

Toetsing van Domein B (Grondslagen) en Domein D (Programmeren)

Martin Bruggink en Renske Weeda

I&I

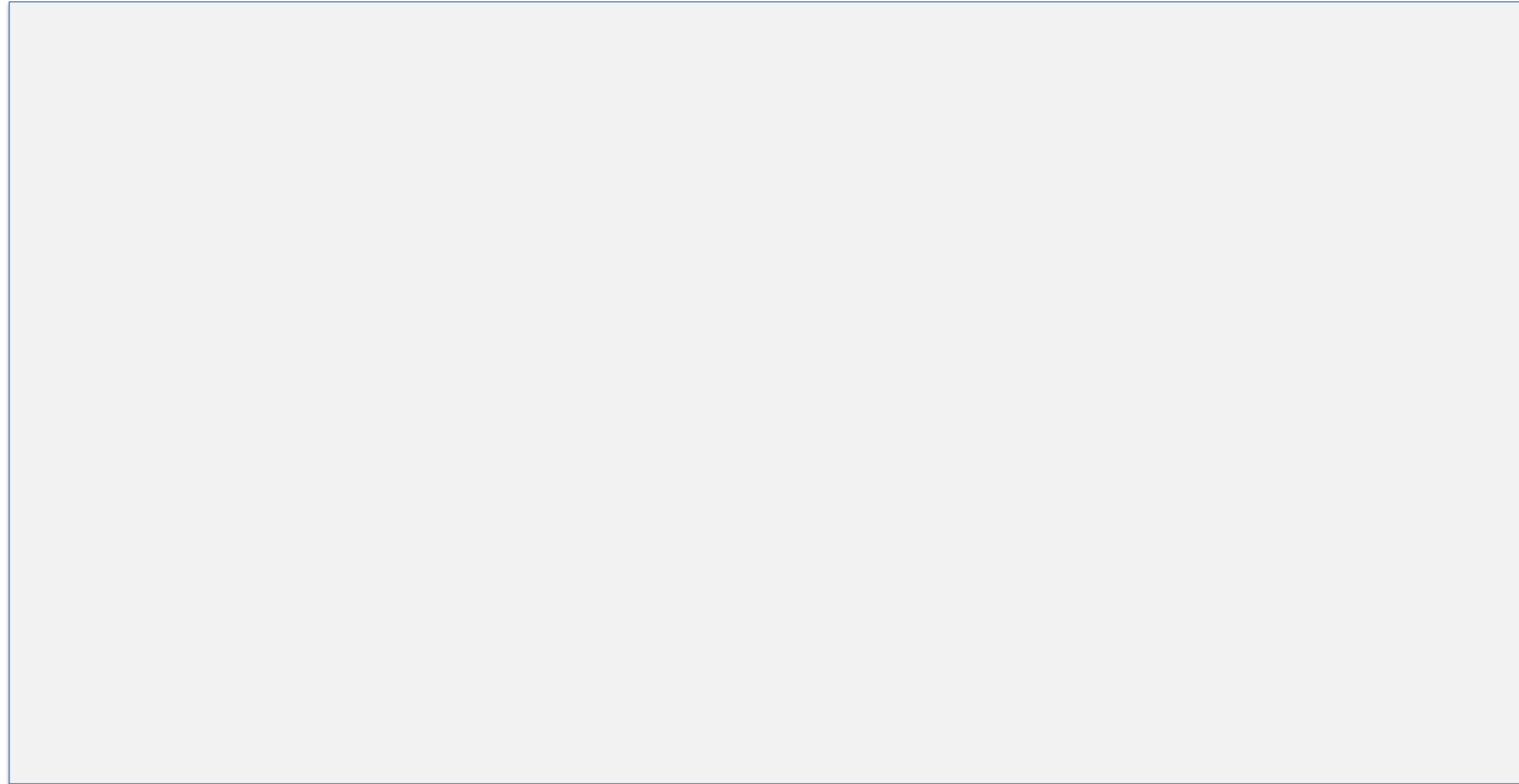
7 november 2019

Agenda

- Domein B (grondslagen)
 - Van eindterm naar leerdoelen
 - Van leerdoelen naar toetsvragen
- Domein D (programmeren)
 - Van eindterm naar leerdoelen
 - Van leerdoelen naar toetsvragen

(Samenvatting) Leerdoelen Domein B en D

Domein B: Grondslagen

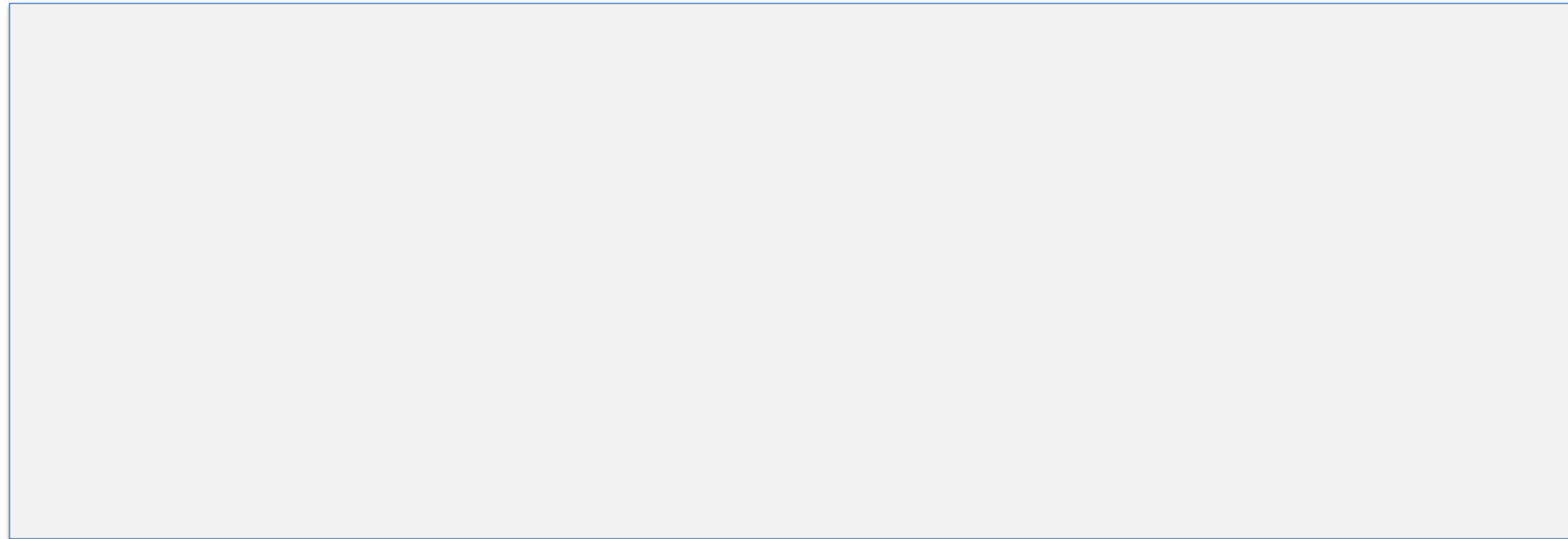


(Samenvatting) Leerdoelen Domein B en D

Domein B: Grondslagen

Subdomein B1: Algoritmen

14. De kandidaat kan een oplossingsrichting voor een probleem uitwerken tot een algoritme, daarbij standaardalgoritmen herkennen en gebruiken, en de correctheid en efficiëntie van digitale artefacten onderzoeken via de achterliggende algoritmen.



(Samenvatting) Leerdoelen Domein B en D

Domein B: Grondslagen

Subdomein B1: Algoritmen

14. De kandidaat kan een oplossingsrichting voor een probleem uitwerken tot een algoritme, daarbij standaardalgoritmen herkennen en gebruiken, en de correctheid en efficiëntie van digitale artefacten onderzoeken via de achterliggende algoritmen.

Subdomein B2: Datastructuren

15. De kandidaat kan verschillende abstracte datastructuren met elkaar vergelijken op elegantie en efficiëntie.

Subdomein B3: Automaten

16. De kandidaat kan eindige automaten gebruiken voor de karakterisering van bepaalde algoritmen.

Subdomein B4: Grammatica's

17. De kandidaat kan grammatica's hanteren als hulpmiddel bij de beschrijving van talen.

Voorbeeld specificaties: website SLO

Home Wat is je zoekopdracht? 🔍



slo ● ● ● Handreiking schoolexamen
Informatica

Voorbeeldspecificaties Domein B
12-2-2018

B1: Algoritmen
De kandidaat kan:

- een gegeven oplossingsrichting voor een probleem weergeven als een **algoritme**. Hierbij wordt verwacht dat kandidaten een algoritme op een gestructureerde wijze kunnen weergeven met bijvoorbeeld een **flowchart** of in **pseudocode**.
 - Het algoritme is opgebouwd uit de basisbouwstenen **openvolging**, **keuze** en **herhaling**.
 - *vwo: Het algoritme kan **recursie** bevatten*
- een gegeven digitaal artefact modelleren met behulp van een algoritme.
- het gedrag van een programma onderzoeken via het onderliggende algoritme, en zo problemen (bijvoorbeeld fouten of inefficiëntie) herleiden tot aspecten van dat algoritme.
- een aantal standaardalgoritmen herkennen en gebruiken.
 - Kandidaten kennen standaardalgoritmen voor elementaire numerieke operaties zoals minimum en maximum bepalen, sommeren, en ten minste twee andere doelen zoals zoeken, sorteren,

Contactpersoon
Victor Schmidt

Dit hoort bij

- Domein B: Grondslagen

Zie ook

- Subdomein B1: Algoritmen
- Subdomein B2: Datastructuren
- Subdomein B3: Automaten
- Subdomein B4: Grammatica's

Algemene informatie
Het examenprogramma
Het PTA
Toetsen in het schoolexamen
Afstemming met andere vakken
Scholing

Sector

B1: Algoritmen

De kandidaat kan:

- een gegeven oplossingsrichting voor een probleem weergeven als een **algoritme**. Hierbij wordt verwacht dat kandidaten een algoritme op een gestructureerde wijze kunnen weergeven met bijvoorbeeld een **flowchart** of in **pseudocode**.
 - Het algoritme is opgebouwd uit de basisbouwstenen **openvolging**, **keuze** en **herhaling**.
 - *vwo: Het algoritme kan **recursie** bevatten*
- een gegeven digitaal artefact modelleren met behulp van een algoritme.
- het gedrag van een programma onderzoeken via het onderliggende algoritme, en zo problemen (bijvoorbeeld fouten of inefficiëntie) herleiden tot aspecten van dat algoritme.
- een aantal standaardalgoritmen herkennen en gebruiken.
 - Kandidaten kennen standaardalgoritmen voor elementaire numerieke operaties zoals minimum en maximum bepalen, sommeren, en ten minste twee andere doelen zoals zoeken, sorteren, datacompressie en graafalgoritmen (bijvoorbeeld routebepaling).
 - Kandidaten kennen bij tenminste één doel ten minste twee standaardalgoritmen.
- de **correctheid** van een gegeven algoritme onderzoeken, en algoritmen (waaronder standaardalgoritmen), vergelijken met betrekking tot **efficiëntie**.
 - Formele bewijzen van de correctheid van een algoritme, bijvoorbeeld met behulp van post- en precondities en invarianten van herhalingslussen, zijn niet noodzakelijk.
 - Absolute uitspraken over efficiëntie van een algoritme in termen van complexiteitsklassen komen aan bod in **keuzedomein G**.

B1: Algoritmen

De kandidaat kan:

- een gegeven oplossingsrichting voor een probleem weergeven als een **algoritme**. Hierbij wordt verwacht dat kandidaten een algoritme op een gestructureerde wijze kunnen weergeven met bijvoorbeeld een **flowchart** of in **pseudocode**.
 - Het algoritme is opgebouwd uit de basisbouwstenen **openvolging**, **keuze** en **herhaling**.
 - *vwo: Het algoritme kan **recursie** bevatten*
- een gegeven digitaal artefact modelleren met behulp van een algoritme.
- het gedrag van een programma onderzoeken via het onderliggende algoritme, en zo problemen (bijvoorbeeld fouten of inefficiëntie) herleiden tot aspecten van dat algoritme.
- een aantal standaardalgoritmen herkennen en gebruiken.
 - Kandidaten kennen standaardalgoritmen voor elementaire numerieke operaties zoals minimum en maximum bepalen, sommeren, en ten minste twee andere doelen zoals zoeken, sorteren, datacompressie en graafalgoritmen (bijvoorbeeld routebepaling).
 - Kandidaten kennen bij tenminste één doel ten minste twee standaardalgoritmen.
- de **correctheid** van een gegeven algoritme onderzoeken, en algoritmen (waaronder standaardalgoritmen), vergelijken met betrekking tot **efficiëntie**.
 - Formele bewijzen van de correctheid van een algoritme, bijvoorbeeld met behulp van post- en precondities en invarianten van herhalingslussen, zijn niet noodzakelijk.
 - Absolute uitspraken over efficiëntie van een algoritme in termen van complexiteitsklassen komen aan bod in **keuzedomein G**.

Leerdoel domein B.1: Standaardalgoritmen

“Kandidaten kunnen standaardalgoritmen voor elementaire numerieke operaties zoals minimum en maximum bepalen, sommeren en ten minste twee andere doelen”

Een **standaardalgoritme** kun je opvatten als een **plan** dat experts inzetten om een **veelvoorkomend** doel te bereiken (Soloway et. al 1985).

Waarom deze leerdoel?

- Kennen van plannen maakt het oplossen van ‘bekende’/’standaard’ problemen eenvoudiger
- Kennen van plannen maakt het mogelijk om deze samen te voegen om complexere problemen op te lossen
- Herkennen van plannen in code maken code makkelijker te lezen (doel achterhalen) en aan te passen / uit te breiden

Standaard recepten expliciet onderwijzen en toetsen (deRaadt, 2008)

Leerdoel domein B.1: Standaardalgoritmen

Welke plannen zou een leerling moeten kennen?

- Sommeerplan (Engels: Sum plan)
- Minimumplan
- Maximumplan
- Telplan: tel het aantal voorkomens (Engels: Count plan)
- Gemiddeldeplan (Engels: Average plan)
- Swapplan

Leerdoel domein B.1: Standaardalgoritmen

Welke plannen zou een leerling moeten **kennen**?

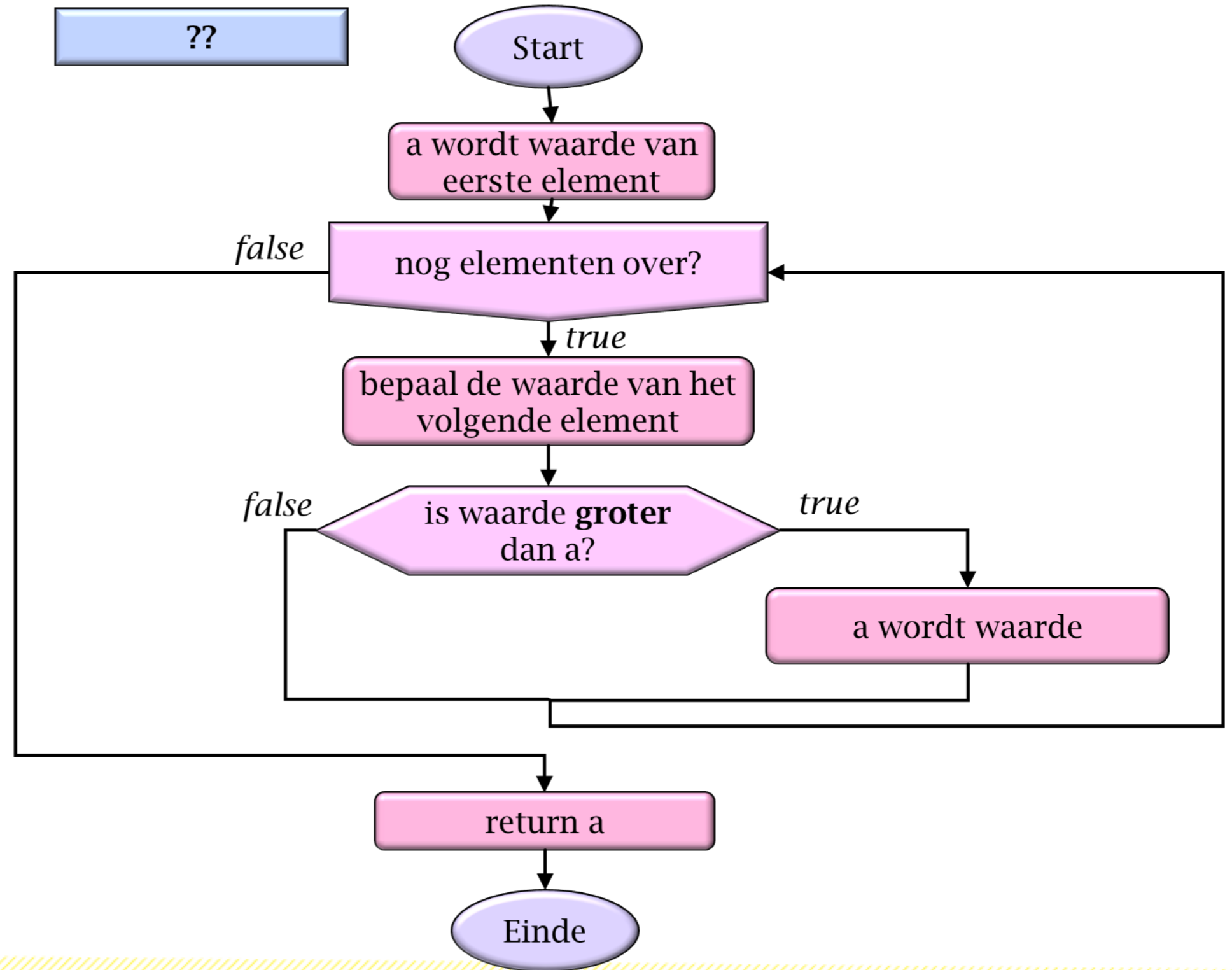
- Sommeerplan (Engels: Sum plan)
- Minimumplan
- Maximumplan
- Telplan: tel het aantal voorkomens (Engels: Count plan)
- Gemiddeldeplan (Engels: Average plan)
- Swapplan

Een leerling kan standaardalgoritmen (en dus plannen) **herkennen, toepassen, aanpassen** of **combineren** om een specifiek probleem in een bepaalde context/domein op te lossen.

Voorbeeld vragen

Herkennen

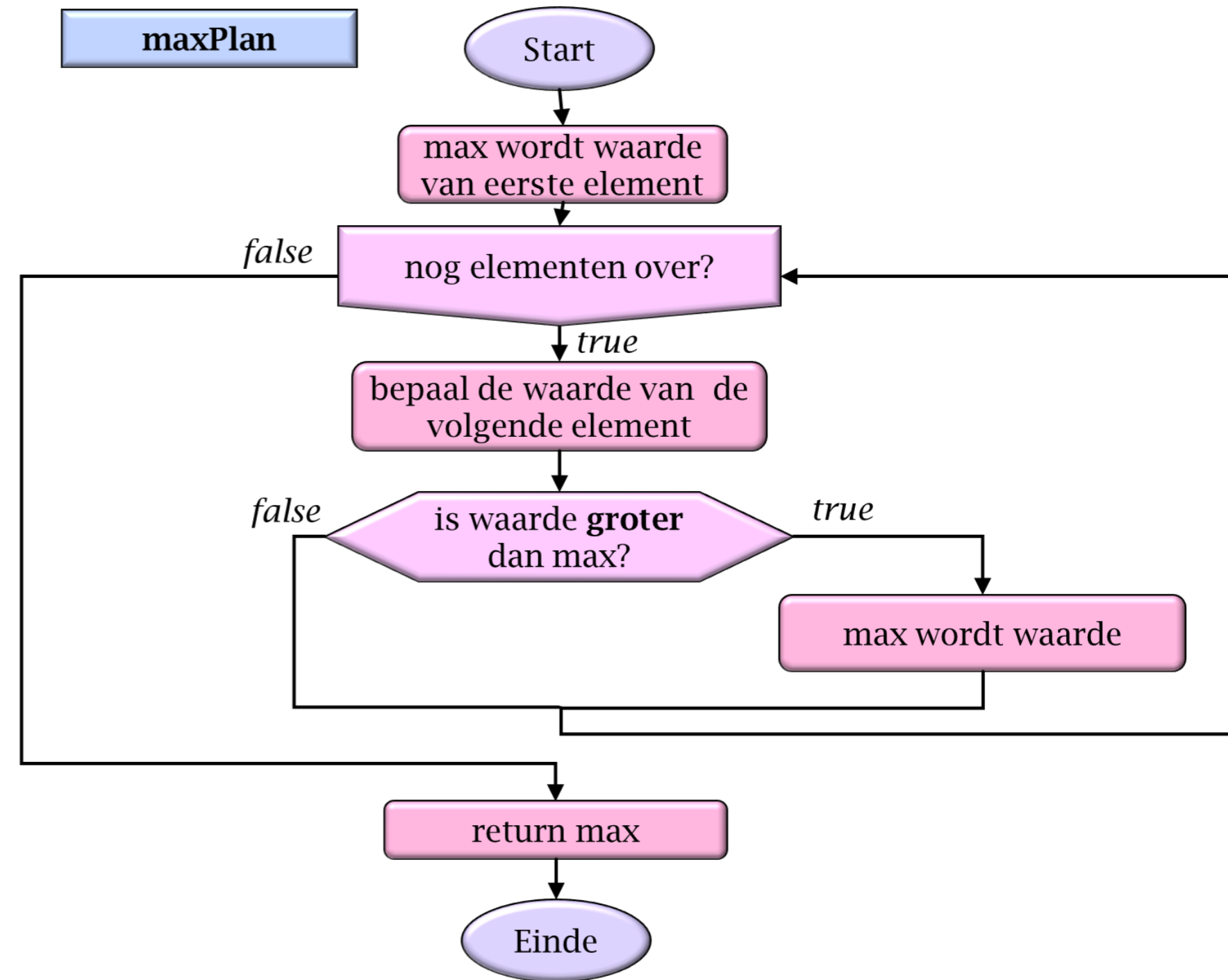
- Gegeven de lijst $[3, -4, 9, 8]$ als invoer, wat levert dit algoritme op?
- Geef een geschikte naam voor a.
- Vat in eigen woorden samen** wat het **doel** van het algoritme is.



Voorbeeld vragen

Toepassen

Een meetstation heeft de dagelijkse **temperatuur** van de afgelopen maand bijgehouden. Teken een stroomdiagram voor het bepalen van de hoogst gemeten temperatuur.

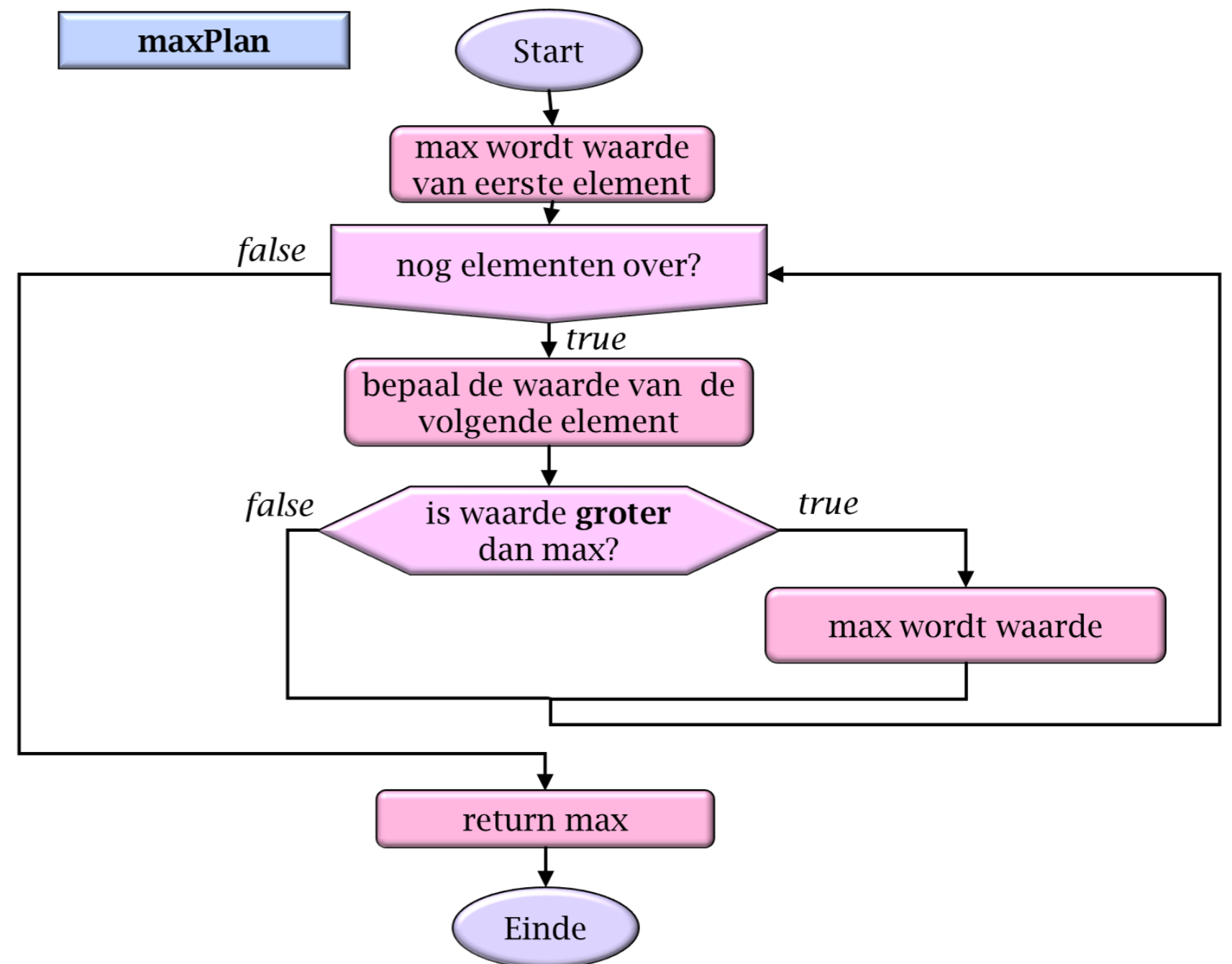


Voorbeeld vragen

Aanpassen

Een meetstation heeft de dagelijkse temperatuur van de afgelopen maand bijgehouden. Het algoritme hiernaast bepaalt de hoogst gemeten temperatuur.

- Een defect meetinstrument geeft soms onterecht de waarde 999 aan. **Pas het stroomdiagram aan zodat deze waarde genegeerd wordt.**
- Pas het stroomdiagram om aan het **minimum** te bepalen.



Voorbeeld vragen

Combineren

Een meetstation heeft de dagelijkse temperatuur van de afgelopen maand bijgehouden. Men wil weten op **hoeveel** dagen de **hoogste** temperatuur van die maand is bereikt. Teken een stroomdiagram voor het bepalen hiervan.

Oplossing 1 (multi-structural)

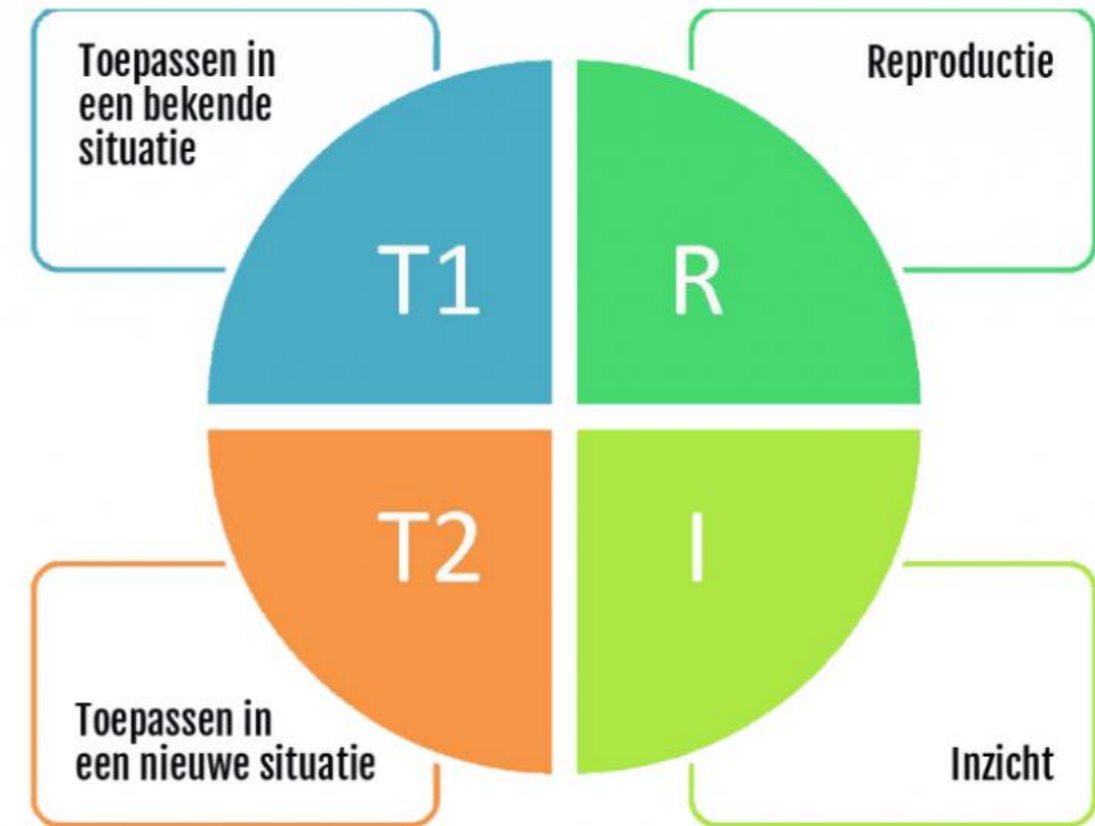
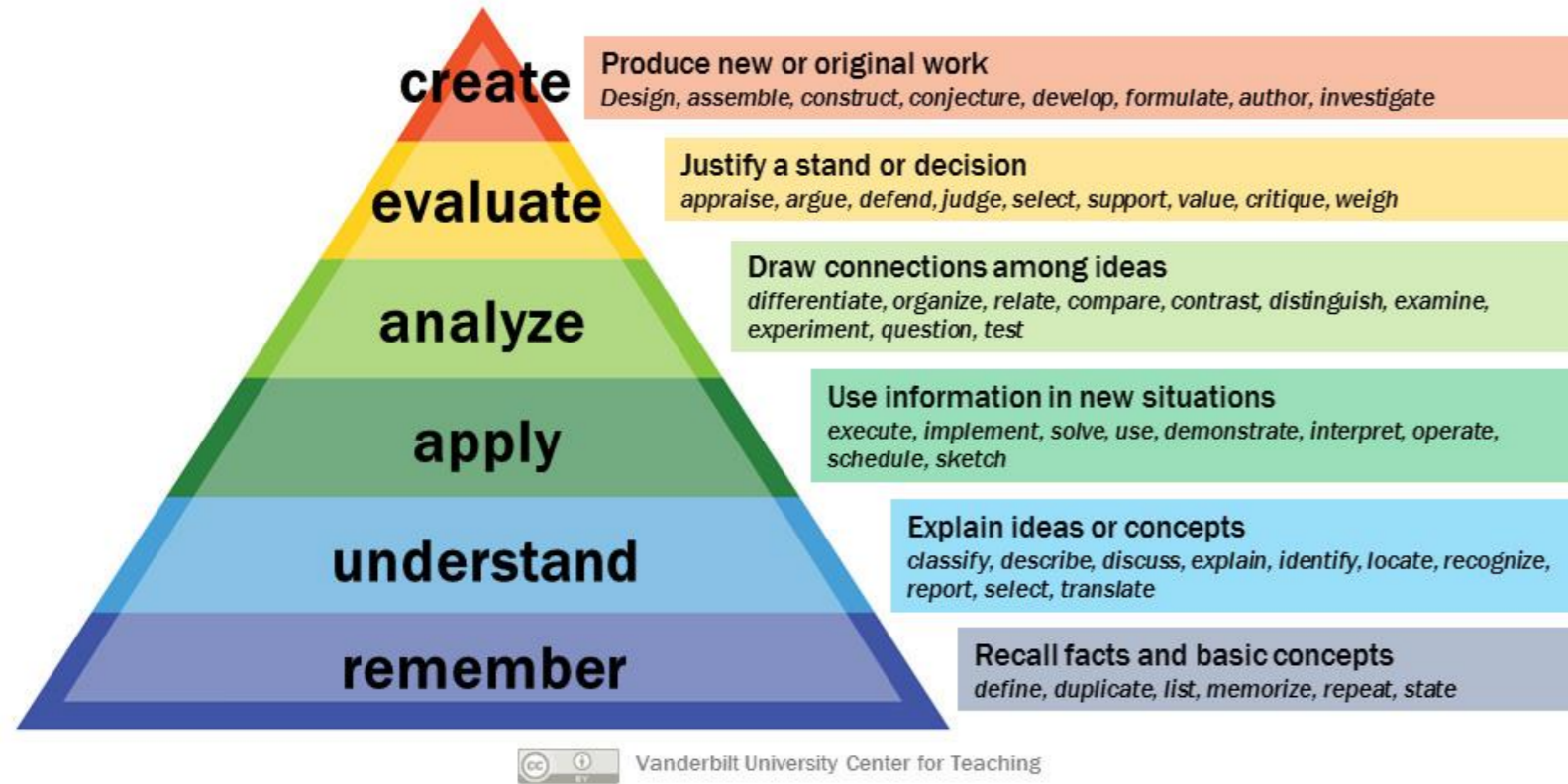
1. Bepaal maximum (maxplan)
2. Tel aantal voorkomens van maximum (telplan)

Oplossing 2 (relational)






Integreren van maxplan en telplan, waardoor minder iteraties nodig zijn.

Taxonomieën: Bloom, RTTI

Bloom's Taxonomy



Different stages of S O L O Taxonomy

<i>Pre-Structural Level</i>	<i>Uni – Structural Level</i>	<i>Multi-Structural Level</i>	<i>Relational Level</i>	<i>At the extended abstract Level</i>
<p><i>I .m not sure about this subject</i></p>  <p>A blue arrow points from the sign to a red square containing a white circle.</p>	<p><i>I have one idea about this subject</i></p>  <p>A blue arrow points from the sign to a yellow square containing a white rectangle.</p>	<p><i>I have several ideas about this subject</i></p>  <p>Three blue arrows point from the sign to three separate white rectangles arranged horizontally on a yellow background.</p>	<p><i>I can link my ideas together to see the big picture</i></p>  <p>A blue arrow points from the sign to a green square containing three white rectangles connected by lines to form a diamond shape.</p>	<p><i>I can look at these ideas in a new and different ways</i></p>  <p>A blue arrow points from the sign to a green square containing three white rectangles connected by lines to form a tree structure, with a small diagram of an atom in the top right corner.</p>

SOLO niveaus

SOLO niveau	Omschrijving
Uni-structural	Een blok(je) code; één of eenvoudig concept.
Multi-structural	Twee of meer blokken code achter elkaar; meerdere concepten. (Verschillende onderdelen, maar niet het geheel.)
Relational	Meerdere blokken in elkaar verweven (meerdere plannen in elkaar verweven); complexe concepten

SOLO niveaus combineren met Bloom

Bloom niveau	Omschrijving	Uni-structural	Multi-structural	Relational
Understanding	The ability to summarize, explain, exemplify, classify, and compare CS concepts, including programming constructs.			
Applying	The ability to execute programs or algorithms, to track them, and to recognize their goals.			
Creating	The ability to plan and produce programs or algorithms.			

Soort vragen

- Traceer
- voeg commentaar toe
- Hernoem
- Omschrijf rol van een variabele
- Vat doel samen
- Pas aan
- Herschrijf
- Analyseer correctheid
- Wijs de fout aan
- Corrigeer
- Vergelijk efficiëntie
- Bepaal oorzaak inefficiëntie
- Omschrijf/implementeer een verbetervoorstel
- Stel voorwaarden aan de invoer
-

Bloom niveau	Omschrijving	Uni-structural	Multi-structural	Relational
Understanding	The ability to summarize, explain, exemplify, classify, and compare CS concepts, including programming constructs.			
Applying	The ability to execute programs or algorithms, to track them, and to recognize their goals.			
Creating	The ability to plan and produce programs or algorithms.			

Oplossing 1: abutment (multi-structural)

Schrijf een programma dat print op hoeveel dagen de maximale hoeveelheid regen is gevallen

```
regenmetingen = [25, 70, 70, 70, 76, 76, 0, 34]
```

```
max = regenmetingen[0]    #pak eerste waarde uit lijst
#bekijk elke waarde in de lijst en onthoud de hoogste
for waarde in regenmetingen:
    if waarde > max:
        max = waarde
```

```
max_teller = 0           #een teller bijhouden voor freq. van max
#bekijk elke waarde in de lijst en houd bij hoe vaak de hoogste voorkomt
for waarde in regenmetingen:
    if waarde == max:
        max_teller += 1
```

```
print( max_teller )
```

Oplossing 2: merging (relational)

Schrijf een programma dat print op hoeveel dagen de maximale hoeveelheid regen is gevallen

```
regenmetingen = [25, 70, 70, 70, 76, 76, 0, 34]

max = regenmetingen[0] #pak eerste waarde uit lijst
max_teller = 0 #een teller bijhouden voor freq. van max
for waarde in regenmetingen:
    if waarde == max: #max opnieuw gevonden dus teller ophogen
        max_teller += 1
    elif waarde > max: #nieuwe max gevonden, begin opnieuw met tellen
        max = waarde
        max_teller = 1

print( max_teller )
```

Stappen bij het opstellen van een toetsvraag

1. Welk **leerdoel**(en) wil je toetsen?
2. Met welk **bewijsmateriaal** kan een leerling beheersing van de kennis aantonen?
3. In wat voor **situatie** kan jij de leerling zetten om die kennis te **kunnen aantonen**?
4. **Variatiemogelijkheden** van de taak (moeilijker/makkelijker/anders)

Voorbeeld van het ontwikkelen van een toetsvraag: herhalingsconstructie

1. Welk **leerdoel**(en) wil je toetsen?

De leerling kan:

- de waarde van variabelen in een programma met een herhaling traceren

Voorbeeld van het ontwikkelen van een toetsvraag: herhalingsconstructie

2. Met welk **bewijsmateriaal** kan een leerling beheersing van de kennis aantonen?
 - Een correcte trace table en/of resultaat van een programma

Voorbeeld van het ontwikkelen van een toetsvraag: concept for-loop

3. In wat voor **situatie** kan jij de leerling zetten om die kennis te **kunnen aantonen**?

- Gegeven een programma => de waarden van variabelen traceren
- Gegeven een programma en de input => de output bepalen.

4. **Variatiemogelijkheden** van de taak (om het moeilijker/makkelijker/anders te maken):

- abstractieniveau (geef algemene uitvoer in termen van n),
- scaffolding (tussenstap: geef de uitvoer als a=3)
- context variëren en/of context-vrij maken
- representatie (code, stroomdiagram, pseudocode, natuurlijke taal)
- complexere/samengestelde algoritmes (SOLO: Uni-structural, Multistructural, Relational)
- voorwaarden aangeven voor het correct werken van het programma (bij welke invoer types/waarden zal het wel/niet werken?)
- ...

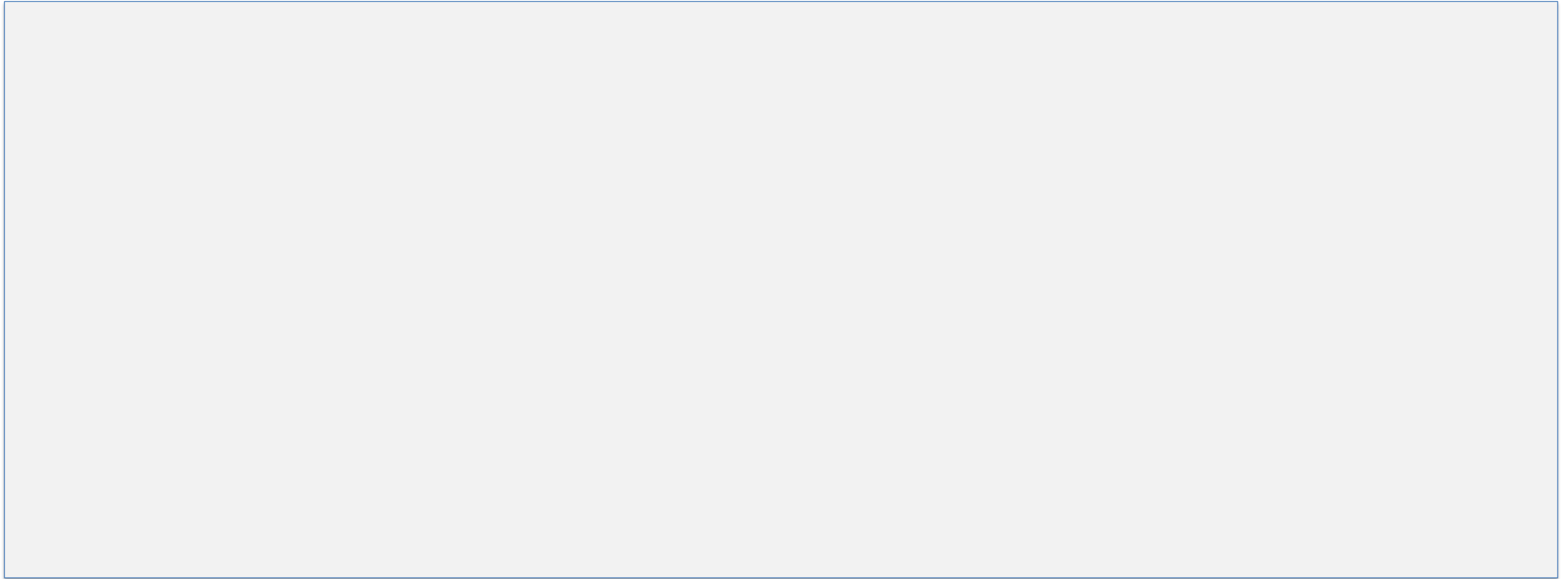
Valkuil: gelijktijdig toetsen van meerdere leerdoele

Tip: theorietoets opbouwen o.a. uit kleinere opgaven die afzonderlijk (aspecten van) leerdoelen toetsen

- Als een leerling bij de ene vraag vastloopt, kan nog wel laten zien wat hij **WEL** kan
- Docent kan beter afleiden wat de knelpunten precies zijn
- Kost de leerling per vraag minder tijd (efficiënter)
- Vragen met veel code kan overdonderend zijn
- Maak onderscheid tussen domeinen B (maak een algoritme) en D (zet een gegeven algoritme om in code)

Eindtermen domein D

Domein D: Programmeren



Eindtermen domein D

Domein D: Programmeren

Subdomein D1: Ontwikkelen

23. De kandidaat kan, voor een gegeven doelstelling, programmacomponenten ontwikkelen in een imperatieve programmeertaal, daarbij programmeertaalconstructies gebruiken die abstractie ondersteunen, en programmacomponenten zodanig structureren dat ze door anderen gemakkelijk te begrijpen en te evalueren zijn.

Eindtermen domein D

Domein D: Programmeren

Subdomein D1: Ontwikkelen

23. De kandidaat kan, voor een gegeven doelstelling, programmacomponenten ontwikkelen in een imperatieve programmeertaal, daarbij programmeertaalconstructies gebruiken die abstractie ondersteunen, en programmacomponenten zodanig structureren dat ze door anderen gemakkelijk te begrijpen en te evalueren zijn.

Subdomein D2: Inspecteren en aanpassen

24. De kandidaat kan structuur en werking van gegeven programmacomponenten uitleggen, en zulke programmacomponenten aanpassen op basis van evaluatie of veranderde eisen.

Voorbeeld specificaties: website SLO



Home Wat is je zoekopdracht? 🔍

slo • Handreiking schoolexamen Informatica

Voorbeeldspecificaties Domein B
12-2-2018

B1: Algoritmen
De kandidaat kan:

- een gegeven oplossingsrichting voor een probleem weergeven als een **algoritme**. Hierbij wordt verwacht dat kandidaten een algoritme op een gestructureerde wijze kunnen weergeven met bijvoorbeeld een **flowchart** of in **pseudocode**.
 - Het algoritme is opgebouwd uit de basisbouwstenen **openvolging**, **keuze** en **herhaling**.
 - *vwo: Het algoritme kan **recursie** bevatten*
- een gegeven digitaal artefact modelleren met behulp van een algoritme.
- het gedrag van een programma onderzoeken via het onderliggende algoritme, en zo problemen (bijvoorbeeld fouten of inefficiëntie) herleiden tot aspecten van dat algoritme.
- een aantal standaardalgoritmen herkennen en gebruiken.
 - Kandidaten kennen standaardalgoritmen voor elementaire numerieke operaties zoals minimum en maximum bepalen, sommeren, en ten minste twee andere doelen zoals zoeken, sorteren,

Contactpersoon
Victor Schmidt

Dit hoort bij

- Domein B: Grondslagen

Zie ook

- Subdomein B1: Algoritmen
- Subdomein B2: Datastructuren
- Subdomein B3: Automaten
- Subdomein B4: Grammatica's

Algemene informatie
Het examenprogramma
Het PTA
Toetsen in het schoolexamen
Afstemming met andere vakken
Scholing

Sector

Inspecteren en aanpassen

De kandidaat kan:

- de structuur en werking van een gegeven programmacomponent uitleggen.
- een gegeven programmacomponent evalueren aan de hand van de eigenschappen correctheid, efficiëntie, en leesbaarheid.
- een bestaande programmacomponent aanpassen
 - als gevolg van een evaluatie van bijvoorbeeld de correctheid, efficiëntie of leesbaarheid van de programmacomponent.
 - als gevolg van een veranderde of uitgebreide doelstelling.

Stappen bij het opstellen van een toetsvraag

1. Welk **leerdoel**(en) wil je toetsen?
2. Met welk **bewijsmateriaal** kan een leerling beheersing van de kennis aantonen?
3. In wat voor **situatie** kan jij de leerling zetten om die kennis te **kunnen aantonen**?
4. **Variatiemogelijkheden** van de taak (moeilijker/makkelijker/anders)

1. Welk leerdoel(en) wil je toetsen?

De kandidaat kan:

Een gegeven programma beredeneren over de uitvoer van het programma

2. Met welk bewijsmateriaal kan een leerling beheersing van de kennis aantonen?

Een correcte trace table en/of resultaat van een programma

3. In wat voor **situatie** kan jij de leerling zetten om die kennis te **kunnen aantonen**?

- Gegeven een programma + invoer => geef de uitvoer

4. **Variatiemogelijkheden** van de taak (om het moeilijker/makkelijker/anders te maken):

- gegeven programma en uitvoer, wat zijn mogelijk juiste invoeren?
- voorwaarden aangeven voor het correct werken van het programma
- context variëren en/of context-vrij maken
- abstractieniveau (leg in eigen woorden uit wat het programma doet?)
- scaffolding (geef een voorbeeld, deel van de code toelichten met commentaar)
- complexere/samengestelde algoritmes (SOLO: Uni-structural, Multi-structural, Relational)

- ...

Multiple Choice

Wat wordt in de console geprint na uitvoer van de code hieronder:

```
voorraad = 48
nodig = 93
bestellen = 0
if voorraad < nodig:
    bestellen = nodig - voorraad
else:
    voorraad = voorraad + nodig
print(voorraad, nodig, bestellen)
```

- a) 48 93 141
- b) 141 93 45
- c) 141 93 0
- d) 48 93 45

Misconcepten:

- a) Nadat de **else** is uitgevoerd kan de **if** nog uitgevoerd worden
- b) Zowel de **if** als de **else** worden uitgevoerd
- c) Alleen de **else** wordt uitgevoerd

Correcte antwoord: Alleen de **if** wordt uitgevoerd

Domein D: Parson's probleem (2D met afleiders)

Q-1: [31c] Het volgende programma moet de getallen 0 tot 10 bij elkaar optellen. Maar de blokken staan in de verkeerde volgorde. Sleep de benodigde code naar de rechterkant en plaats deze in de juiste volgorde. Zorg ook dat de regels code juist staan ingesprongen. Als je denkt dat jouw oplossing helemaal goed is, klik dan op de knop *Check Me* om jouw oplossing te controleren.

Drag from here

Drop blocks here

```
1a while teller <
  10:
1b while teller
  <= 10:
2 teller = 0
  totaal = 0
3a teller +=
  totaal
3b totaal +=
  teller
4a teller += 1
4b totaal += 1
5a print(totaal)
5b print(teller)
```

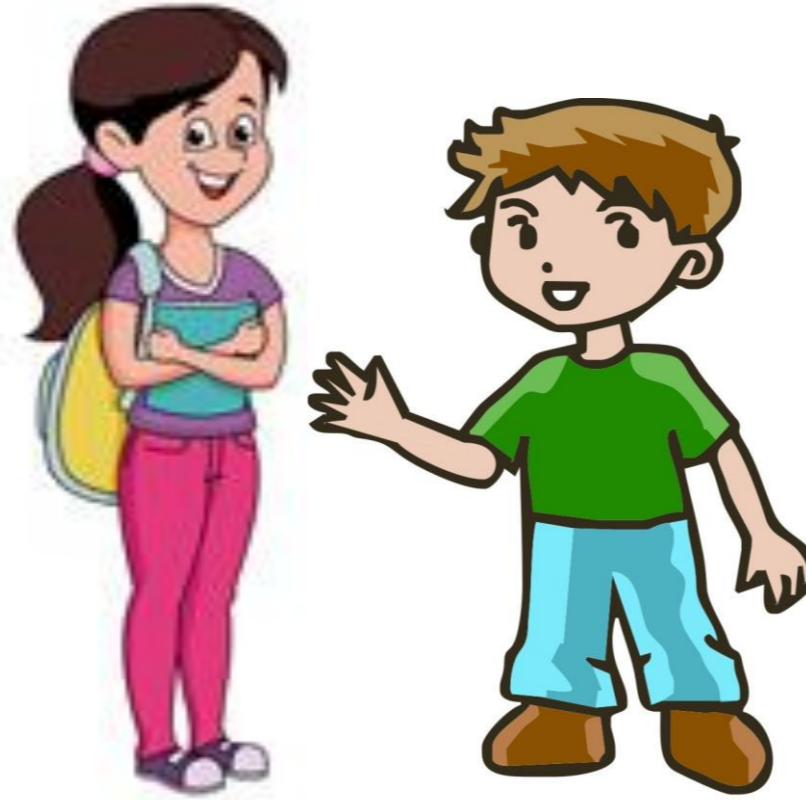
Taak:

- Juiste volgorde (algoritmiëk)
- Selecteren benodigde code (afleiders)
- Semantiek (juist inspringen)

Eigenschappen van vraag:

- Objectief en efficiënt na te kijken
- Toetsen op misconcepten
- Efficiënt voor leerling
- Plannen vs doel

Toetsen



Criteria van Nuy (1994)

- 1) *De toets is representatief (dekkend) voor de behandelde stof.*
- 2) *Er worden relevante vragen gesteld.*
- 3) *De leerling kan laten zien wat hij kan (demonstreerbaarheid).*
- 4) *De toets meet de kennis op een valide (geldige) manier.*
- 5) *De toets meet de kennis op een betrouwbare manier.*
- 6) *De toets meet de kennis op een efficiënte manier.*
- 7) *De toets is rechtvaardig.*
- 8) *De toets is objectief.*

De toets is representatief (dekkend) voor de behandelde stof.

- *De vragen zijn te maken aan de hand van dat wat de leerlingen in de les hebben gehad.*
- *De vragen zijn naar evenredigheid verdeeld over de stof.*

Er worden relevante vragen gesteld.

Nuy schrijft hierover: 'Een goede toets bestaat uit vragen die relevant zijn voor de eigenlijke bedoelingen van het gegeven onderwijs (waar gaat het om?).' En: 'De toets moet niet naar triviale details en dergelijke vragen en geen problemen voorleggen die een beroep doen op triviale vormen van beheersing van de stof.'

De leerling kan laten zien wat hij kan (demonstreerbaarheid).

Nuy schrijft: 'Met een toets moet een leerling niet alleen kunnen laten zien wat hij niet beheerst, maar ook wat hij wel beheerst'.

De toets meet de kennis op een valide (geldige) manier.

Het gaat volgens Nuy om de volgende drie voorwaarden:

- De keuze van een passende toetsvorm. Bijvoorbeeld: praktische vaardigheden kunnen niet goed gemeten worden met een schriftelijke toets.
- De toetsvragen moeten de leerdoelen goed vertolken. Bijvoorbeeld: bij het toetsen van een bepaald begrip door een meerkeuzevraag dient kennis van dit begrip bepalend te zijn voor het al of niet juist kunnen beantwoorden van de vraag en niet de leesvaardigheid van de leerling. Een niet-valide vraag schiet zijn doel voorbij.
- De toets moet op de juiste wijze uit de valide vragen zijn opgebouwd.

De toets meet de kennis op een betrouwbare manier.

Nuy noemt vier eisen ten aanzien van de betrouwbaarheid van een toets:

- De toets moet bestaan uit een voldoende aantal vragen. In het algemeen geldt: hoe meer vragen (gespreid over de stof), hoe betrouwbaarder de toets.
- In de toets moeten geen dingen staan die je gemakkelijk verkeerd leest of waar je gemakkelijk overheen leest.
- Een toets moet niet onder tijdsdruk afgenomen worden.
- Een toets moet niet erg gevoelig zijn voor correctiefouten door de leraar.

De toets meet de kennis op een efficiënte manier.

De toets is rechtvaardig.

Nuy noemt drie criteria voor een rechtvaardige toets.

- Alle leerlingen worden gelijk behandeld.
- De leerlingen zijn tijdens het maken van de toets op de hoogte van de wijze van beoordeling.
- De normering van de toets is onafhankelijk van de toetsprestaties in de klas waartoe de leerling toevallig behoort.

De toets is objectief.

Nuy noemt twee criteria.

- Er moet sprake zijn van een zodanig volledig en eenduidig antwoordmodel en scoringsvoorschrift dat toepassing daarvan leidt tot (vrijwel) hetzelfde toetscijfer, ongeacht degene die het nakijkt.
- Er dient overeenstemming te bestaan tussen deskundigen op het betreffende vakgebied (vakcollega's) over de juistheid van het antwoordmodel en het scoringsvoorschrift.

Bekende misconcepten gebruiken bij het opstellen van een toetsopgave

Bekende misconcepten bij een for-loop:

- A. De teller begint bij 1
- B. De teller gaat door t/m het eind getal
- C. Variabelen buiten de loop kunnen niet gebruikt worden / problemen met bereik

We bekijken deze aan de hand van de volgende voorbeelden:

```
for i in range(4):  
    print(i)
```

```
x = 0  
for i in range(4):  
    x = x + i  
    print(x)  
print(x)
```

Formatieve evaluatie met gebruik van bekende misconcepten

Vraag: Wat wordt er in de console afgedrukt?

```
for i in range(4):  
    print(i)
```

Bekende misconcepten bij een for loop:

A. De teller begint bij 1

Leerling verwacht dat 1 2 3 geprint wordt

B. De teller gaat door t/m het eind getal

Leerling verwacht dat 0 1 2 3 4 geprint wordt

Handige Multiple-choice opties:

- 1 2 3 **#misconcept A**
- 0 1 2 3 **#correcte antwoord**
- 0 1 2 3 4 **#misconcept B**
- 1 2 3 4 **#misconcept A en B**

Summatieve toetsing met gebruik van bekende misconcepten

Vraag: Wat wordt er in de console afgedrukt?

```
x = 0
for i in range(4):
    x = x + i
    print(x)
print(x)
```

Bekende misconcepten bij een for loop:

A. De teller begint bij 1

Leerling verwacht dat 1+2+3 geprint wordt

B. De teller gaat door t/m het eind getal

Leerling verwacht dat 0+1+2+3 geprint wordt

C. Variabelen buiten de loop kunnen niet gebruikt worden

Leerling verwacht een foutmelding

Handige Multiple-choice opties:

- 0 1 2 3 4 **#misconcept B**
- 1 3 3 6 **#misconcept A**
- 0 1 3 6 6 **#correcte antwoord**
- 1 3 6 10 **#misconcept A en B**
- Foutmelding **#misconcept C**

Hoe kom je aan een lijst misconcepten?

- Houd zelf bij wat je leerlingen zeggen/doen in de klas/toets
- Teemu Sirkiä and Juha Sorva. 2012. Exploring programming misconceptions: an analysis of student mistakes in visual program simulation exercises. In Proceedings of the 12th Koli Calling International Conference on Computing Education Research (Koli Calling '12). ACM, New York, NY, USA, 19-28. DOI=<http://dx.doi.org/10.1145/2401796.2401799>

Misc5: Conditional into loop control variable

```
while i < 7:
```

After evaluating the expression, the student assigns its value to i.

- Professional Development for CS Principles Teaching: <https://www.pd4cs.org/>

M&C2: Correct use of lists and arrays

Misconceptions, unfamiliarity, and difficulties with lists and arrays are magnified in a loop environment. Common errors are related to indexing and include

- Indexing starting at 0 versus 1 (e.g., for mystr = '123', mystr[0] is '1' and mystr[1] is '2')
- Incorrect use or not understanding the termination condition in while loops.
- Mistakes in the boundary conditions. This includes not properly handling cases where inputs are negative or zero, lists, arrays, or empty strings.
- Fencepost Errors. If we build a straight fence 30 meters long with posts spaced 3 meters apart, how many posts do we need? The intuitive answer of 10 spaces between them, or vice versa, or neglecting to consider whether one should count one or both ends of a row leads to execution errors or incorrect

Soorten vragen: vorm

- Multiple – choice
- Korte antwoord (ligt toe, geef commentaar, vergelijk, benoem..)
- Combinatie van beiden hierboven
- Code schrijven (korte programma's)

Combinatie korte (gesloten) vragen gecombineerd met toelichting

Stacy runs a food bank.

- The types of cans donated to the food bank are vegetables, fruits, meat, and soup.
- Volunteers put the cans randomly on the storage shelves, wherever they find space.
- Stacy packs the cans into many food boxes a week.
- Each box has the same number and types of canned food.
- It takes Stacy a long time to find the cans she needs from the shelves.

Stacy wants to create a method for organizing the cans on the shelves.

a) What problem does Stacy hope to solve by creating a method for organizing the cans on the shelves?

In parts (b) and (c), you will create a step-by-step method that can be used to *systematically* organize the cans that are currently on the shelves.

b) List one piece of information you need to know to create your method.

c) Create a method for Stacy to systematically organize the cans that are currently on the shelves. List your method as a series of steps.

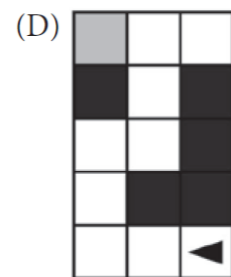
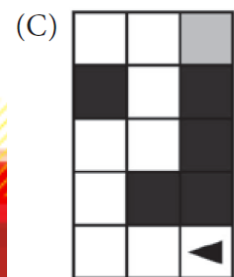
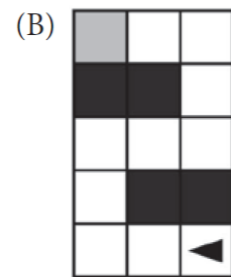
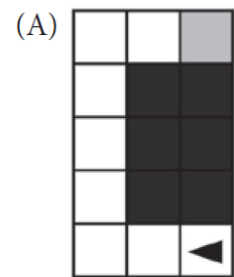
d) A new type of food—condensed milk—is found on the shelves. Does your method from part (c) still work? If yes, explain how the method would work. If no, explain how to modify your method.

Voorbeeldvraag algoritmen (Domein B)

The code segment below is intended to move a robot in a grid to a gray square. The program segment uses the procedure `GoalReached`, which evaluates to `true` if the robot is in the gray square and evaluates to `false` otherwise. The robot in each grid is represented as a triangle and is initially facing left. The robot can move into a white or gray square but cannot move into a black region.

```
REPEAT UNTIL (GoalReached ())
{
  IF (CAN_MOVE (forward))
  {
    MOVE_FORWARD ()
  }
  IF (CAN_MOVE (right))
  {
    ROTATE_RIGHT ()
  }
  IF (CAN_MOVE (left))
  {
    ROTATE_LEFT ()
  }
}
```

For which of the following grids does the code segment NOT correctly move the robot to the gray square?



Hogere orde leerdoelen (analytische vaardigheden):

- Problemen analyseren
- Algoritmen evalueren op correctheid

Eigenschappen van vraag:

- Objectief en efficiënt nakijken (meerkeuze)
- Efficiënt voor de leerling:
 - Geen code schrijven
 - Korte code lezen
- Taal onafhankelijk
- Foutieve antwoorden:
 - zijn niet opvallend (reëel)
 - toetsen op misconcepten (scharnier vraag)

Domein D: Begrip over een programma toetsen

Bekijk de volgende methode:

```
private int surprise( int m, int n ) {
    int j = 0;
    int i = 0;
    while (i < n) {
        j = j + m;
        i = i + 1;
    }
    return j;
}
```

a) Bereken welke retourwaarde de volgende functie-aanroep zal geven:

```
surprise(10,5)
```

b) Geef een trace-tabel waarin de waarden van de verschillende variabelen tijdens de uitvoering van de functie worden aangegeven.

c) Beredeneer (zonder een trace-tabel te maken) wat het resultaat is van de aanroep `surprise(99, 100)`. Licht je antwoord toe.

of

Geef een geschikte functienaam voor het programma

Taak:

- Traceren / bepaal uitvoer
- Samenvatten van wat een brok code doet
- Beredeneren / voorspellen van gedrag

Eigenschappen van vraag:

- Objectief en efficiënt na te kijken
- Toetsen op misconcepten
- Efficiënt voor leerling: korte code
- Demonstreerbaarheid: leerling kan kennis en hogere orde vaardigheden aantonen (abstractie)

Domein D: Parson's probleem (2D met afleiders)

Q-1: [31c] Het volgende programma moet de getallen 0 tot 10 bij elkaar optellen. Maar de blokken staan in de verkeerde volgorde. Sleep de benodigde code naar de rechterkant en plaats deze in de juiste volgorde. Zorg ook dat de regels code juist staan ingesprongen. Als je denkt dat jouw oplossing helemaal goed is, klik dan op de knop *Check Me* om jouw oplossing te controleren.

Drag from here

Drop blocks here

```
1a while teller <
  10:
1b while teller
  <= 10:
2 teller = 0
  totaal = 0
3a teller +=
  totaal
3b totaal +=
  teller
4a teller += 1
4b totaal += 1
5a print(totaal)
5b print(teller)
```

Taak:

- Juiste volgorde (algoritmiëk)
- Selecteren benodigde code (afleiders)
- Semantiek (juist inspringen)

Eigenschappen van vraag:

- Objectief en efficiënt na te kijken
- Toetsen op misconcepten
- Efficiënt voor leerling
- Plannen vs doel

Type of question

Type1: Development of a solution

Type2: Development of a solution that uses a given module

Type3: Tracing a given solution

Type4: Analysis of code execution

Type5: Finding the purpose of a given solution

Type6: Examination of the correctness of a given solution

Type7: Completion a given solution

Type8: Instruction manipulations

Type9: "Efficiency" estimation

Type10: Question design

Type11: "Programming" style questions

Type12: Transformation of a solution

Type of question	An example pattern
Type1: Development of a solution	Design a finite automaton that recognizes a regular language L.
Type2: Development of a solution that uses a given module	Given the A1 finite automaton that recognizes the language L1 and A2 finite automaton that recognizes the language L2, design a finite automaton that recognizes the language $L1 \cup L2$.
Type3: Tracing a given solution	Given a push down automaton P, and a word w, show the sequence of states that P goes through while processing w.
Type4: Analysis of code execution	Given a finite automaton A, present: <ul style="list-style-type: none"> – A word that the automaton accepts; – A word that the automaton rejects; – A word that its processing is terminated in the trap state.
Type5: Finding the purpose of a given solution	Given a Turing Machine T, determine what language it accepts.
Type6: Examination of the correctness of a given solution	Does the given Turing Machine T recognize the language L?
Type7: Completion a given solution	Complete the Push Down Automaton P, so it will recognize the language L.
Type8: Instruction manipulations	Given a Turing Machine T, if the transition from state q1 to q2 is replaced by the next transition [to be described], what language will the machine recognize?
Type9: “Efficiency” estimation	Given a finite automaton A that recognizes the language L, can you present a different finite automaton that recognizes the same language with fewer states?
Type10: Question design	Design a question that requires the presentation of a BNF grammar for an irregular language.
Type11: “Programming” style questions	Given three different Push Down Automata that recognize a language L, examine the automata and state which of them, in your opinion, is the best. Explain your answer.
Type12: Transformation of a solution	Given a Turing machine T, present a BNF grammar that expands the same language.

Example: The target of the following two methods is to determine whether an integer number n is a prime number or not.

Method A

```
public static boolean prime (int n) {  
    for (int i=2; i<n; i++) {  
        if (n % i == 0)  
            return false;  
    }  
    return true;  
}
```

Method B

```
public static boolean prime (int n) {  
    if (n % 2 == 0)  
        return false;  
    for (int i=3; i<n; i=i+2) {  
        if (n % i == 0)  
            return false;  
    }  
    return true;  
}
```

Here is a list of questions of different types that can be asked with respect to these methods separately or in any combination according to the teacher's pedagogical purposes.

1. *Type3. Tracing a given solution:* Trace each method when n is 19.
2. *Type4. Analysis of code execution:*
 - For each method, determine how many times the loop is executed for $n = 19$.
 - Find a value of n , for which the loop in Method B is executed ten times. Is there only one answer?
3. *Type5. Finding the purpose of a given solution:* What is the purpose of each method? (can be asked if the problem is not indicated).
4. *Type6. Examination of the correctness of a given solution:* Check the correctness of the two solutions. Do they solve the problem correctly?
5. *Type9. Efficiency estimation:* What is the efficiency of each method?
6. *Type6. Correctness; Type8. Instruction manipulations; Type9. Efficiency:*
 - If you change the upper loop limit in Method B to be $n/2$ (instead of n), is the solution still correct? If it is, what is the method efficiency after the change?
 - If you change the loop limit in Method B to be \sqrt{n} (instead of n), is the solution still correct? If it is, what is the method efficiency after the change?

The task	Pure-algorithmic formulation	Narrative-algorithmic formulation
Find the maximum of a list of numbers.	Write a method that gets as input a list of integers and returns the maximum value in the list.	In a sport competition, 5 classes of 30 pupils each participates in two jumping competitions. Write a program that gets as input, for each class the two results of each student, and for each class displays the best result in each of the two jumping competitions.
Checks whether a given array is sorted.	Write a method that gets an array as a parameter and returns true if the array is sorted; otherwise, it returns false.	A teacher wishes to encourage his or her pupils, and to give them a written recognition if their grades are improved in each test. Write a method that helps the teacher performing this task. The method gets as input the list of grades of a specific student and determines whether the said pupil deserves the recognition.
Change characters to their successive characters according to the Unicode table.	Write a method that gets as input an array of characters, and changes the array in a way that replaces each character with its successive character according to the Unicode table.	A message that should be sent between financial partners should be encoded. The message includes words, spaces, and dots. Write a method that gets as a parameter a String which represents the message, and returns a coded message in which each letter is replaced by its successive letter in the alphabetical order. The letter "Z" will be replaced with the letter "A". Spaces and dots should not be changed.

Professionele Leergemeenschap Toetsen



Professionele Leergemeenschap Toetsen

- Met elkaar scherp maken wat precies het eindniveau is dat van de leerlingen mag worden verwacht.
- Kennis en expertise delen over toetsing van informatica-onderwerpen.
- Gezamenlijk tot een toets komen (en die afnemen)
- Focus ligt in eerste instantie op domein B (grondslagen) en vervolgens op domein D
- 3 bijeenkomsten komend voorjaar op een centrale plek in Nederland, van 16.00 tot 20.00 uur
 - Donderdag 6 februari
 - Donderdag 2 april
 - Donderdag 11 juni

Meedoen? Mail M.bruggink@tudelft.nl

Vragen