***Belangrijk**: Schrijf je antwoorden kort en bondig in de daartoe voorziene velden. Indien nodig gebruik je de achterkant van het blad (met duidelijke vermelding van de vraag). Het gebruik van een eenvoudig rekenmachine is toegestaan. Andere hulpmiddelen (smartphone, laptop, cursusnota's, ...) zijn niet toegestaan tijdens het examen.*
Elke theorie-vraag staat op 2 punten (totaal op 24). De oefeningen staan in totaal op 6 punten. Het geheel staat op 30 punten.

1. Introduction [... / 2]

   Waarom is het "*waterval*" model onrealistisch? (3 antwoorden)

   - *complete: the customer is forced to define all the requirements soon and very explicitly (very difficult and in practice not feasable), future changes on system will be more difficult.*
   - *time: only at the very end of development we have a working version of the product. This may cause the customer to change his mind (no feedbacks in the process, customers may not agree on the late delivery or worse they had another product in mind)*
   - *idealistic: in real projects iteration occurs.*
   - *change: difficult and costly to adapt later steps when requirements change.*

   Waarom wordt het nog steeds gebruikt?

   Popular because it is visible for upper management, easy to control project progress

2. Project Management [... / 2]

   Som de vijf project management functies op, en beschrijf elke functie kort.

   1) Planning: opdelen in taken, en resources inplannen
   2) Organizeren: taken toedelen aan personen
   3) Staffing: recruteren en personeel motiveren
   4) Directing: verzekeren dat het team als een geheel werkt
   5) Monitoring: afwijkingen t.o.v. het plan opmerken en correctieve acties ondernemen

3. Use Cases [... / 2]

   Noem de twee regels die strikt gevolgd moeten worden bij "*Project Plan Negotiations"* en motiveer elke regel.

   ontwikkelaars schatten kosten, klanten moeien niet [0,5]
   => problemen in planning zijn verantwoordelijkheid van ontwikkelaars [0,5]
   klanten geven prioriteiten, ontwikkelaars moeien niet [0,5]
   => geld uitgeven is verantwoordelijkheid van klant [0,5]

4. Domain Models [... / 2]

Benoem drie technieken om verantwoordelijkheden te identificeren in het kader van CRC cards

1) rollenspel
2) identificatie van zinnen met nadruk op werkwoorden
3) enumeratie van kandidaat-klassen

5. Testing [... / 2]

Leg telkens uit wat "*basic path testing*", "*condition testing*" en "*loop testing*" is. Waarom zijn deze technieken complementair?

basic path testing tests all possible paths ***by total coverage of every statement, branch****.*
condition testing tests all ***conditions by making them true or false****. (not all these possibilities were done by basic path testing)*
loop testing tests every loop by passing it (n is number of allowable passes)

0, 1, 2,
m passes with 2 < m < n
n - 1, n, n +1 passes
(not all these possibilities were done by basic path testing and condition testing)

each technique tests a different aspect since not all possibilities were done by the previous technique(s) => complementary

6. Design by Contract [... / 2]

Wat is het onderscheid tussen "*Design by Contract*" en "*Testing*"?

Design by contract prevents defects [0.5 pnt]
Testing detects defects [0.5 pnt]

Geef twee redenen waarom dit complementaire technieken zijn:

- *[0.5 pnt] Design by contract can be used in Equivalence Partitioning & Boundary Value Analysis (Black box testing) to determine classes of input data & prediction of corresponding output.*
- *[0.5 pnt] (Condition) Testing used to verify whether parties satisfy their pre- & postconditions (design by contract)*

7. Formal Specifications [... / 2]

Wat is het verband tussen "*Clean Room Development*" en "*Formele Specificaties*"?

Cleanroom is voornamelijk gebaseerd op formele specificaties (aangevuld met testen).

Waarom is het noodzakelijk om sequence diagrams aan te vullen met state charts?

een state chart is een specificatie van alle mogelijke en onmogelijke scenarios (sequences)

8. Software Architecture [... / 2]

Wat is "*een pattern*"?

the essence of a solution to a recurring problem in a particular context [1]

Geef 2 redenen waarom dit bruikbaar is om een architectuur te beschrijven:

1) [0.5 pnt] Experts recall a similar solved problem and customize the solution
2) [0.5 pnt] Patterns document existing experience
3) [0.5 pnt] The context of a pattern states when and when not to apply the solution
4) [0.5 pnt] A pattern lists the tradeoff's involved in applying the solution

9. Quality Control [... / 2]

Bespreek kort drie vereisten voor een kwaliteitsplan. Een kwaliteitsplan moet:

1) gewenste produkt-kwaliteiten aangeven en bespreken hoe deze beoordeeld worden;
2) toe te passen organisatorische standaarden aangeven
3) het kwaliteitsassessment proces definieren

10. Software Metrics [... / 2]

Wanneer voldoet een coupling metriek niet aan de "*representation condition*"?

Geef ook een voorbeeld.

If the metric doesn't measure what we understand as coupling.
vb1: high LCOM value (high cohesion) by accessor methods
vb2: classes with low coupling but nevertheless high CBO value, because there is no difference between data, method or inheritance coupling

11. Refactoring [... / 2]

Geef 4 code smells die met refactoring kunnen opgelost worden.

- *Duplicated code*
- *Nested conditionals*
- *Large classes/methods*
- *Abusive inheritance*

12. Conclusion [... / 2]

Als je het "No Silver Bullet" artikel hebt gelezen:

Waarom is programma verificatie geen silver bullet?

Programma verificatie doet iets aan de accidentele complexiteit : nagaan of een programma voldoet aan zijn specificatie. Maar de specificatie kan fouten bevatten en bovendien nog incompleet zijn. De essentiële complexiteit (= hetgeen inherent moeilijk is aan het bouwen van software) is net het opstellen van een complete en consistente specificatie.

Als je het “Killer Robot” artikel hebt gelezen:

Waarom was in dit specifieke geval het “*waterfall process*” zo rampzalig?

(From KillerRobotCase/articel-4.html) The waterfall model goes through definite stages of development. As the project passes from one stage to the next, there are limited opportunities to change earlier decisions. A drawback of this approach is that potential users are not able to interact iwth the sytem until very late in the process.

The Robot project involves a high degree of interaction, both between the robot components and between the robot and the operator. Since operator interaction with the robot is so important, the interface cannot be designed as an afterthought.

13. Oefeningen – Testing [... / 5]

Beschouw de onderstaande functie die een array *data* sorteert door gebruik te maken van het Mergesort algoritme.

```
1  void mergeSort(int[] data) {
2     if(data.length <= 1)
3         return;
4
5     int[] a = new int[data.length / 2];
6     int[] b = new int[data.length - a.length];
7     for(int i = 0; i < data.length; i++) {
8         if(i < a.length)
9             a[i] = data[i];
10        else
11            b[i - a.length] = data[i];
12    }
13
14    mergeSort(a);
15    mergeSort(b);
16
17    int ai = 0;
18    int bi = 0;
19    while(ai + bi < data.length) {
20        if(bi >= b.length||(ai < a.length && a[ai] < b[bi])) {
21            data[ai + bi] = a[ai];
22            ai++;
23        } else {
24            data[ai + bi] = b[bi];
25            bi++;
26        }
27    }
28 }
```

a) Teken de *flow graph* voor bovenstaande functie.

b) Bereken de *cyclomatische complexiteit* en geef kort aan hoe je hiertoe gekomen bent.

…….

…….

…….

c) Hoe verhoudt het aantal onafhankelijke paden zich tot de *cyclomatische complexiteit*?

…….

…….

…….

d) Bepaal een volledige verzameling *onafhankelijke paden*. (Nummer ze).

…….

…….

…….

…….

…….

…….

…….

…….

…….

e) Bepaal voor elk pad een test case (input en verwachte output) om dit onafhankelijk pad te kunnen uitvoeren. Gebruik dezelfde nummering als in de vorige oefening om te verwijzen naar een pad.

…….

…….

…….

…….

…….

…….

…….

…….

…….