", "Peter"]  );

  let cphy =
        new Course("physics",   ["Mette", "Peter"] );

  array_of_courses.push(cbio);
  array_of_courses.push(clit);
  array_of_courses.push(cphy);
}
let n = array_of_courses.length;
```

Forudsat at denne kode køres i konsollen, hvad udskrives i hvert af de tre tilfælde (1), (2) og (3). Hint: `n` er længden af arrayet af kurset efter indsættelse af de tre kurser.

```
// (1) What does this print?
console.log( array_of_courses[n-1].students[1] );
```

```
// (2) What does this print?
console.log( array_of_courses[0].prettyPrint() );
```

```
// (3) What does this print?
let arr_of_strs = array_of_courses.map( s => s.name.toLowerCase() );
console.log( arr_of_strs.toString() );
```

Question 12(b): forklar objekter (2 pts) Når du kalder `new Course("biology", ["Anne", "Mette"])` bedes du kort forklare mekanismen, hvorved felterne/egenskaberne for kursusobjektet initialiseres.

Question 13: Recursion (6 pts) Implementer den **rekursive** funktion GCD, der implementerer den største fælles divisor (GCD) algoritme ifølge den rekursive definition:

$$\mathbf{GCD}(a,b) = \begin{cases} a & \text{if } b = 0 \\ \mathbf{GCD}(\ b,\ a\%b\ ) & \text{otherwise} \end{cases}$$

Det betyder, at GCD tager to positive heltal a og b som argumenter og beregner følgende:

- hvis b er nul, så skal GCD(a, b) returnere a (base case)
- Ellers returneres GCD(b, a % b), hvor a % b beregner resten ved heltaldivisionen og skrives ens i Javascript.

(Selvom det ikke er vigtigt for øvelsen, beregner GCD(a, b) det største tal, der går op i både a og b.)

```
function GCD(a, b) {
    // please implement me (in maximum 5 lines)




}
```

Question 14: Tegn træet (6 pts) Tegn det binære træ skabt af koden nedenfor — for eksempel ved at repræsentere en node som en cirkel, der indeholder tallet gemt i dets datafelt.

```
class BinTree {
    data;  // the data stored
           //    in this node
    left;  // the  left subtree
    right; // the right subtree

    constructor(d, l, r) {
        this.data = d;
        this.left = l;
        this.right = r;
    }
}
```

```
// this is the root
let root = new BinTree(100, null, null);

let n1   = new BinTree(25, null, null);
n1.right = new BinTree(200, null, null);
n1.left  = new BinTree(400, null, null);

root.left = n1;

root.right = new BinTree(75, null, null);
root.right.left  = new BinTree(10, null, null);
root.right.right = new BinTree(50, null, null);
```

Question 15: Rekursion på træer (6 pts). Funktionen nedenfor tæller antallet af noder i et binært træ — som er implementeret af klassen `BinTree` i din tidligere opgave.

```
function countTreeNodes(node) {
    if (node === null) {  //base case
        return 0;
    }

    let num_left  = countTreeNodes(node.left);
    let num_right = countTreeNodes(node.right);

    return (1 + num_left + num_right);
}
```

Omskriv funktionen til at beregne antallet af blade i træet, dvs. de noder, der har både venstre og højre børn lig med `null`. Ideen er kun at tælle op med `1` (til `num_left+num_right`), hvis den aktuelle node har både venstre og højre børn `null`, og ellers at tælle op med `0`.