***Belangrijk**: Schrijf je antwoorden kort en bondig in de daartoe voorziene velden. Elke theorie-vraag staat op 2 of 3 punten (totaal op 30). Elke oefening staat op 5 of 10 punten (totaal op 25). Het geheel staat op 55 punten.*

Vraag 1. [..../3]

Wat is:

a) *Iterative Development*: gecontroleerd herwerken van een deel van het systeem...
om verbeteringen aan te brengen =>wrong before right [1].....

b) *Incremental Development*: vooruitgang in kleine stappen, early results.....
=> always have running version [1].....

Wat is hun verband met *Evolutionary Prototyping*?

combineert beide [1].....

.....

Vraag 2. [..../3]

Wat is het verschil tussen de *0/100*, de *50/50* en de *milestone* techniek om de *Earned Value* te berekenen?

0/100: 0 when started, 100 when complete [1].....

.....

50/50: 50 when starrted, another 50 when completed [1].....

.....

milestone: #milestones completed / total #milestones [1].....

.....

Vraag 3. [..../2]

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

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

Vraag 4. [..../2]

Waarom is het noodzakelijk om de requirements te *valideren* én te *analyseren*?

valideren: zijn we het juiste systeem aan het bouwen [1].....

analyseren: modeleren we het probleem domein juist [1].....

Vraag 5. [..../3]

Definieer:

a) *Testing*: programma uitvoeren met de bedoeling *defects* te vinden [1].....

.....

b) *Testing Techniques*: technieken met een hoge kans om **nieuwe** fouten.....

te vinden [1].....

c) *Testing Strategies*: plan dat zegt wanneer je welke techniek moet toepassen [1]..

.....

Vraag 6. [..../2]

Verklaar het *substitutie principe*.

je mag een instantie van een superklasse vervangen.....

door elke van zijn subklasses [1].....

Waarom is dit belangrijk in *OO development* en voor *contracten*?

Het legt de regels vast voor overerving en dus voor subcontracten [1].....

.....

Vraag 7. [..../3]

Wat betekent het voor een *state-based* specificatie om *consistent*, *complete* en *unambiguous* te zijn?

a) *Consistent*: elke state is bereikbaar vanuit de initial state + vanuit elke staat.....

is de eindstaat bereikbaar [1].....

b) *Complete*: elk event/state koppel heeft een transitie [1].....

.....

c) *Unambiguous*: zelfde event (**incl guard** [-0,5]) komt niet voor op meer dan .....

1 transitie uit een zelfde state [1].....

Vraag 8. [..../2]

Geef twee redenen waarom je **geen** *adapter/wrapper* zou introduceren in een design.

1. twee uit: performance overhead, maintenance overhead,.....
2. whole hierarchy needs adapting, merging in two directions needed.....

Vraag 9. [..../3]

Wat is het verschil tussen *product* en *process quality attributes*?

kwaliteit op het product geleverd aan de klant [0,5].....

kwaliteit op het proces om dat product te maken [0,5].....

Wat is het verschil tussen *internal* en *external quality attributes*? Wat is het verschil in vereiste op het systeem of het proces om elk attribuut te kunnen meten?

external: relatie tussen omgeving en het systeem/proces [0,5].....

=> systeem/proces moet draaien [0,5].....

internal: rechtstreeks afgeleid van product of proces [0,5].....

=> beschrijving van systeem is genoeg [0,5].....

Vraag 10. [..../2]

Wanneer voldoet een *coupling* metriek **niet** aan de *representation condition*? Geef ook een voorbeeld.

Als de metriek niet meet wat wij verstaan onder *coupling* [1].....

vb1: hoge LCOM waarde (dus hoge cohesie) door accessor methods [1].....

vb2: klasses met lage coupling maar toch hoge CBO waarde, omdat er geen.....

onderscheid is tussen data, method of inheritance coupling.....

Vraag 11. [..../3]

Noem de drie fasen van de *iterative development life-cycle*.

1. prototyping [0,5].....
2. expansion [0,5].....
3. consolidation [0,5].....

Welke van de drie is het meest geschikt voor *refactoring*? Motiveer.

consolidation [0,5].....

want refactoring is gedragsbehoudend [1].....

Vraag 12. [..../2]

Noem twee belangrijke ethische regels en verklaar ze kort.

1. [0,5].....: [0,5].....
   .....
   .....
2. [0,5].....: [0,5].....
   .....
   .....

Vraag 13. [..../5]

Gegeven de *Pert-chart* op de volgende pagina:

a) Vul voor elke node in de *Pert-chart* de *earliest start date* in (uitgedrukt in weeknummers).[1], een fout is 0

Beschrijf kort hoe je dit berekend hebt.

ESD(start node) := start date project.....

breadth-frist for each task n:.....

ESD (n) := latest of (ESD (preceding) + estimated time (preceding)) [1].....

.....

b) Vul voor elke node in de *Pert-chart* de *latest end date* in (uitgedrukt in weeknummers).[1], een fout is 0

Beschrijf kort hoe je dit berekend hebt.

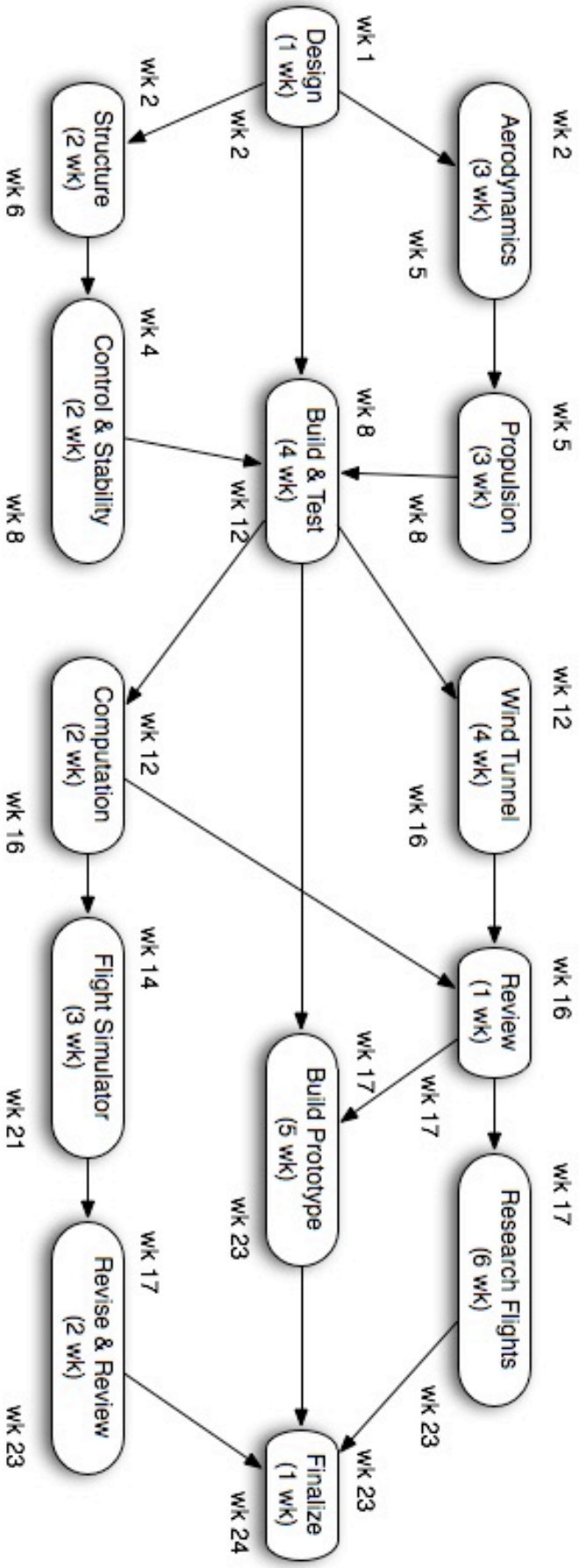
LED(end node) := ESD(end-node) + estimated time.....

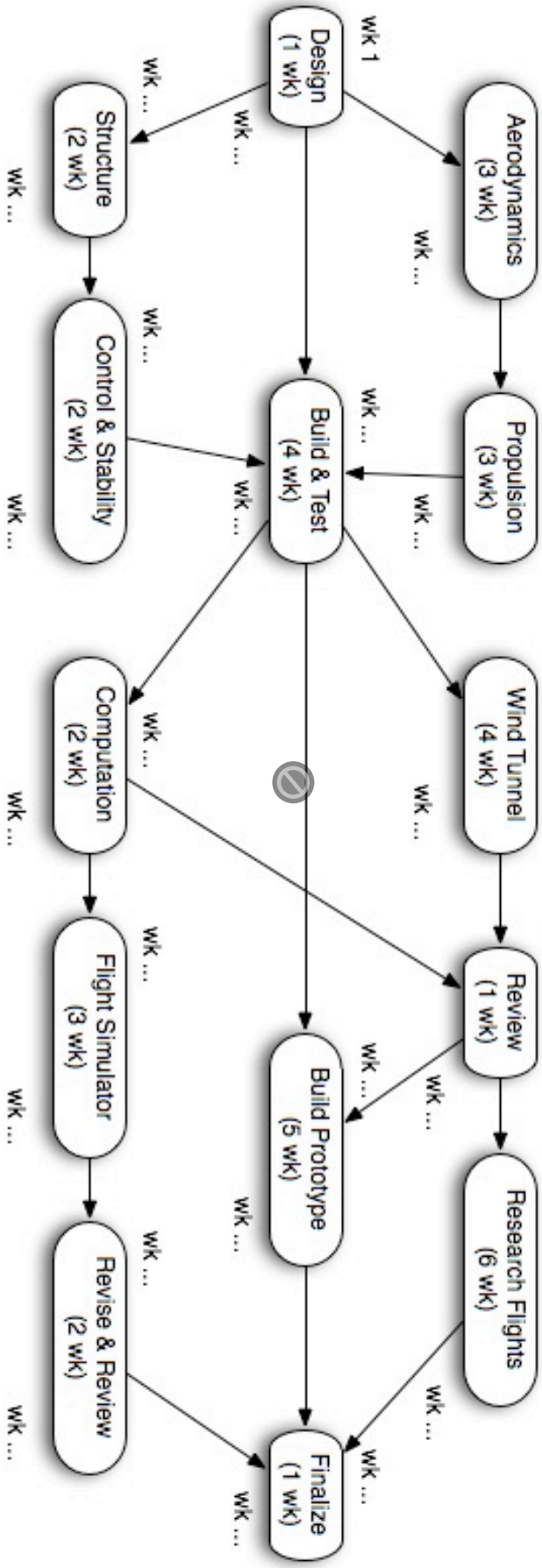
breadth-frist reverse for each task n:.....

LED (n):= earliest of (LED (subsequent) - estimated time (subsequent)) [1].....

.....

c) Duid op de figuur het *critical path* aan. [1]


Design
(1 wk)
wk 1
wk 2
wk 2
Aerodynamics
(3 wk)
wk 2
wk 5
Structure
(2 wk)
wk 6
Propulsion
(3 wk)
wk 5
wk 8
Control & Stability
(2 wk)
wk 8
wk 4
Build & Test
(4 wk)
wk 8
wk 12
Wind Tunnel
(4 wk)
wk 12
wk 16
Computation
(2 wk)
wk 16
wk 12
Review
(1 wk)
wk 16
wk 17
Flight Simulator
(3 wk)
wk 21
wk 14
Build Prototype
(5 wk)
wk 23
wk 17
Research Flights
(6 wk)
wk 23
wk 17
Revise & Review
(2 wk)
wk 23
wk 17
Finalize
(1 wk)
wk 24
wk 23


Design
(1 wk)
wk 1
Aerodynamics
(3 wk)
Structure
(2 wk)
Propulsion
(3 wk)
Control & Stability
(2 wk)
Build & Test
(4 wk)
Wind Tunnel
(4 wk)
Computation
(2 wk)
Review
(1 wk)
Flight Simulator
(3 wk)
Build Prototype
(5 wk)
Research Flights
(6 wk)
Revise & Review
(2 wk)
Finalize
(1 wk)
wk ...

Vraag 14. [..../10]

Het kortstepadalgoritme van Dijkstra is jullie wellicht bekend. Dit graaf-algoritme berekent, gegeven een gerichte graaf *Graph* waarin de afstand tussen ieder tweetal verbonden knopen van *Graph* ten minste 0 bedraagt, voor een bepaalde beginknoop *source* in *Graph,* de kortste afstand van die beginknoop tot alle punten van de graaf.

Hieronder vinden jullie het algoritme in pseudo-code.

```
function Dijkstra(Graph, source):
  if Graph has less than 2 vertices or
    source vertex is not in Graph:
      return

  for each vertex v in Graph: //Initializations
    dist[v] := infinity       //Unknown distance function from source to v
    previous[v] := undefined  //Previous vertex in optimal path from source

  dist[source] := 0           //Distance from source to source
  Q := the set of all vertices in Graph //All vertices in the graph are
                                        //unoptimized - thus are in Q
  while Q is not empty:       //The main loop
    u := vertex in Q with smallest dist[]
    remove u from Q
    for each neighbor v of u: //where v has not yet been removed from Q
        alt := dist[u] + dist_between(u, v)  //be careful, in 1st step
                                             //dist[u] is still infinity
        if alt < dist[v]      //Relax (u,v)
            dist[v] := alt
            previous[v] := u
  return previous[]
```

Opgave

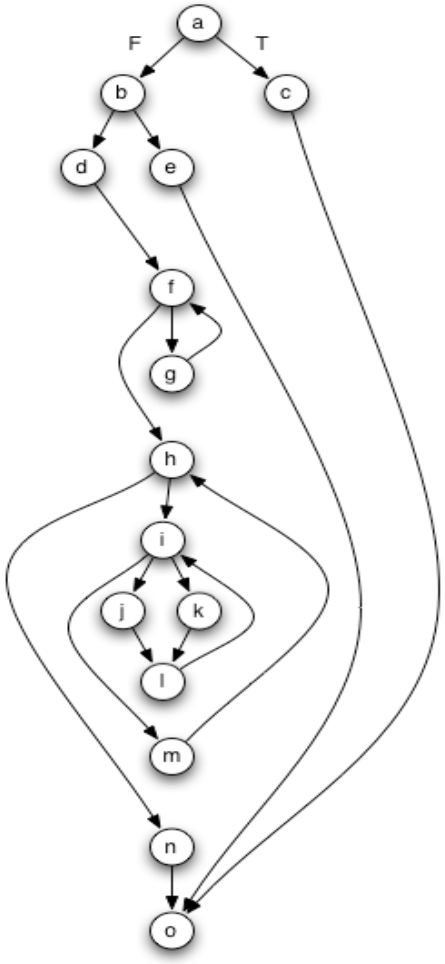
a) Teken de *control flow graph* voor bovenstaande functie.

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

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

d) Geef aan welke *input* vereist is voor elk van de *onafhankelijke paden* die je in c) hebt opgesomd. (Gebruik de nummering voor verwijzingen naar elk onafhankelijk pad.)

e) Hoe verhoudt het aantal paden in deze verzameling zich tot de *cyclomatische complexiteit*? Waarom?



Extra pagina voor vervolg oplossing vraag 13.

a) [4]

concept loops en condities correct?

dubbele conditie a-b-c-d-e correct?

return

true/false en richting van pijlen geannoteerd?

b) [2]

#e - #n + 2 = 20 – 15 +2 = 7

aantal binaire condities + 1 = 7

aantal graafregios = 7

c) [1]

1. a-c
2. a-b-e
3. a-b-d-f-g-h-i-j-l-i-m-h-n-o
4. a-b-d-f-g-h-i-k-l-i-m-h-n-o

d) [1]

Graph met < 2 knopen

source is geen knoop van de graaf

Graph waar nieuwe afstand groter is dan eerdere afstand (Graph met minstens drie elementen nodig)

elke Graph met minstens 2 knopen: initieel staat afstand op oneindig, nieuwe afstand is sowieso kleiner.

e) [2]

aantal paden dat nodig is om alle knopen te doorlopen is maximaal de CC. Anderzijds kan je door het kiezen van de input data sommige paden combineren. Het minimale aantal hier lijkt 3 aangezien je paden 3 en 4 kan combineren.

Extra pagina.

Vraag 15. [..../10]

In deze opdracht bekijken we de *requirements* voor de software van een bepaald soort bankautomaat, en schatten we met behulp van *Function Points* het verwachte aantal lijnen code af. Er bestaat uiteraard geen unieke oplossing, vandaar dat we jullie vragen om de berekeningen stap voor stap te maken.

Requirements

De bankautomaat bestaat uit volgende (hardware) onderdelen:

- een lezer voor magnetische kaarten,
- een console waarmee de interactie met de klant tot stand komt,
- een opening om enveloppen in te deponeren,
- een verdeler van cash,
- een printer om ontvangstbewijzen uit te printen,
- een communicatielijn met de centrale server van de bank

De bankautomaat zou de volgende functionaliteit moeten aanbieden:

- De klant moet zijn bankkaart kunnen aanbieden, waarbij gevraagd wordt naar de pincode. Zowel de kaartnummer als de pincode worden vervolgens naar de bank verzonden voor validatie. Indien gevalideerd moet de klant de mogelijkheid hebben 1 of meerdere transacties uit te voeren.
- De bankautomaat zal interne logs van de transacties bijhouden. Elk bericht dat naar de bank verzonden wordt, het eventuele antwoord van de bank, het opvragen van cash geld en de ontvangst van een enveloppe wordt gerapporteerd. Rapporteringen kunnen kaart nummers en geldhoeveelheden omvatten, maar om veiligheidsredenen nooit een PIN code.
- De kaart blijft in de kaartlezer tot wanneer de klant
  - de “Cancel” knop op de console gebruikt, waardoor een eventuele transactie in uitvoering geannuleerd wordt,
  - of de softwarematige afsluitprocedure volgt. In beide gevallen wordt de kaart teruggegeven.
- De klant moet geld kunnen afhalen van elke account die gelinkt is aan de kaart. De bank moet hiervoor toestemming geven. De bankautomaat houdt bij hoeveel bankbriefjes hij van welke soort voorradig heeft.
- De klant moet cash geld kunnen storten op elke rekening die gelinkt is aan de kaart. De klant moet aangeven op welke rekening hij het geld wilt storten, hij moet het bedrag van de storting ingeven en vervolgens de enveloppe in de daar voor bestemde opening deponeren. De inhoud van de enveloppe zal manueel geverifieerd worden door de operator. Toestemming van de bank moet verkregen worden vooraleer de enveloppe aanvaard wordt.
- Een klant moet geld kunnen overschrijven tussen 2 van de aan de kaart gelinkte accounts.
- Een klant moet de mogelijkheid hebben om de stand van zijn bankaccounts op te vragen.
- Als de bank vast stelt dat de PIN code die door de klant werd ingegeven niet correct is, dan zal de klant opnieuw de PIN code moeten ingeven. Na 3 pogingen wordt de kaart achter gehouden door de bankautomaat.

- Als een transactie faalt voor een andere reden dan een foute PIN code, dan zal de ATM een verklaring voor het probleem melden aan de klant, en vragen aan of hij/zij een andere transactie wil starten.
- De bankautomaat zal voor elke succesvolle transactie een ontvangstbewijs afdrukken. Datum, tijd, plaats van de machine, transactietype, rekeningnummer, hoeveelheid, eindstand en beschikbare hoeveelheid van de rekening.

Opgave *(antwoorden op volgende pagina, behalve vraag c)*

a) Identificeer volgende programma karakteristieken. Maak een lijstje per karakteristiek.

   a. External Inputs
   b. External Outputs
   c. Inquiries
   d. External Interfaces
   e. Logical Files

b) Bereken de *unadjusted function points*, gegeven de volgende gewichten: External Inputs: 3, External Outputs: 4, Inquiries: 3, External Interfaces: 5, Logical Files: 7.

c) Geeft elk van onderstaande complexiteitsfactoren een *rating* van 0 t.e.m. 5. Bereken hiermee de *Value Adjustement Factor* (VAF).

```
Performance Objectives         rating ... x weight 1   = ....
End-use Efficiency             rating ... x weight 1   = ....
Complex Internal Processing    rating ... x weight 1   = ....
Code Must Be Reusable          rating ... x weight 1   = ....
Easy To Install                rating ... x weight 0.5 = ....
Easy To Use                    rating ... x weight 0.5 = ....
Portable                       rating ... x weight 2   = ....
Easy To Change                 rating ... x weight 1   = ....
Special Security               rating ... x weight 1   = ....
Direct Access for 3rd Parties  rating ... x weight 1   = ....
-------------------------------------------------------------
Totaal:                                                  ....
```

d) Bereken de *Adjusted Function Points*.

e) Hoeveel lijnen code schat je dat de software voor de bankautomaat zal omvatten indien je het zou implementeren in Java (#LOC/FP = 46)?

Extra pagina voor oplossing vraag 15.

a) [5]

a. Internal Logical Files
   - Logs
   - Bestand met voorraad aan cash

b. External Interfaces
   - Server van de bank
   - Bankkaart

c. External Inputs
   - Bankkaart aanbieden + PINCODE vragen en valideren
   - PINCODE vragen en valideren
   - Gebruik van CANCEL-knop + Normale afsluitprocedure
   - Normale afsluitprocedure
   - Geld afhalen
   - Geld storten
   - Geld overschrijven

d. External Outputs
   - Foute PIN-code
   - Afdrukken van ontvangstbewijs
   - Opgevraagd geld aan klant geven

e. Inquiries
   - Stand opvragen

b) [1]

External inputs: [5,7] X 3 = [15,21]

External outputs: [2,3] X 4 = [8,12]

Inquiries: 1 X 3 = 3

External Interfaces: [1,2] X 5 = [5,10]

Logical Files: 2 X 7 = 14

=> Totaal: [45,60] unadjusted function points

c) [2,5]

Performance Objectives: [2..4] x 1 = [2..4]

End-use Efficiency: [4,5] x 1 = [4,5]

Complex Internal Processing: [1..3] x 1 = [1..3]

Code Must Be Reusable: [2..4] x 1 = [2..4]

Easy To Install: [1..2] x 0.5 = [0.5..1]

Easy To Use: [4,5] x 0.5 = [2,2.5]

Portable: [1,2] x 2 = [2,4]

Easy To Change: [3,4] x 1 = [3,4]

Special Security: [4,5] x 1 = [4,5]

Direct Access for 3rd Parties: [3..5] x 1 = [3..5]

=> Totaal: [23.5,37.5]

d) [1]

Adjusted Function Points: [0.89,1.03] x [45,60] = [40.05,61.8]

e) [0,5]

[42.72,49.44] x [40.05,61.8] = [1710.94,3055.39]

Extra pagina voor oplossing vraag 15.