slo / een doordacht curriculum
dat doen we *samen*

Toetsing domein B: Grondslagen

EXAMENPROGRAMMA INFORMATICA HAVO EN VWO VANAF 2019

Toetsing domein B: Grondslagen

Examenprogramma
Informatica havo en vwo vanaf 2019

Augustus 2021

slo

een doordacht curriculum
dat doen we *samen*

Verantwoording

Auteurs:
Paul Bergervoet, Martin Bruggink, Adriaan Gijssen, Jacco Gnodde, Chris Hendriks, Johan Hoekstra, Renske Weeda

Informatie
SLO
Postbus 502, 3800 AM Amersfoort
Telefoon (033) 4840 840
Internet: www.slo.nl
E-mail: info@slo.nl

AN:
1.7976.805

Inhoudsopgave

slo

1. Inleiding

Tot nu was er nog geen leidraad voor het samenstellen van toetsen voor het vak informatica. Elke docent moest hiervoor zelf aan de slag. Daarom zijn we in 2020 als team van informaticadocenten en vakdidactici aan de slag gegaan om een reeks toetsvragen te ontwikkelen voor kerndomein B: Grondslagen van het vernieuwde examenprogramma.
Hiermee kun je als docent op een objectieve manier toetsen. Zo kun je meten of je leerlingen het basisniveau hebben bereikt. Bij andere schoolvakken wordt hiervoor het centraal examen gebruikt. Bij informatica is dat er niet. De auteurs hopen met deze publicatie het toetsen te ondersteunen van leerlingvaardigheden op het gebied van de algoritmiek.
We keken naar de volgende aspecten:

Waar gaat het in de kern om?

Wat is belangrijk voor de leerlingen om te leren, en wat wil je dus toetsen? De eindtermen en voorbeeldspecificaties zijn daarbij de leidraad. Deze hebben we vertaald naar leerdoelen. Voor elk leerdoel vind je hierten minste een toetsvraag.

Hoe kun je variëren in toetsvragen?

We hebben subdoelen geformuleerd op basis van de eindtermen en voorbeeldspecificaties. Denk bijvoorbeeld aan het onderscheid tussen het traceren, begrijpen en opstellen van een stroomdiagram. Dit heeft geleid tot verschillende soorten toetsvragen.

Hoe kun je het niveau van een toetsvraag verhogen of verlagen?

Door scaffolding in te zetten, kun je leerlingen ondersteunen in een stapsgewijze aanpak van een opdracht. Zo zorg je dat de opdracht beter aansluit bij het ontwikkelingsniveau van de leerling. Een mooi voorbeeld van deze aanpak staat beschreven in bijlage A, waarbij één algoritme tot een hele reeks vragen leidt van verschillend niveau. Zo kun je ook goed het onderscheid maken tussen vragen voor havo en vwo.

Misconcepten

Hoe kun je toetsen op veelgemaakte fouten of verkeerde denkwijzen? Dit heeft geleid tot een overzicht van misconcepten. Dit vind je in bijlage B. Sommige vragen zijn gebaseerd op een van deze misconcepten.

Andere kwaliteitsaspecten

Je wilt ook toetsen op efficiëntie. Hoeveel tijd kost het de leerling om de vraag te lezen en beantwoorden en hoe lang kost het beoordelen van een antwoord? Verder moeten beoordelingscriteria eenduidig zijn. De vraag moet ook eenduidig zijn en natuurlijk leesbaar.

In dit document vind je twee bijlagen. Bijlage A geeft een voorbeeld van hoe je kunt variëren in het niveau van toetsvragen, gebaseerd op één algoritme (context). Bijlage B geeft je een overzicht van mogelijke misconcepten bij leerlingen.

Het team bestaat uit:

- Paul Bergevoet (vakdidacticus Universiteit Utrecht, auteur Informatica Actief)
- Martin Bruggink (vakdidacticus TU Delft)
- Adriaan Gijssen (docent vo, auteur Instruct)
- Jacco Gnodde (vakdidacticus VU, docent vo)
- Chris Hendriks (docent HBO)
- Johan Hoekstra (docent vo)
- Renske Weeda (onderzoeker Radboud Universiteit, docent vo)

2. Examenprogramma domein B: Grondslagen

De basis voor de toetsvragen is het examenprogramma.

Subdomein B1: Algoritmen
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
De kandidaat kan verschillende abstracte datastructuren met elkaar vergelijken op elegantie en efficiëntie.

Subdomein B3: Automaten
De kandidaat kan eindige automaten gebruiken voor de karakterisering van bepaalde algoritmen.

Subdomein B4: Grammatica's
De kandidaat kan grammatica's hanteren als hulpmiddel bij de beschrijving van talen.

Een verdere detaillering is te vinden in de voorbeeldspecificaties op de website van SLO.

3. Leerdoelen domein B: Grondslagen

Op basis van de eindtermen in het examenprogramma en de voorbeeldspecificaties hebben we de volgende leerdoelen voor domein B omschreven.

Subdomein B1: Algoritmen
De kandidaat kan:
14.1: een gegeven oplossingsrichting voor een probleem weergeven als een algoritme;
14.2: een gegeven digitaal artefact modelleren met behulp van een algoritme;
14.3: het gedrag van een programma onderzoeken via het onderliggende algoritme;
14.4: een aantal standaardalgoritmen herkennen en gebruiken;
14.5: de correctheid van een gegeven algoritme onderzoeken, en algoritmen (waaronder standaardalgoritmen), vergelijken met betrekking tot efficiëntie.

Subdomein B2: Datastructuren
De kandidaat kan:
15.1: meerdere (abstracte) datastructuren herkennen, onderling vergelijken en beoordelen op toepasbaarheid.

Subdomein B3: Automaten
De kandidaat kan:
16.1. eindige automaten gebruiken voor de karakterisering van bepaalde digitaal artefact;
16.2. bij een gegeven (deterministische) eindige automaat een algoritme construeren.

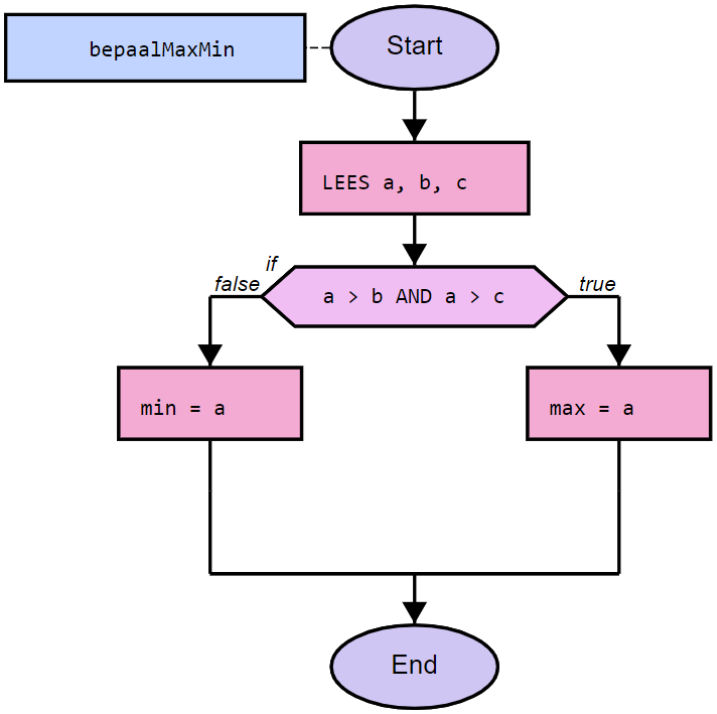
Subdomein B4: Grammatica's
De kandidaat kan:
17.1. het concept grammatica's uitleggen aan de hand van ten minste de termen alfabet, woord en productieregels;
17.2. bij een gegeven grammatica en gegeven woorden vaststellen of de woorden aan de grammatica voldoen;
17.3. (vwo) een gegeven grammatica aanpassen, bijvoorbeeld naar aanleiding van een uitbreiding van de taal.

4. Domein B1 Algoritmen

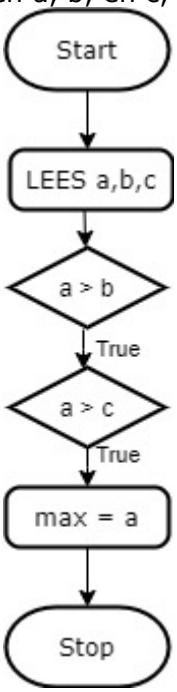
Er zijn 13 voorbeeldtoetsvragen.

- B1.1 Stroomdiagram lezen
- B1.2 Stroomdiagram lezen
- B1.3 Standaardalgoritme herkennen
- B1.4 Standaardalgoritme herkennen
- B1.5 Standaardalgoritme aanpassen
- B1.6 Graafalgoritme toepassen
- B1.7 Algoritme beschrijven
- B1.8 Programma in natuurlijke taal aanpassen
- B1.9 Algoritmes vergelijken
- B1.10 Pseudocode lezen
- B1.11 Pseudocode lezen
- B1.12 Pseudocode lezen

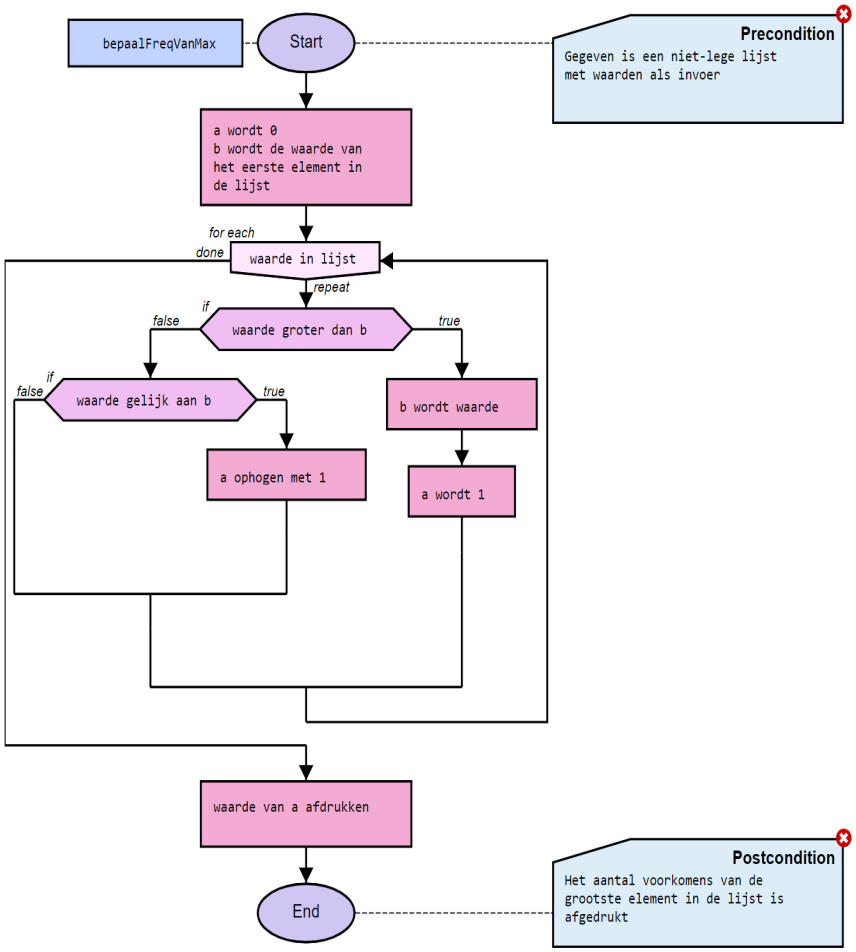
In bijlage A 'Toetsvragen opbouw algoritmen domein B1' vind je een voorbeeld van hoe je op basis van één algoritme een reeks van vragen kunt genereren van verschillende niveaus.

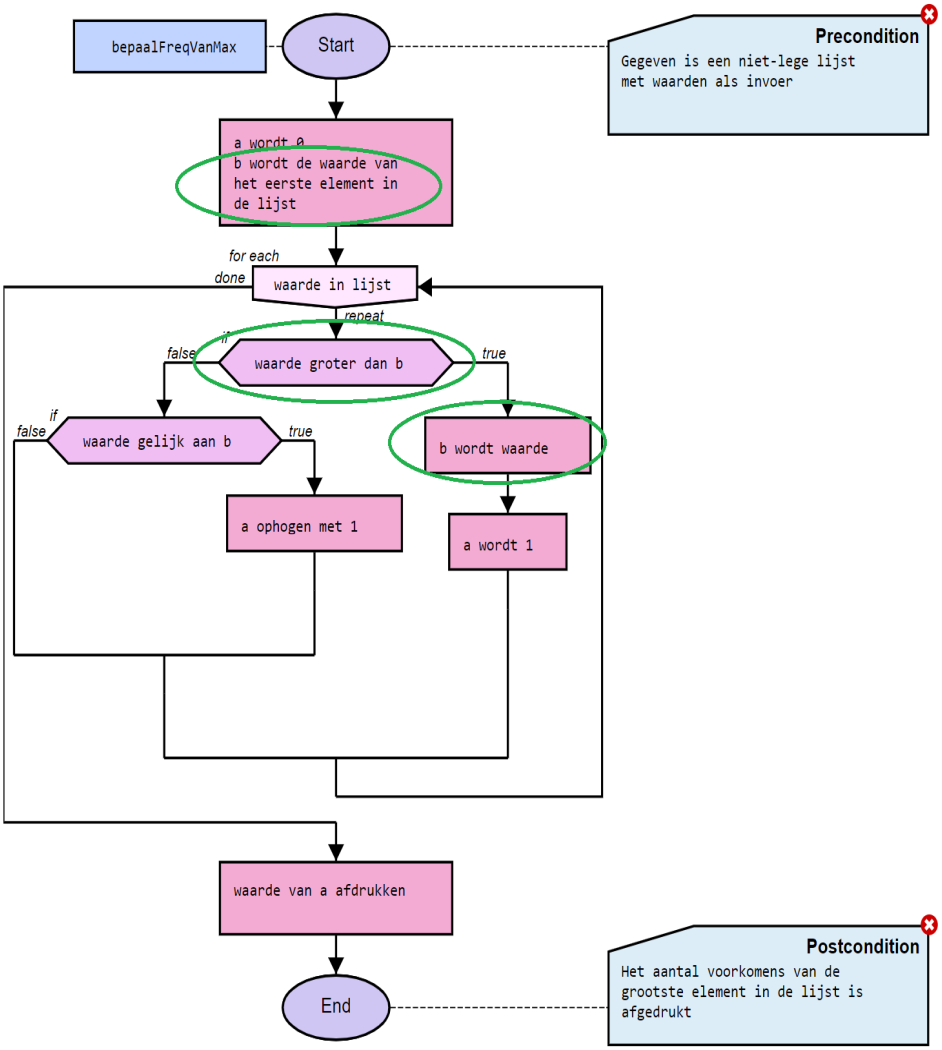
Naam	B1.1 Stroomdiagram lezen
Leerdoel	14.5 De correctheid van een gegeven algoritme onderzoeken.
Vraag	[2pt] Bekijk het onderstaande stroomdiagram. Dit stroomdiagram bevat een algoritme om te bepalen of de waarde van a het maximum of het minimum is van de 3 waarden a, b, en c.  Bepaal of het programma correct is, licht je antwoord toe.
Beoordelingsmodel	[1pt] Het is incorrect. [1pt] Een juiste toelichting: Bijvoorbeeld als: a=2, b=1 en c=3, dan wordt min gelijk aan a, terwijl a niet het minimum is.
Toelichting	Zoek-de-fout-opdrachten zoals deze zijn didactisch vaak effectief. Deze vraag is gebaseerd op een veelvoorkomend misconcept. De situatie dat de waarde van a tussen b en c in ligt, wordt niet meegenomen.
Variatiemogelijkheden	- Deze vraag kan op allerlei stroomdiagrammen worden toegepast, eventueel ook gebaseerd op andere misconcepten. - Andere (standaard)algoritmen gebruiken, zoals bijvoorbeeld min, som, gemiddelde, …

	- Representatie aanpassen: geef pseudocode/natuurlijke taal in plaats van een stroomdiagram. - Meer scaffolding (makkelijker): o Expliciet vragen te traceren voor bepaalde waarden o Bij welke waarden levert het stroomdiagram een onjuist antwoord op? (Je geeft dus aan dat het fout is.) - Corrigeren van fout: stroomdiagram laten aanpassen zodat deze wel juist is. (moeilijker) - Het stroomdiagram als onderdeel binnen een groter algoritme gebruiken. (moeilijker)

Naam	B1.2 Stroomdiagram lezen
Leerdoel	14.5 De correctheid van een gegeven algoritme onderzoeken.
Vraag	[2pt] Bekijk het onderstaande stroomdiagram. Dit stroomdiagram bevat een algoritme om van de 3 waarden a, b, en c, de grootste waarde te bepalen.  Bepaal of het programma correct is, licht je antwoord toe.
Beoordelingsmodel	[1pt] Het is incorrect. [1pt] Een juiste toelichting: bijvoorbeeld als a <= b dan geeft het algoritme geen resultaat.
Toelichting	Deze vraag is gebaseerd op een veelvoorkomend misconcept, namelijk dat een keuze geen *false* heeft.
Variatiemogelijkheden	Deze vraag kan op allerlei stroomdiagrammen worden toegepast.

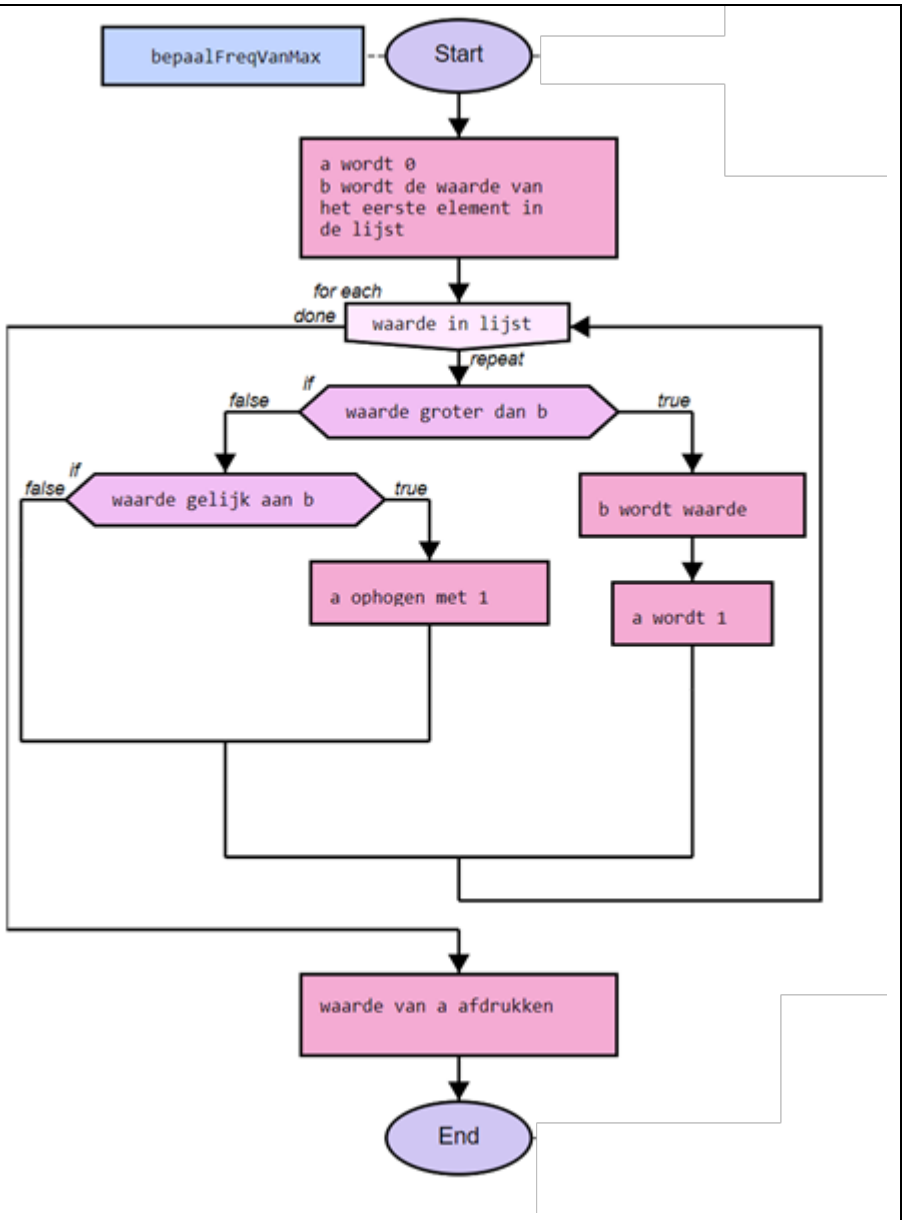
Naam	B1.3 Standaardalgoritme herkennen
Leerdoel	14.4 Een aantal standaardalgoritmen herkennen en gebruiken.
Vraag	[2pt] Bekijk het stroomdiagram hieronder. Dit stroomdiagram bevat een algoritme om te tellen hoe vaak de maximale waarde in een gegeven (niet-lege) lijst voorkomt. Om dit voor elkaar te krijgen wordt onder andere het standaardalgoritme maximum gebruikt. Maximum bepaalt wat de grootste waarde is in een lijst.

	 Omcirkel welke onderdelen van het algoritme horen bij het bepalen van het maximum.
Beoordelingsmodel	[2pt] Alle drie onderdelen juist aangegeven, zoals in onderstaande figuur. 1 pt aftrek voor elke fout, 1pt aftrek voor elke omissie.

	
Toelichting	Het hanteren van standaardalgoritmen zoals het maximum of minimum bepalen, is een expliciet onderdeel van het examenprogramma. Daar gaat deze vraag op in. In plaats van te laten omcirkelen, kun je elk onderdeel een label geven (A, B, C, ..) zodat een leerling daarmee kan aanduiden om welke het gaat.
Variatiemogelijkheden	- Andere (standaard)algoritmen gebruiken, zoals bijvoorbeeld min, som, gemiddelde, sorteren (zie ook B1.7). - Representatie aanpassen: geef pseudocode in plaats van. een stroomdiagram. - Meer scaffolding (makkelijker): ○ Geef het doel van **alle** gebruikte (standaard)subalgoritmes aan (bijvoorbeeld: "Ook het algoritme *tellen* wordt gebruikt om het aantal voorkomens van een getal te bepalen.")

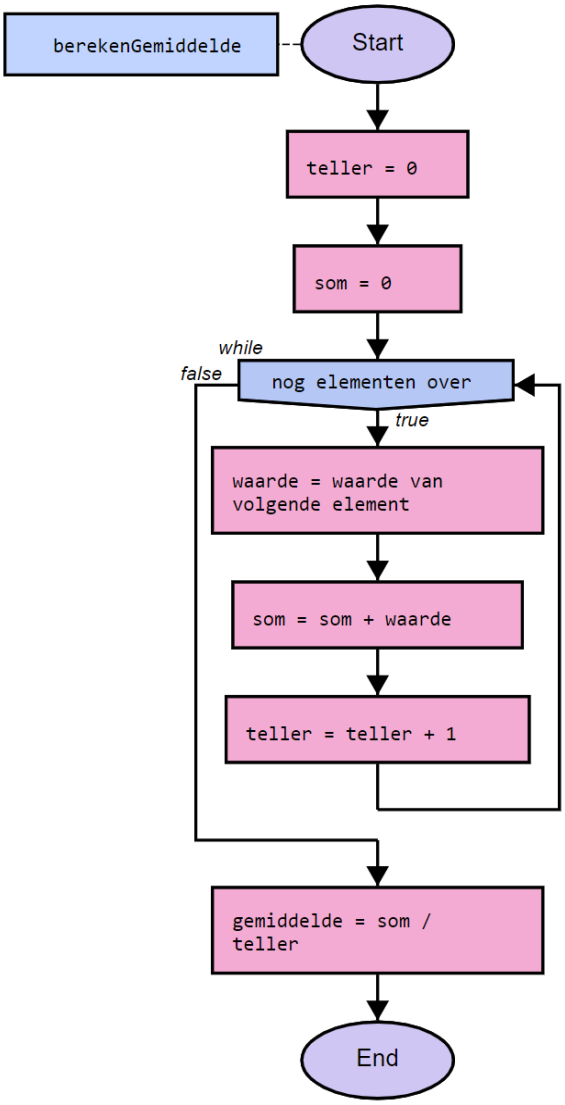
	○ Geef een paar hints of subvragen, bijvoorbeeld: "Bepaal de rol van variabele a.", "Geef een geschikte naam voor variabele a", "Leg uit waarom a wordt opgehoogd", "gegeven de lijst [4, 2, -1, -1, 4], traceer de waarde van a.") - Aangepaste invoerwaarden: pas het probleem/oplossing aan zodat invoer ongeldige waarden kan bevatten (zoals een lege lijst, of alle waarden boven de 100 duiden op een kapot meetinstrument en dienen weggelaten te worden, of moet een foutmelding gegeven worden). (moeilijker) - Aangepaste invoertype: pas het probleem/oplossing aan zodat ander invoerwaarden mogelijk zijn (zoals een lijst met decimale getallen, waarbij alle waarden eerst nog afgerond moeten worden naar gehele getallen voordat het maximum bepaald wordt). (moeilijker) - Aangepaste uitvoertype: waarde opleveren (return) in plaats van afdrukken (print). (moeilijker) - Meerkeuze: de leerling tussen verschillende algoritmes laten kiezen. (makkelijker)

Naam	B1.4 Standaardalgoritme herkennen
Leerdoel	14.3 Het gedrag van een programma onderzoeken via het onderliggende algoritme. 14.2 Een gegeven oplossingsrichting voor een probleem weergeven als een algoritme.
Vraag	[2pt] Bekijk het stroomdiagram hieronder. Dit stroomdiagram bevat een algoritme om te tellen hoe vaak de maximale waarde in een gegeven (niet-lege) lijst voorkomt.

	 Geef aan uit welk van de volgende standaard/basisalgoritmes deze bestaat (meerdere antwoorden mogelijk): a) Tellen b) Minimum bepalen c) Maximum bepalen d) Gemiddelde bepalen e) Sorteren f) Geen van de bovenstaande
Beoordelingsmodel	Antwoord: [1pt] a) Tellen 1. [1pt] c) Maximum bepalen

	1pt aftrek voor elk verkeerd antwoord.
Toelichting	Mogelijke misconcepten of moeilijkheden zijn: - verwisselen min en max plan; - niet begrijpen van geneste plannen in plaats van het achter elkaar uitvoeren van plannen.
Variatiemogelijkheden	- Andere (standaard)algoritmen gebruiken, zoals min, som of gemiddelde. - Representatie aanpassen: geef pseudocode in plaats van een stroomdiagram. Meer scaffolding (makkelijker): o Geef het doel van alle gebruikte (standaard)subalgoritmes aan, bijvoorbeeld: "Ook het algoritme *tellen* wordt gebruikt om het aantal voorkomens van een getal te bepalen.") o Geef een paar hints of subvragen, bijvoorbeeld: "Bepaal de rol van variabele a.", "Geef een geschikte naam voor variabele a", "Leg uit waarom a wordt opgehoogd.")

Naam	B1.5 Standaardalgoritme aanpassen
Leerdoel	14.3 Het gedrag van een programma onderzoeken via het onderliggende algoritme. 14.2 Een gegeven oplossingsrichting voor een probleem weergeven als een algoritme.
Vraag	[6pt] Bekijk het onderstaande algoritme. Dit stroomdiagram bevat een algoritme om het gemiddelde van alle waarden in de lijst te berekenen.



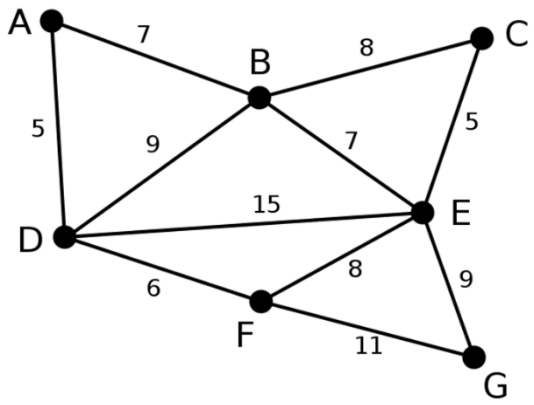
a) [2pt] Het algoritme in het stroomdiagram wordt uitgevoerd met de volgende invoer: [3, 0, -5, 7]. Bepaal de waarden van de volgende vier variabelen na afloop: teller, som, waarde, gemiddelde.

b) [1pt] Geef een voorbeeld voor een invoer waarbij het algoritme niet correct zal werken.

c) [3pt] Geef aan welke aanpassing aan het algoritme nodig is zodat alleen de waarden die groter dan of gelijk aan nul zijn worden meegenomen in de berekening van het gemiddelde. Alle waarden kleiner dan nul moeten dus worden genegeerd in de berekening.

Beoordelingsmodel	a) [2pt, 1 punt aftrek voor elke fout antwoord]: Teller is 4, som is 10, waarde is 7, gemiddelde is 2,5 b) [1pt] Een lege lijst c) [3 pt, 1 pt aftrek voor elke fout]: 1. [1pt] Binnen de loop moet na het bepalen van de waarde 2. [1pt] moet nog een conditie komen: als waarde >=0 dan wordt som uitgevoerd. 3. [1pt] De teller moet niet worden opgehoogd worden.
Toelichting	Een standaard opbouw (van makkelijk naar moeilijk) voor het scaffolding van dit soort vragen kan zijn: a) Wat is het resultaat voor een specifieke invoer? b) Voor welke invoer waarden is het algoritme incorrect? c) Geef aan waar de fout zit. d) Corrigeer de fout.
Variatiemogelijkheden	- Andere (standaard)algoritmen gebruiken, zoals bijvoorbeeld min, max, som. - Representatie aanpassen: geef pseudocode in plaats van een stroomdiagram. - Deze vraag kan op allerlei (standaard) algoritmes worden toegepast: minimum/maximum bepalen, aantal elementen tellen. - Meer scaffolding (makkelijker): bijvoorbeeld de plek van de aanpassing voorgeven als leeg blokje in stroomdiagram, waarbij alleen de conditie nog ingevuld dient te worden. - Context toevoegen: gemeten regenval (>=0) , gemeten gewicht (>=0), temperatuur (alle waarden), verschil in gewicht (alle waarden) (moeilijker).

Naam	B1.6 Graafalgoritme toepassen
Leerdoel	14.4 Een aantal standaardalgoritmen herkennen en gebruiken.
Vraag	[3pt] Bekijk de volgende graaf.

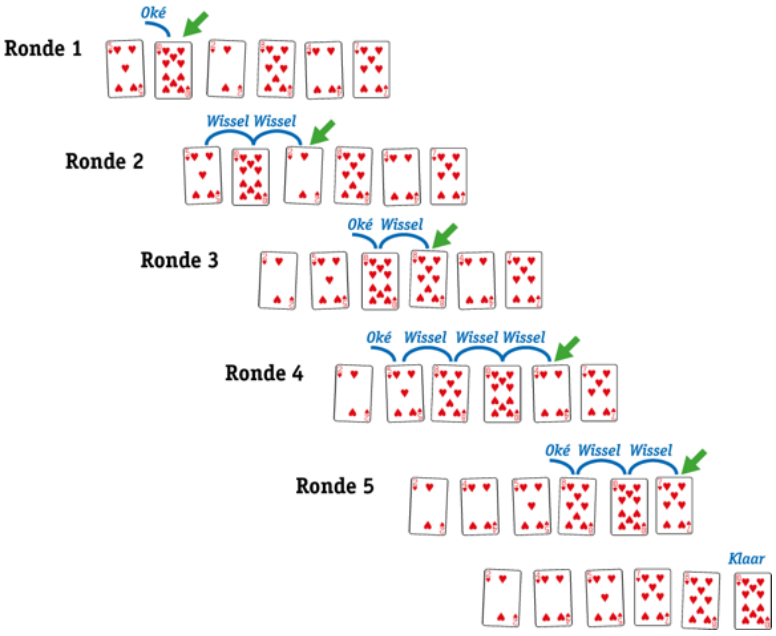
<table>
<tr>
<td></td>
<td>


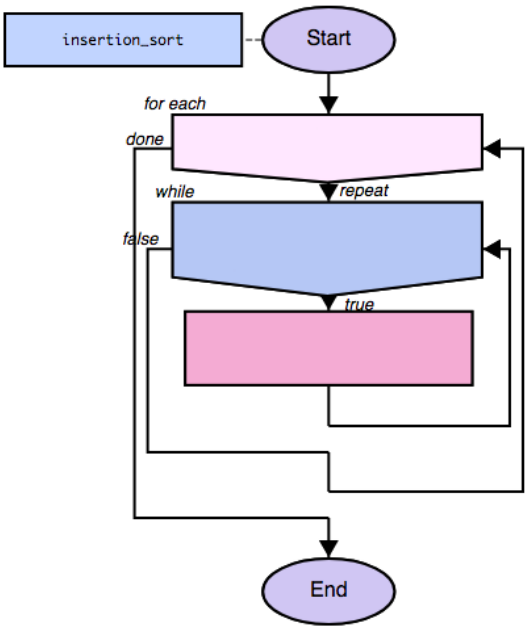
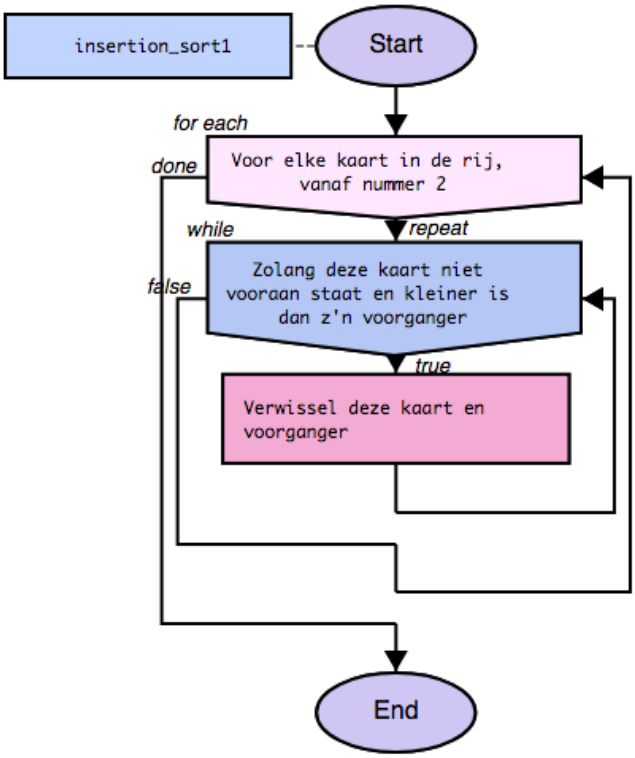
Bron: Wikipedia

We zijn op zoek naar het kortste pad tussen knoop A en knoop E. Pas het kortstepadalgoritme van Dijkstra toe op deze graaf om het kortste pad tussen deze twee punten te bepalen. Beschrijf elke stap in het algoritme. Gebruik eventueel de onderstaande tabel.

<table>
<tr><th>Plaats</th><th>Kortste afstand tot A</th><th>Vorige plaats</th></tr>
<tr><td>A</td><td>0</td><td>-</td></tr>
<tr><td>B</td><td>∞</td><td></td></tr>
<tr><td>C</td><td>∞</td><td></td></tr>
<tr><td>D</td><td>∞</td><td></td></tr>
<tr><td>E</td><td>∞</td><td></td></tr>
<tr><td>F</td><td>∞</td><td></td></tr>
<tr><td>G</td><td>∞</td><td></td></tr>
</table>
</td>
</tr>
<tr>
<td>Beoordelingsmodel</td>
<td>[1pt] Kortste pad: A naar E: 14
[2pt, 1 punt aftrek voor elke fout]:
A: 0
A->D, D: 5
A->B, B: 7
D->F, F: 11
D->E, E: 20
B->C, C: 15
B->E, E: 14
F->G, G: 22</td>
</tr>
<tr>
<td>Toelichting</td>
<td>Het examenprogramma biedt keuze tussen enkele toepassingen zoals zoeken/sorteren/graafalgoritmen/datacompressie.</td>
</tr>
</table>

	Het is dus niet verplicht om bijvoorbeeld het kortstepadalgoritme van Dijkstra te toetsen.
Variatiemogelijkheden	Variaties zijn onder meer: - Andere (standaard)algoritmen voor grafen gebruiken, zoals bijvoorbeeld nearest neighbor, brute-force, of een geheel nieuw algoritme, mits goed omschreven. - Ander soort graafprobleem, zoals minimaal opspannende boom (met Prim, Kruskal, brute force). - Meer scaffolding (makkelijker): ○ Het algoritme voorgegeven in natuurlijke taal of middels een stroomdiagram. ○ Geef de eerste stap cadeau, zodat de leerling daarop kan voortborduren.

Naam	B1.7 Algoritme beschrijven
Leerdoel	14.1 Een gegeven oplossingsrichting voor een probleem weergeven als een algoritme.
Vraag	[3pt] Hieronder zie je hoe stap voor stap een rij van zes kaarten wordt gesorteerd met behulp van *insertion sort.* 

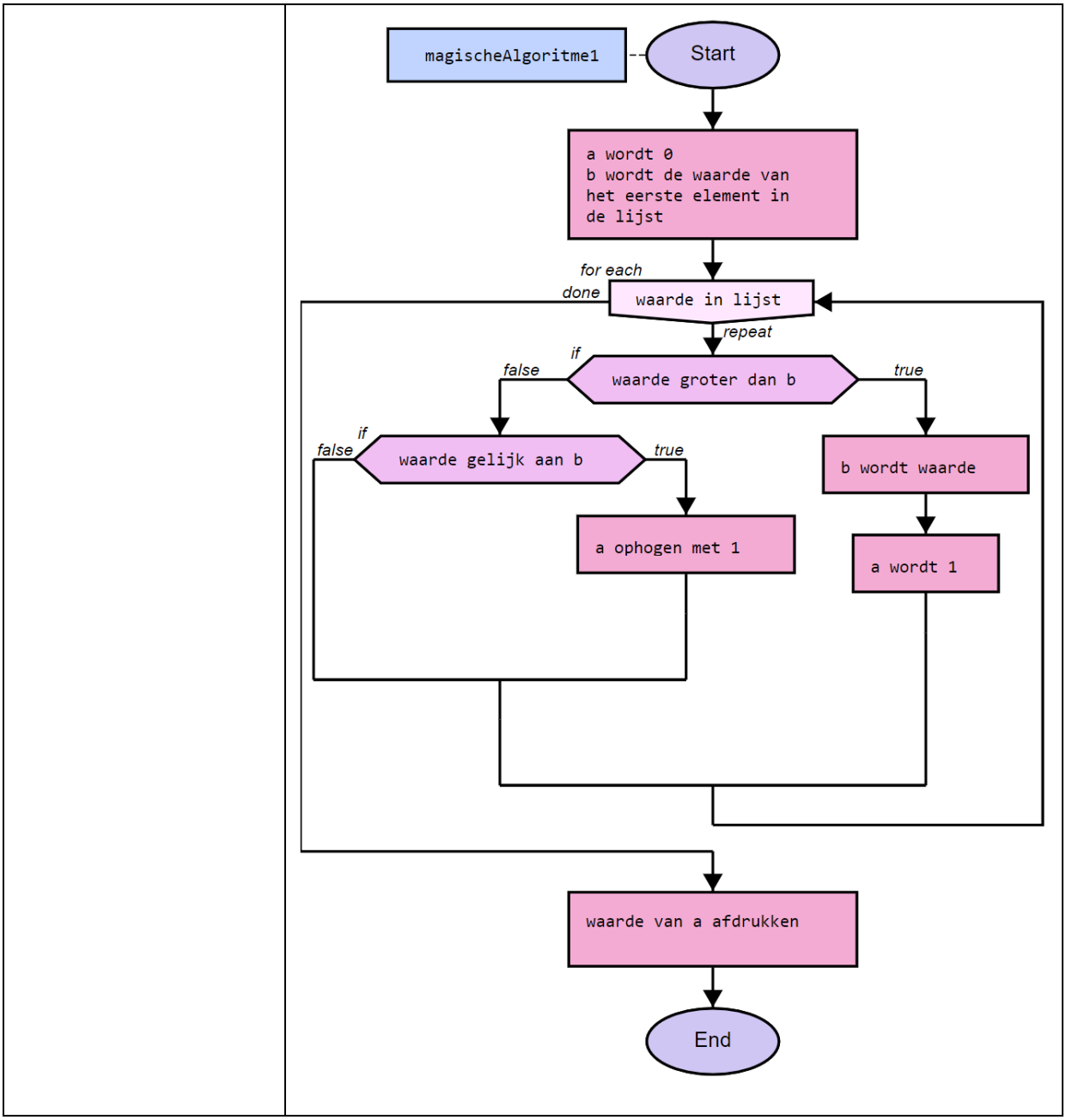
	Maak een stroomdiagram dat dit algoritme beschrijft. Gebruik daarvoor onderstaande basis en vul de lege vakjes in. 
Beoordelingsmodel	

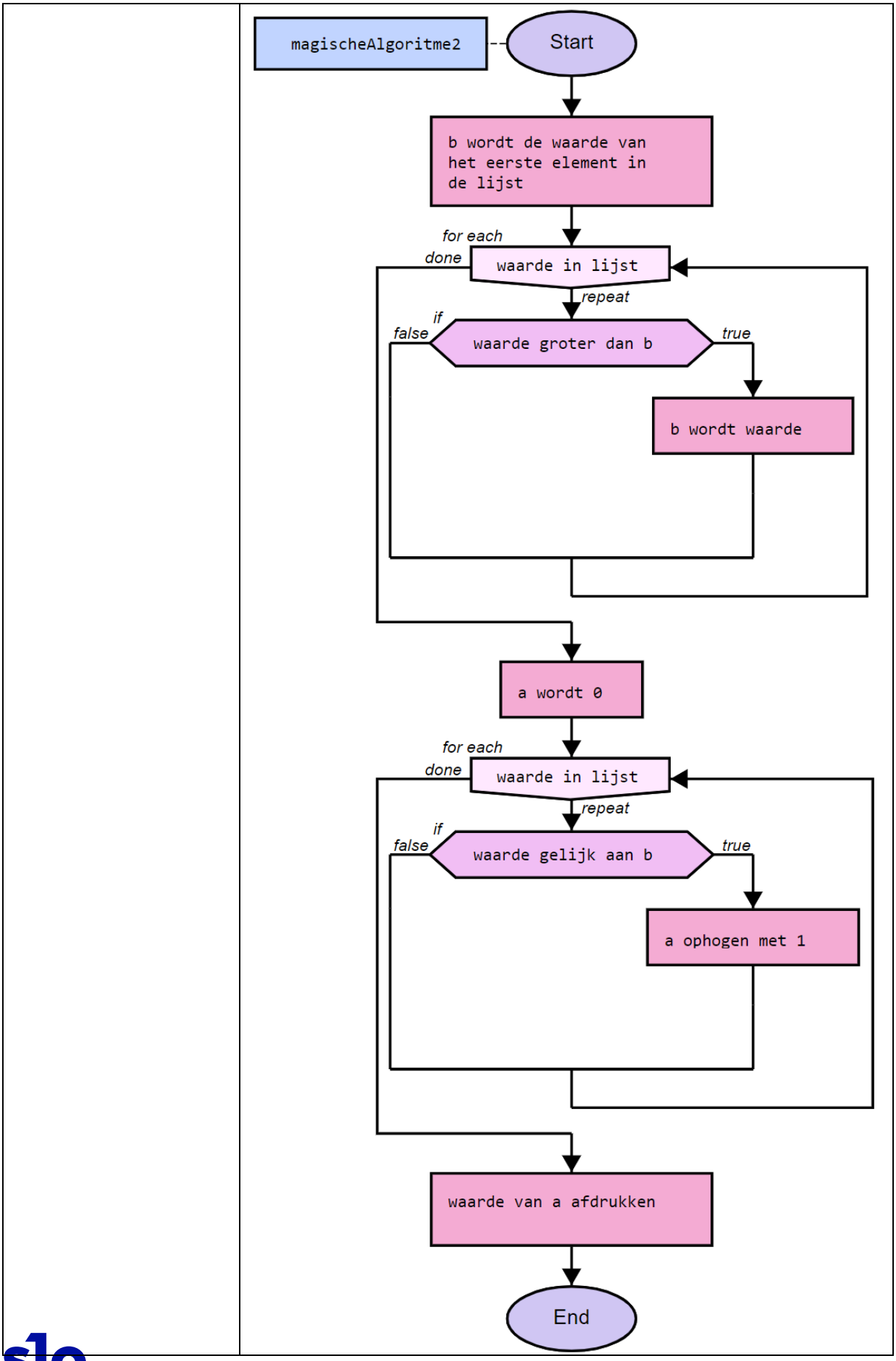
	[3pt] 1 punt voor elk correct gevuld vakje.
Toelichting	Er is bewust voor kaarten gekozen in plaats van getallen. Dat maakt het concreter.
Variatiemogelijkheden	Variaties zijn onder meer: - De mogelijke instructies (eventueel met extra afleiders) geven en de leerling laten aangeven op welk plek elk instructie hoort. (makkelijker) - Een deel van de instructies invullen. (makkelijker) - Meerkeuze: de leerling tussen verschillende stroomdiagrammen laten kiezen. (makkelijker) - Geen template geven. (moeilijker)

Naam	B1.8 Programma in natuurlijke taal aanpassen
Leerdoel	14.1 Een gegeven oplossingsrichting voor een probleem weergeven als een algoritme.
Vraag	[3pt] Bekijk het volgende algoritme, die bedoeld is om een lijst met getallen te sorteren van klein naar groot. 1. Vergelijk het eerste en tweede getal in de lijst. 2. Als het eerste getal groter is dan het tweede, verwissel dan het eerste en tweede getal in de lijst. 3. Herhaal stap 1 en 2 voor het tweede en derde getal, voor het derde en vierde getal, enz. totdat je het eind van de lijst hebt bereikt. a) [1pt] Geef het resultaat als je dit algoritme uitvoert op de volgende lijst met de getallen: 18, 4, 15, 9, 31, 27 (in die volgorde). b) [2pt] Het algoritme is nog niet compleet. Pas het algoritme aan zodat het in alle gevallen een gesorteerde lijst oplevert.
Beoordelingsmodel	a) [1pt] 4, 15, 9, 18, 27, 31 b) [2pt] Dit is Bubblesort, maar de lijst wordt maar één keer doorlopen voor het eerste element. Je moet dus nog toevoegen dat het algoritme steeds moet herhalen voor elk element in de lijst: • Stap 4: • Loop n-2 maal opnieuw door de rij, waarbij n het aantal elementen in de lijst is. Elke keer herhaal je stappen 1 t/m 3. De eerste keer ga je door tot

	het voorlaatste element, omdat het laatste element het grootste in de rij was. De keer daarop negeer je de twee laatste elementen, enzovoort.
Toelichting	
Variatiemogelijkheden	Variaties zijn onder meer: - Geef de algoritmeomschrijving in pseudocode of een stroomdiagram. - Leerlingen vragen om bij deel a alle tussenstappen te laten zien (makkelijker). - Zonder scaffolding (moeilijker): deel a overslaan en niet laten traceren

Naam	B1.9 Algoritmes vergelijken
Leerdoel	14.5 De correctheid van een gegeven algoritme onderzoeken.
Vraag	[2pt] Bekijk de twee stroomdiagrammen hieronder. Beide algoritmes hebben hetzelfde doel. Je mag ervan uit gaan dat er een niet-lege lijst met waarden als invoer is. a) [1pt] Vat samen wat het doel is van beide algoritmes. Beschrijf dit in één zin. b) [1pt] Welke van de twee algoritmes doorloopt meestal de minste stappen? Onderbouw je keuze.

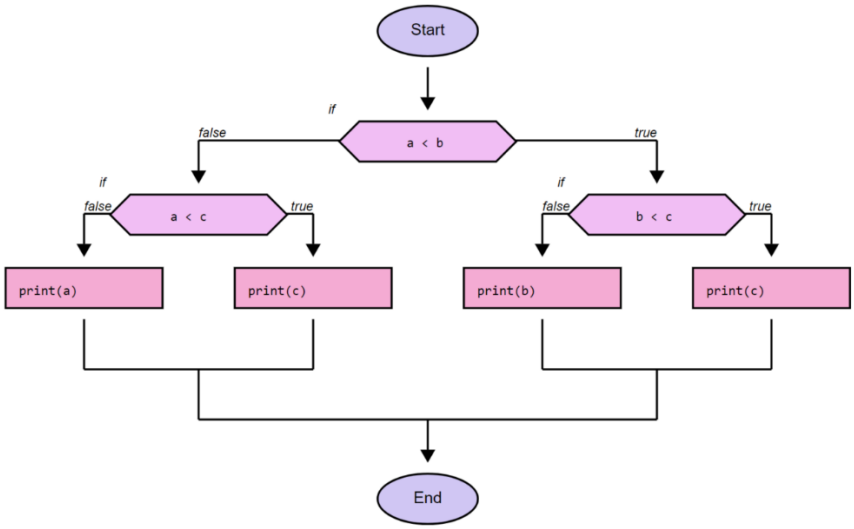

magischeAlgoritme1
Start
a wordt 0
b wordt de waarde van het eerste element in de lijst
for each
done
waarde in lijst
repeat
if
false
waarde groter dan b
true
if
false
waarde gelijk aan b
true
b wordt waarde
a ophogen met 1
a wordt 1
waarde van a afdrukken
End


magischeAlgoritme2
Start
b wordt de waarde van het eerste element in de lijst
for each
done
waarde in lijst
repeat
if
false
waarde groter dan b
true
b wordt waarde
a wordt 0
for each
done
waarde in lijst
repeat
if
false
waarde gelijk aan b
true
a ophogen met 1
waarde van a afdrukken
End

Beoordelingsmodel	a) [1pt] Het doel van beide algoritmes is te tellen hoe vaak de maximale waarde in een gegeven niet-lege lijst voorkomt. b) [1pt] Het eerste algoritme heeft maar 1 herhalingslus en doorloopt daarom minder stappen.
Toelichting	
Variatiemogelijkheden	- Representatie aanpassen: geef pseudocode in plaats van een stroomdiagram. - Andere (standaard)algoritmen gebruiken, zoals min, som of gemiddelde. - Meer scaffolding (makkelijker): ○ Geef het doel van **ieder** gebruikt (standaard)subalgoritme aan, bijvoorbeeld: "Ook het algoritme *tellen* wordt gebruikt om het aantal voorkomens van een getal te bepalen." ○ Geef een paar hints of subvragen, bijvoorbeeld: "Bepaal de rol van variabele a.", "Geef een geschikte naam voor variabele a", "Leg uit waarom a wordt opgehoogd", "gegeven de lijst [4, 2, -1, -1, 4], traceer de waarde van a") - Combineer andere (standaard)algoritme tot een complexere algoritme. (moeilijker)

Naam	B1.10 Pseudocode lezen
Leerdoel	14.3 Het gedrag van een programma onderzoeken via het onderliggende algoritme.
Vraag	[2pt] Bekijk onderstaande pseudocode. `IF (getal1 > getal2) THEN` `  IF (getal1 > getal3) THEN` `  PRINT getal1` `  ELSE` `  PRINT getal3` `  ENDIF` `ELSE` `  IF (getal2 >= getal3) THEN` `  PRINT getal2` `  ELSE` `  PRINT getal3` `  ENDIF` `ENDIF`

	a) [1pt] Wat is de uitkomst van het programma bij de volgende invoer: getal1=3, getal2=5 en getal3=1 b) [1pt] Vat in een paar woorden samen wat het doel is van dit programma.
Beoordelingsmodel	a) [1pt] Het getal 5 wordt geprint. b) [1pt] Het doel van dit programma is om uit 3 getallen de hoogste waarde te bepalen en deze af te drukken.
Toelichting	Er is bewust voor een opbouw gekozen. Eerst moet de leerlingen het programma traceren voor een bepaalde invoer (makkelijk). Daarna moet de leerling de algemene werking beschrijven (moeilijker). De leerlingen hebben soms moeite met het verschil tussen < en >, en met het verschil tussen > en >=.
Variatiemogelijkheden	- Je kunt het algoritme ook als stroomdiagram geven. - Vraag de leerling dit om te zetten naar een stroomdiagram. - Laat de leerling eerst de resultaten bepalen op basis van specifieke waarden van getal1, getal2 en getal3 (makkelijker). - Alleen > of alleen >= tekens (makkelijker). - Gebruik een context, bijvoorbeeld de straal of oppervlakte van een cirkel of de inhoud van een blok (moeilijker). - Meerkeuzevraag in plaats van een open vraag (makkelijker). - Een programma dat de kleinste van 3 ingevoerde waarden bepaalt en print (in het algoritme alle > vervangen door <); of > vervangen door >= (alternatief, even moeilijk). - Vervang print door return, geef parameters mee (moeilijker). - Zonder getallen, maar met iets meer context, bijvoorbeeld met boodschappenlijstjes of namen. Dat maakt het tastbaarder voor leerlingen, waardoor ze gemiddeld gemotiveerder zijn.
Variant 1	Hieronder vind je een variant van het programma met twee variabelen (makkelijker).

	Bekijk onderstaande pseudocode. `IF (getal1 > getal2) THEN` `  PRINT getal1` `ELSE` `  PRINT getal2` `ENDIF` Het doel van dit programma is om uit twee getallen de hoogste waarde te bepalen en deze af te drukken.
Variant 2	Bekijk onderstaand stroomdiagram.  Vat in een paar woorden samen wat het doel is van het algoritme in dit stroomdiagram is. Antwoord: [1pt] Dit programma bepaalt uit drie waarden de grootste waarde en drukt die af.

Naam	B1.11 Pseudocode lezen
Leerdoel	14.3 Het gedrag van een programma onderzoeken via het onderliggende algoritme.
Vraag	[2pt] Bekijk onderstaande pseudocode. `IF (getal1 > getal2) THEN` `  IF (`*`VERGELIJKING1`*`) THEN` `  PRINT getal1`

	`` ELSE PRINT getal3 ENDIF ELSE IF (VERGELIJKING2) THEN PRINT getal2 ELSE PRINT getal3 ENDIF ENDIF ``` In dit programma worden drie getallen met elkaar vergeleken. Het doel van het programma is om het grootste getal af te drukken. 1. [1pt] Wat moet er staan op de plek waar nu VERGELIJKING1 staan? a) getal1 > getal2 b) getal1 > getal3 c) getal2 < getal3 d) getal2 > getal3 2. [1pt] Wat moet er staan op de plek waar nu VERGELIJKING2 staan? a) getal1 > getal2 b) getal1 > getal3 c) getal2 < getal3 d) getal2 > getal3
Beoordelingsmodel	Antwoord 1. [1punt] B. getal1>getal3 2. [1punt] D. getal2>getal3
Toelichting	
Variatiemogelijkheden	- Gebruik >= in plaats van >. - Vervang > met < om het kleinste te bepalen.

Naam	B1.12 Pseudocode lezen
Leerdoel	14.3 Het gedrag van een programma onderzoeken via het onderliggende algoritme.
Vraag	[1pt] Het doel van het onderstaande programma is om van 3 getallen de hoogste te bepalen en dat af te drukken.

	<table><tr><td>1. 2. 3. 4. 5. 6. 7. 8. 9. 10. 11. 12. 13.</td><td>IF (getal1 > getal2) THEN IF (getal2 > getal3) THEN PRINT getal1 ELSE PRINT getal3 ENDIF ELSE IF (getal2 > getal3) THEN PRINT getal2 ELSE PRINT getal3 ENDIF ENDIF</td></tr></table> Door een fout vindt dit programma niet altijd de hoogste waarde. Door welke regel code uit het programma komt dat? Noteer het regelnummer.
Beoordelingsmodel	[1pt] regel 2. IF (getal2 > getal3) THEN De conditie moet zijn: getal1>getal3.
Toelichting	
Variatiemogelijkheden	Andere variaties zijn: - Leg uit waarom de hoogste waarde niet wordt gevonden. - Herschrijf de code zodat het hoogste getal wel wordt gevonden. - Herschrijf het programma zodat in plaats van het hoogte getal het laagste getal wordt afgebeeld.

5. Domein B2 Datastructuren

We leggen de nadruk op de verschillen tussen rijen, lijsten en bomen (alleen vwo) in relatie tot toepassingen zoals zoeken en het toevoegen of verwijderen van elementen.
Er zijn 14 voorbeeld toetsvragen:

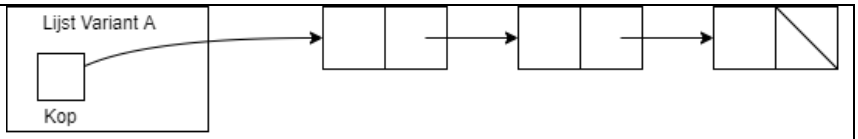
- B2.1 Functie op een lijst
- B2.2 Lijst met referentie naar het laatste element

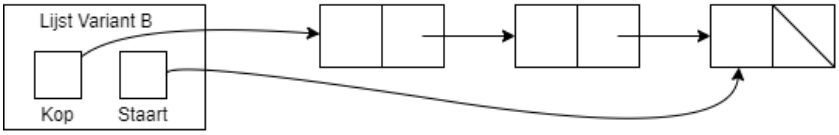
- B2.3 Bomen
- B2.4 Functie op een boom
- B2.5 Rijen
- B2.6 Lijsten
- B2.7 Rijen en lijsten
- B2.8 Record
- B2.9 Invoegen element in een lijst
- B2.10 Invoegen element in een lijst
- B2.11 Invoegen element in een lijst of rij
- B2.12 Binair zoeken in rijen en lijsten
- B2.13 Binair en lineair zoeken
- B2.14 Binair zoeken in een rij

Naam	B2.1 Functie op een lijst
Leerdoel	15.1 De kandidaat kan meerdere (abstracte) datastructuren herkennen, onderling vergelijken en beoordelen op toepasbaarheid.
Vraag	[1pt] Hieronder zie je een beschrijving van de functie *mysterie* dat wordt gebruikt voor een lijst. Als invoer krijgt de functie een lijst met gehele getallen en een geheel getal. Ga er vanuit dat: • *lijst.kop* het eerste element van de lijst oplevert; • *huidige_knoop.waarde* de waarde van het huidige element in de lijst oplevert; • *huidige_knoop.volgende* het volgende element uit de lijst oplevert. `FUNCTION mysterie (lijst, getal)` `  huidige_knoop = lijst.kop` `  vorige_knoop = null` `  WHILE (huidige_knoop != null AND huidige_knoop.waarde != getal) DO` `  vorige_knoop = huidige_knoop` `  huidige_knoop = huidige_knoop.volgende` `  ENDWHILE` `  RETURN vorige_knoop` `ENDFUNCTION` Welke van de volgende stellingen omschrijft het doel van de mysteriemethode het beste?

	a) Het levert het element op vóór het element dat *getal* bevat, of het levert *null* op als *getal* in de kop zit, of het levert het laatste element van de lijst op als *getal* niet wordt gevonden. b) Het levert het element op dat *getal* bevat, of het levert het laatste element van de lijst op als *getal* niet is gevonden. c) Het levert het element op dat *getal* bevat, of het levert *null* op als *getal* niet wordt gevonden. d) Het levert het element op vóór het element dat *getal* bevat, of het levert *null* op als getal in de kop zit, of het levert *null* op als *getal* niet wordt gevonden.
Beoordelingsmodel	[1pt] Antwoord: A
Toelichting	Dit is een vrij complexe functie met meerdere uitzonderingen. Het dient echter als voorbeeld, waarbij de leerling op basis van een gegeven functie en een datastructuur zoals rij, lijst of boom moet aangeven wat het doel van de functie is.
Variatiemogelijkheden	Het is mogelijk om deze vraag voor allerlei functies te stellen en op verschillende datastructuren. Bijvoorbeeld: het tellen, toevoegen, verwijderen, zoeken van een element in een rij/lijst/boom. Zie ook vraag B2.4.

Naam	B2.2 Lijst met referentie naar het laatste element
Leerdoel	15.1 De kandidaat kan meerdere (abstracte) datastructuren herkennen, onderling vergelijken en beoordelen op toepasbaarheid.
Vraag	[2pt] Bekijk de volgende twee varianten van een lijst en de bijbehorende figuren daaronder. Variant A: Een lijst met een referentie naar het eerste element van de lijst. Daarnaast bevat elk element een referentie naar het volgende element in de lijst. Variant B: Een lijst met een referentie naar het eerste element van de lijst *en* een referentie naar het laatste element van de lijst. Daarnaast bevat elk element bevat een referentie naar het volgende element in de lijst.





Bij sommige operaties zal variant A sneller werken ten opzichte van variant B, of ze zijn even snel. Geef bij elk van de vier stellingen hieronder aan welke waar is:

	Variant A is sneller	Variant B is sneller	Varianten A en B zijn even snel
aangeven of een gegeven element in de lijst voorkomt			
verwijderen van het laatste element van de lijst			
een gegeven element toevoegen aan het begin van de lijst			
een gegeven element toevoegen aan het eind van de lijst			

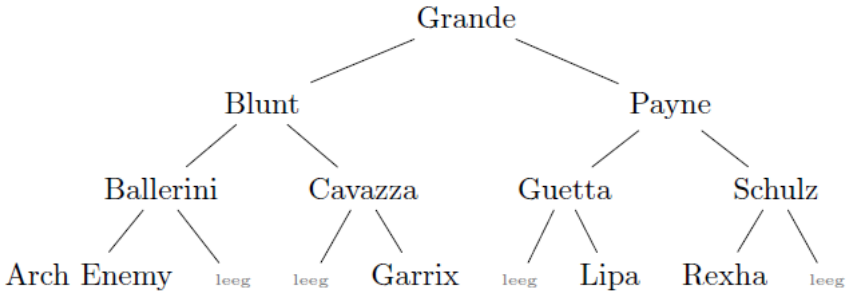
Beoordelingsmodel

[2punten] Antwoord: 4 antwoorden:
[-1pt per incorrect beantwoorde stelling]

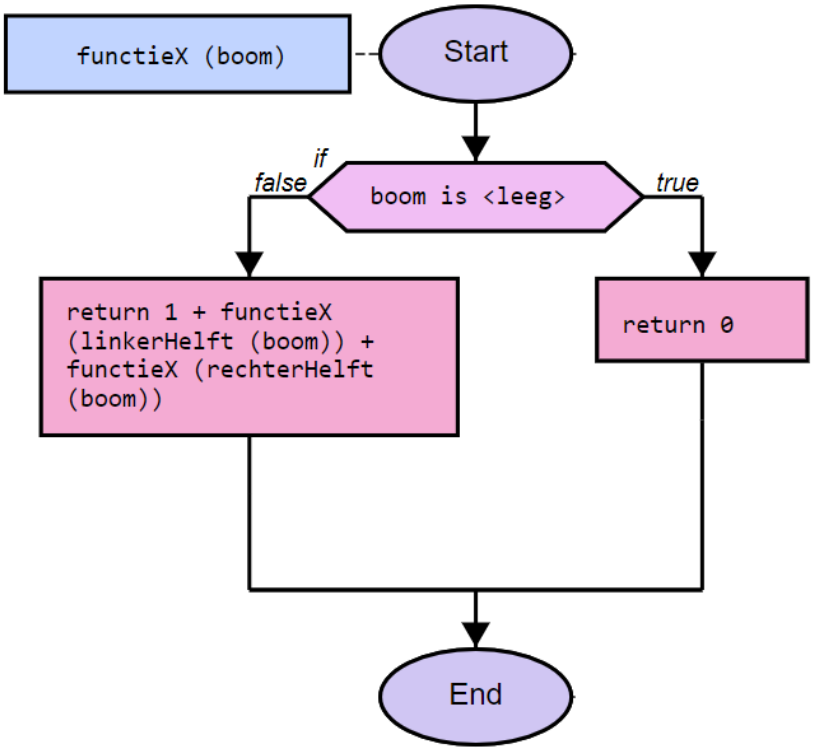
	Variant A is sneller	Variant B is sneller	Varianten A en B zijn even snel

	aangeven of een gegeven element in de lijst voorkomt			X
	verwijderen van het laatste element van de lijst		X	
	een gegeven element toevoegen aan het begin van de lijst			X
	een gegeven element toevoegen aan het eind van de lijst		X	
Toelichting				
Variatiemogelijkheden	Als je de figuur weglaat, wordt het abstracter en moeilijker. Je kunt hetzelfde doen voor een rij. Bij een rij zal echter geen van de operaties wezenlijk verschil maken.			

Naam	B2.3 Bomen
Leerdoel	15.1 De kandidaat kan meerdere (abstracte) datastructuren herkennen, onderling vergelijken en beoordelen op toepasbaarheid.
Vraag	Hieronder zie je een boom met daarin de namen van artiesten. Door de gegevens in deze boom op te slaan wordt zoeken naar gegevens efficiënter omdat er gemiddeld genomen minder vergelijkingen nodig zijn om een naam te vinden.

	 a) [2pt] In maximaal hoeveel vergelijkingen weet je of een artiest voorkomt in de boom? b) [1pt] Wat is het nadeel van gegevens opslaan in zo'n boom?
Beoordelingsmodel	a) [2pt] 7 (de eerste drie lagen eerst == dan < en dan in de bladen == dus 3x2+1. b) [1pt] Als je een boom laat groeien zul je hem regelmatig of bij iedere toevoeging, of verwijdering van een knoop moeten balanceren. Dat kost tijd.
Toelichting	Bomen zijn alleen onderdeel van het vwo-examenprogramma. Veelvoorkomende fout: Het zou goed kunnen dat leerlingen als incorrecte antwoord 4 geven, omdat alleen het aantal == vergelijkingen beschouwd worden.
Variatiemogelijkheden	(moeilijker) Je kunt de vraag ook in het algemeen stellen, zonder concreet voorbeeld. Daarmee wordt het abstracter.

Naam	B2.4 Functie op een boom
Leerdoel	15.1 De kandidaat kan meerdere (abstracte) datastructuren herkennen, onderling vergelijken en beoordelen op toepasbaarheid.
Vraag	Hieronder zie je een stroomdiagram voor een functie met de naam functieX. Deze functie heeft één parameter waarin een datastructuur van type boom wordt meegegeven.

	 [1pt] Vat in je eigen woorden in één zin samen wat het doel is van deze functie.
Beoordelingsmodel	Antwoord: [1punt] Het telt het aantal elementen in de boom.
Toelichting	Bomen zijn alleen onderdeel van het vwo-examenprogramma. Daarnaast gaat het om een recursieve functie, dat is alleen voor vwo.
Variatiemogelijkheden	Zie vraag B2.1

Naam	B2.5 Rijen
Leerdoel	15.1 De kandidaat kan meerdere (abstracte) datastructuren herkennen, onderling vergelijken en beoordelen op toepasbaarheid.
Vraag	[1pt] We maken onderscheid in rijen en lijsten. Welke van de onderstaande uitspraken over rijen is **niet** waar? Meerdere antwoorden mogelijk. a) Een rij kan meerdere elementen bevatten. b) Een rij heeft een index. c) De gegevens in een rij staan altijd achter elkaar in het geheugen opgeslagen.

	d) Een rij kan data van verschillende typen opslaan.
Beoordelingsmodel	[1pt] Antwoord: d. [1pt aftrek voor elke fout antwoord]
Toelichting	Het gaat om het theoretische verschil tussen rijen (elementen staan achter elkaar in het geheugen) en lijsten (elk element verwijst weer naar het volgende element). Verwarrend kan zijn dat dit onderscheid in programmeeromgevingen niet altijd helder is. In Python kun je bijvoorbeeld elementen van een lijst opvragen met behulp van een index.
Variatiemogelijkheden	Zie ook vraag B2.6 over lijsten.

Naam	B2.6 Lijsten
Leerdoel	15.1 De kandidaat kan meerdere (abstracte) datastructuren herkennen, onderling vergelijken en beoordelen op toepasbaarheid.
Vraag	[1pt] We vergelijken rijen en lijsten. Welke van de onderstaande uitspraken over lijsten is **niet** waar? Meerdere antwoorden mogelijk. a) Een lijst kan meerdere elementen bevatten. b) Een lijst heeft een index. c) De data in een lijst staan altijd achter elkaar in het geheugen opgeslagen. d) Een lijst kan data van verschillende typen opslaan. e) In een lijst kun je sneller elementen tussenvoegen dan in een rij.
Beoordelingsmodel	[1pt] Antwoord: b en c. [1pt aftrek voor elke fout antwoord, 1pt aftrek voor elk ontbrekend antwoord.]
Toelichting	Het gaat om het theoretische verschil tussen rijen (elementen staan achter elkaar in het geheugen) en lijsten (elk element verwijst weer naar het volgende element). Verwarrend kan zijn dat dit onderscheid in programmeeromgevingen niet altijd helder is. In Python kun je bijvoorbeeld elementen van een lijst opvragen met behulp van een index. Ook kun je in Python elementen van verschillende typen opslaan.
Variatiemogelijkheden	Zie ook vraag B2.5 over rijen.

Naam	B2.7 Rijen en lijsten
Leerdoel	15.1 De kandidaat kan meerdere (abstracte) datastructuren herkennen, onderling vergelijken en beoordelen op toepasbaarheid.
Vraag	[2pt] Geef voor ieder van de volgende stellingen aan of ze, waar of niet waar zijn: a) Een rij en een lijst kunnen beide meerdere elementen bevatten. b) Een rij en een lijst met precies dezelfde data zullen ook precies evenveel geheugenruimte innemen. c) Zowel in één rij als in één lijst kan data van verschillende typen worden bewaard.
Beoordelingsmodel	[2pt, 1 punt aftrek voor elke fout antwoord] a) Waar b) Niet waar c) Niet waar
Toelichting	Het gaat om het theoretische verschil tussen rijen (elementen staan achter elkaar in het geheugen) en lijsten (elk element verwijst weer naar het volgende element). Verwarrend kan zijn dat dit onderscheid in programmeeromgevingen niet altijd helder is. In Python kun je bijvoorbeeld elementen van een lijst opvragen met behulp van een index.
Variatiemogelijkheden	Zie ook vraag B2.5 en B2.6 over rijen en lijsten

Naam	B2.8 Record
Leerdoel	15.1 De kandidaat kan meerdere (abstracte) datastructuren herkennen, onderling vergelijken en beoordelen op toepasbaarheid.
Vraag	[1pt] Welke van de onderstaande definities beschrijft een record? a) Een verzameling gegevens die (afgezien van het laatste element) steeds naar een volgend element verwijzen. b) Een verzameling van gegevens die een volgnummer of index hebben.

	c) Een verzameling van gegevens waarbij de elementen knopen worden genoemd. d) Een combinatie van een aantal gekoppelde gegevens die samen een betekenis hebben.
Beoordelingsmodel	[1pt] Antwoord: d
Toelichting	Ter info: a) lijst b) lijst c) rij d) record
Variatiemogelijkheden	Alternatief: andere datastructuur. Je zou ook kunnen vragen naar lijst, boom of rij.

Naam	B2.9 Invoegen element in een lijst
Leerdoel	15.1 De kandidaat kan meerdere (abstracte) datastructuren herkennen, onderling vergelijken en beoordelen op toepasbaarheid.
Vraag	[1pt] De volgende gegevens staan opgeslagen in een *lijst*, in de gegeven volgorde: 'Homeland', 'La Casa de Papel', 'Undercover, 'Bodyguard', 'The Crown' Welk algoritme kun je gebruiken om het gegeven 'The Last Dance' aan het eind van deze lijst toe te voegen? a) Begin bij het eerste element van de lijst. Doorloop de lijst tot je bij het laatste element komt. Voeg dan het nieuwe gegeven toe aan de lijst. b) Vraag het aantal elementen van de lijst op, dit is lengte. Voeg het nieuwe gegeven direct toe aan de lijst op de plek van lengte. c) Vraag het laatste element van de lijst op. Voeg daar het nieuwe gegeven direct toe aan de lijst.
Beoordelingsmodel	[1pt] Antwoord: a.
Toelichting	Door de concrete gegevens wordt het minder abstract voor de leerlingen.
Variatiemogelijkheden	- (moeilijker) Geen voorbeelddata geven, maar de vraag in het algemeen stellen. - Data gebruiken die uit cijfers of enkele woorden bestaat (strings zonder spatie). - Stroomdiagrammen of pseudocode geven in plaats van beschrijvingen. Dit is lastig zonder het heel specifiek te maken met variabelen etcetera.

Naam	B2.10 Invoegen element in een lijst
Leerdoel	15.1 De kandidaat kan meerdere (abstracte) datastructuren herkennen, onderling vergelijken en beoordelen op toepasbaarheid.
Vraag	[2pt] Geef aan of de volgende uitspraak waar of niet waar is. Licht kort toe waarom. *Het invoegen van een element in een lijst kost meestal minder tijd dan het invoegen van een element in een rij.*
Beoordelingsmodel	[1pt] Antwoord: waar. [1pt] Juiste toelichting: want bij een rij moet je alle element daarna één voor één doorschuiven, bij een lijst hoeft dat niet.
Toelichting	
Variatiemogelijkheden	Geef concrete waarden in plaats van een algemene (abstracte) stelling.

Naam	B2.11 Invoegen element in een lijst of rij
Leerdoel	15.1 De kandidaat kan meerdere (abstracte) datastructuren herkennen, onderling vergelijken en beoordelen op toepasbaarheid.
Vraag	[1pt] Geef een voorbeeld waarbij het invoegen van een element evenveel tijd kost als bij een *rij* als een *lijst*.
Beoordelingsmodel	[1pt] Antwoord: Een element achteraan de rij/lijst toevoegen.
Toelichting	
Variatiemogelijkheden	Geef concrete waarden in plaats van een algemene (abstracte) stelling.

Naam	B2.12 Binair zoeken in rijen en lijsten
Leerdoel	15.1 De kandidaat kan meerdere (abstracte) datastructuren herkennen, onderling vergelijken en beoordelen op toepasbaarheid.
Vraag	[1pt] We willen zo efficiënt mogelijk binair zoeken in een verzameling van gesorteerde gegevens. Welke datastructuur kan het beste gebruikt worden om deze

	gegevens te zoeken, een gesorteerde *lijst* of een gesorteerde *rij*?
Beoordelingsmodel	[1pt] Antwoord: gesorteerde rij
Toelichting	Een lijst heeft geen index en daardoor kost het meer acties om steeds het middelste element te benaderen.
Variatiemogelijkheden	Je kunt ook vragen: *We willen snel zoeken in een verzameling gegevens. Welke datastructuur kun je het beste gebruiken, een gesorteerde rij of een gesorteerde lijst? Gebruik je kennis over zoekalgoritmen en licht je antwoord kort toe.* De leerling moet dan zelf tot het inzicht komen dat binair zoeken mogelijk is en dat dat het beste kan in een gesorteerde rij.

Naam	B2.13 Binair en lineair zoeken
Leerdoel	15.1 De kandidaat kan meerdere (abstracte) datastructuren herkennen, onderling vergelijken en beoordelen op toepasbaarheid.
Vraag	[1pt] Bij *binair* zoeken vergelijk je het middelste element van de array met de waarde dat je zoekt. Als ze niet gelijk zijn, wordt verder gezocht in de helft waar de waarde wel in voor zou kunnen komen. Dit halveringsprincipe wordt steeds herhaald. Stel dat je *binair* gaat zoeken naar een element in een *ongesorteerde* rij. Ga ervan uit dat het element aanwezig is in de rij. Wat gebeurt er? a) Je vindt altijd het element, het zal trager gaan dan binair zoeken in een *gesorteerde* rij. b) Je vindt altijd het element, het zal even snel gaan als bij binair zoeken in een *gesorteerde* rij. c) Je vindt het element soms. d) Je vindt het element nooit.
Beoordelingsmodel	[1pt] Antwoord: c
Toelichting	
Variatiemogelijkheden	Alternatief algoritme: je kunt dezelfde vraag stellen over *lineair* zoeken. Het juiste antwoord is dan b: je

	vindt altijd het element en het gaat even snel als lineair zoeken in een gesorteerde rij.

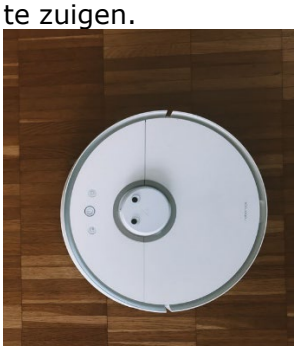
Naam	B2.14 Binair zoeken in een rij
Leerdoel	15.1 De kandidaat kan meerdere (abstracte) datastructuren herkennen, onderling vergelijken en beoordelen op toepasbaarheid.
Vraag	Stel dat je *binair* gaat zoeken naar een element in een rij. Concreet gaat het om het element 'Boon' in de volgende rij: Aardappel, Boon, Courgette, Doperwt, Erwt, Flespompoen, Groenlof. a) [1pt] Geef een voorbeeld van een volgorde van de elementen in de rij waarbij je het element 'Boon' niet zult vinden. b) [1pt] Geef een voorbeeld van een volgorde van de elementen in de rij waarbij je het element 'Boon' wel zult vinden.
Beoordelingsmodel	Mogelijke antwoord: a) [1pt] Aardappel, Courgette, Doperwt, Erwt, Flespompoen, Groenlof, Boon b) [1pt] Boon, Courgette, Erwt, Doperwt, Aardappel, Flespompoen, Groenlof
Toelichting	Ten opzichte van eerdere vragen gaat het om concrete waarden. Dat kan makkelijker zijn voor de leerlingen, want het is minder abstract.
Variatiemogelijkheden	

6. Domein B3 Automaten

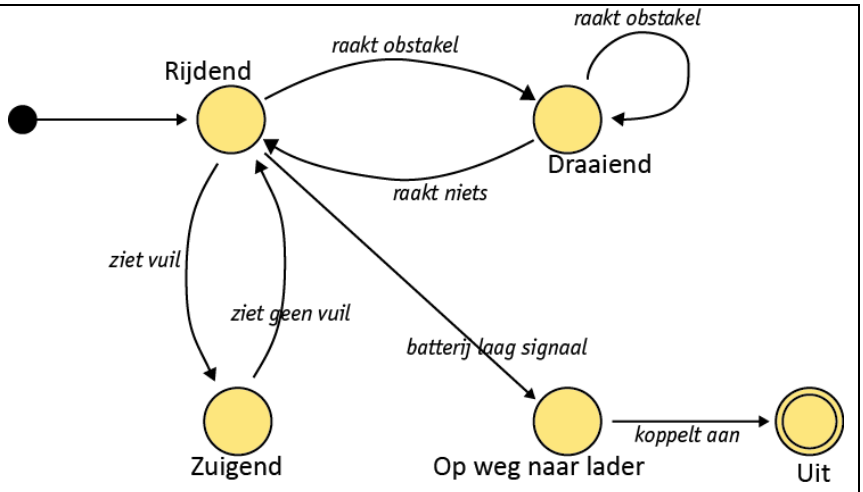
Er zijn 6 voorbeeldtoetsvragen:

- B3.1 Eindige automaat traceren en analyseren
- B3.2 Eindige automaat traceren en analyseren
- B3.3 Eindige automaat traceren en analyseren
- B3.4 Eindige automaat ontwerpen
-

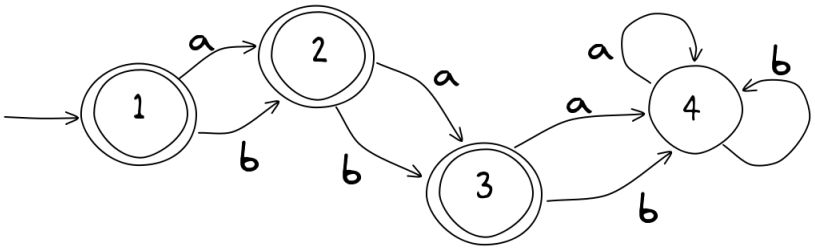
Opmerking: in de voorbeeldspecificaties bij dit domein staat het volgende doel:
(16.1.2) vwo: De kandidaat kan inschatten of een eindige automaat een geschikt instrument is om een bepaald digitaal artefact te karakteriseren.
Voor dit doel hebben we geen vragen opgenomen.

Naam	B3.1 Eindige automaat traceren en analyseren
Leerdoel	16.1 De kandidaat kan eindige automaten gebruiken voor de karakterisering van een digitaal artefact.
Vraag	Een robotstofzuiger rijd door de woonkamer heen om vuil op te zuigen.  [1] Gegeven het onderstaande toestandsdiagram voor een robotstofzuiger. De nummers verwijzen naar de beschrijvingen eronder.

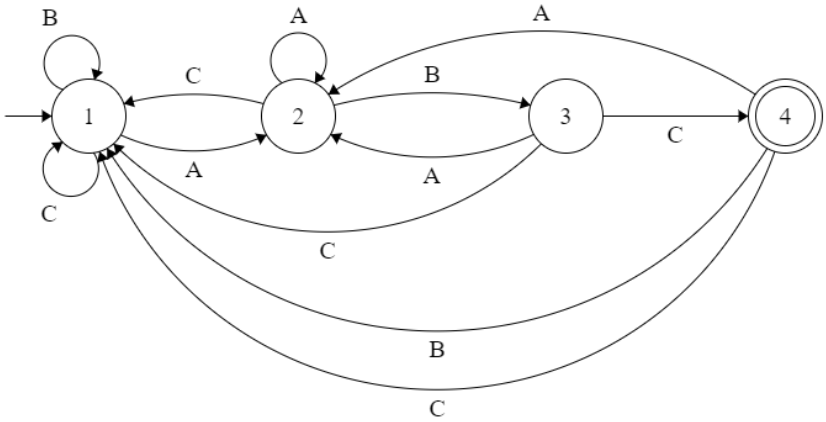
[1] Bronvermelding afbeelding: Jan Antonin Kolar (https://unsplash.com/photos/DeGvnKKETFM)

	 a) Neem de volgende zin over en vul de juiste woorden in op de plaatsen van de stippellijntjes: *De bollen zijn ….. en de pijlen zijn …..* b) Geef het nummer van de toestand waar het systeem in is na de volgende invoer: *Obstakel, obstakel, obstakel, vrij, vuil, schoon, obstakel* c) Beschrijf een reeks van invoer die als resultaat heeft dat het systeem in de eindtoestand komt. d) Beschrijf een situatie waarbij de stofzuiger nooit in de eindtoestand komt.
Beoordelingsmodel	a) [1pt] Toestanden, toestandsovergangen. b) [1pt] Toestand 2. c) [1pt] Mogelijke antwoord: batterij laag, gekoppeld. d) [1pt] Mogelijk antwoord: als de stofzuiger vuil blijft detecteren zal het systeem in toestand 3 blijven. Na verloop van tijd is de batterij leeg en doet het systeem niets meer. Het systeem zal dan nooit in toestand 5 (=eindtoestand) terechtkomen.
Toelichting	Er is bewust voor gekozen een toestandsdiagram te maken voor een fysiek object (automatische stofzuiger). Deze vraag is gebaseerd op: https://www.youtube.com/watch?v=4XEK7OU2gIw https://maken.wikiwijs.nl/125778/Automaten#!page-4702745
Variatiemogelijkheden	- Het toestandsdiagram kan ook zonder context worden gegeven, met alleen nummers en letters als toestandsovergangen, zie vraag B3.2. Daarmee wordt het abstracter en waarschijnlijk moeilijker. Let wel: als

	er geen context is, is er ook geen bijbehorend artefact, namelijk de automatische stofzuiger. - Het is mogelijk om het aantal toestanden uit te breiden waardoor het complexer wordt. - Het is natuurlijk ook mogelijk slechts een deel van de subvragen te gebruiken. - De leerling kan ook gevraagd worden om de diagram uit te breiden. Bijvoorbeeld: 'uit' is geen eindtoestand, met transitie 'batterij vol' gaat de robot weer naar 'Rijdend'.

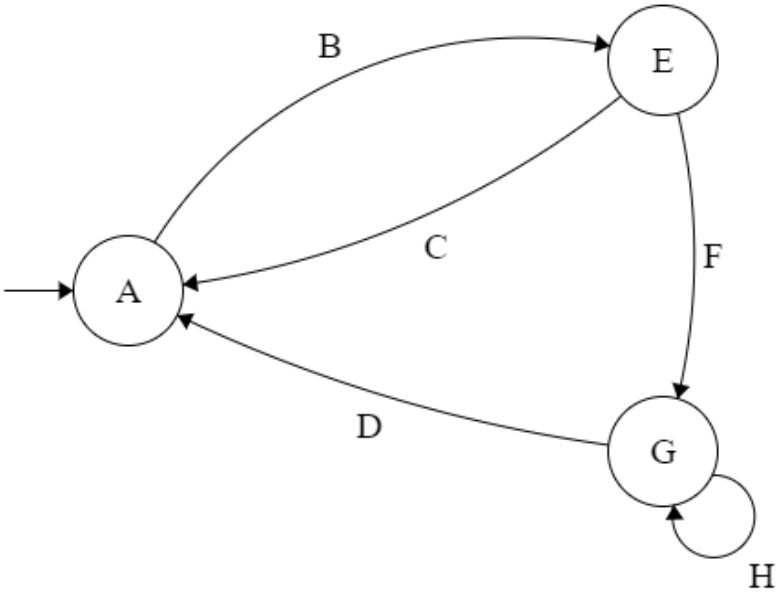
Naam	B3.2 Eindige automaat traceren en analyseren
Leerdoel	16.1 De kandidaat kan eindige automaten gebruiken voor de karakterisering van een digitaal artefact.
Vraag	Gegeven is het onderstaande toestandsdiagram.  a) Geef aan in welke toestand het systeem komt bij de volgende invoer: aabba en bbaab. b) Geef twee invoeren die eindigen in toestand 3. c) Geef aan hoeveel verschillende mogelijkheden van invoer leiden naar een eindtoestand en licht toe hoe je aan dit antwoord komt.
Beoordelingsmodel	a) [1pt] In beide gevallen: toestand 4. b) [1pt, 1pt aftrek voor een incorrect of ontbrekende antwoord.] Mogelijke antwoorden: aa, bb, ab, ba c) [1pt] Antwoord: 7 mogelijkheden. • Toelichting: deze automaat accepteert alle woorden van lengte 0, 1 of 2, bestaande uit de letters a en b. Dus geldige invoer is: <leeg woord>, a, aa, b, bb, ab, ba. De overige woorden zijn niet geldig. Het lege woord wordt vaak vergeten, maar hoort wel bij de oplossing.

Toelichting	In tegenstelling tot vraag B3.1 gaat het hier om een abstract toestandsdiagram. Bij vraag a) en b) gaat het meer om traceren, bij vraag c) moet de leerling het gehele toestandsdiagram kunnen overzien. Deze vraag is gebaseerd op: https://www.csfieldguide.org.nz/en/chapters/formal-languages/finite-state-automata/
Variatiemogelijkheden	- Door wat meer pijlen naar links/terug te laten wijzen (in ieder geval de laatste a die van 4 naar 4 loopt) maak je het diagram iets minder intuïtief. Of in ieder geval moet je echt snappen waar het over gaat om de vraag goed te doen. - Je kunt de leerling op weg helpen (scaffolding) door een voorbeeld van een geldig woord en/of een ongeldig woord te geven. - Je kunt ook een bijbehorend digitaal artefact noemen, zoals een systeem met 2 knoppen, zie vraag B3.2. - Je kunt natuurlijk ook slechts een deel van de subvragen gebruiken.

Naam	B3.3 Eindige automaat traceren en analyseren
Leerdoel	16.1 De kandidaat kan eindige automaten gebruiken voor de karakterisering van een digitaal artefact.
Vraag	Hieronder zie je een toestandsdiagram voor een systeem dat 3 knoppen heeft om het te bedienen: knop A, knop B, knop C.  a) Geef aan in welke toestand het systeem komt als de knoppen in deze volgorde worden ingedrukt. • AABA

	• BBCC • CABC b) Bedenk een volgorde van precies 7 knoppen die ervoor zorgt dat het systeem in de eindtoestand komt. c) Geef in één zin een beschrijving van de combinaties van knoppen die leiden tot de eindtoestand.
Beoordelingsmodel	a) [2 pt, 1 punt aftrek voor elke fout antwoord] • Toestand 2 • Toestand 1 • Toestand 4 b) [1pt] Alle invoer van 7 letters die eindigt op ABC is correct, bijvoorbeeld: AAAAABC. c) [1pt] Alle combinaties die eindigen op ABC leiden tot de eindtoestand.
Toelichting	Door te zeggen dat het om knoppen gaat (A, B, C) wordt het wat concreter dan bij vraag B3.2. Echter, het blijft abstract wat het doel is. Bij vraag a) en b) gaat het meer om traceren, bij vraag c) moet de leerling het gehele toestandsdiagram kunnen overzien.
Variatiemogelijkheden	- Maak het helemaal abstract, dus zonder de knoppen, alleen letters. - Voeg nog meer toestanden en/of invoer toe. (moeilijker) - Door alleen vraag c te stellen, ontbreekt de opbouw (scaffolding) en wordt het moeilijker om te beantwoorden voor de leerling. (moeilijker)

Naam	B3.4 Eindige automaat ontwerpen
Leerdoel	16.1 De kandidaat kan eindige automaten gebruiken voor de karakterisering van een digitaal artefact.
Vraag	Hieronder zie je een toestandsdiagram voor het volgende systeem: een muizenval bestaat uit een kooitje met een deur, die door een gespannen veer dicht kan klappen.



De beschrijvingen ontbreken nog in het toestandsdiagram:

1. Val is open.
2. Val is dicht.
3. Muis raakt trigger (geen actie).
4. Muis raakt trigger (actie: deurtje klapt dicht).
5. Val staat op scherp.
6. Mens maakt val open.
7. Mens spant veer aan.
8. Mens ontspant de veer.

Geef voor iedere letter uit het toestandsdiagram aan welke beschrijving erbij hoort. Gebruik hiervoor de onderstaande tabel. Gebruik alle nummers precies één keer. Het eerste antwoord is gegeven.

Letter uit diagram	Nummer beschrijving
A	1
B	
C	
D	
E	
F	
G	
H	

Beoordelingsmodel	[2pt]: bij volledig correct antwoord.

<table>
<tr><td></td><td>[1pt aftrek bij verwisseling van 2 antwoorden, 1 punt aftrek voor elk overige fout.]
Correct antwoord:
<table><tr><th>Letter uit diagram</th><th>Nummer beschrijving</th></tr><tr><td>A</td><td>1</td></tr><tr><td>B</td><td>7</td></tr><tr><td>C</td><td>8</td></tr><tr><td>D</td><td>6</td></tr><tr><td>E</td><td>5</td></tr><tr><td>F</td><td>4</td></tr><tr><td>G</td><td>2</td></tr><tr><td>H</td><td>3</td></tr></table></td></tr>
<tr><td>Toelichting</td><td></td></tr>
<tr><td>Variatiemogelijkheden</td><td>- Verdeel de beschrijvingen over twee categorieën: toestanden en toestandsovergangen (makkelijker).
- Voeg afleiders toe, dus extra beschrijvingen die niet gebruikt moeten worden (moeilijker), zoals:
o Val gaat open (is een actie, kan niet zonder trigger).
o Val is open -> span veer aan (is een toestand, hoort geen actie bij).
o Span veer aan (kan geen actie van het systeem zijn).
- Het is ook mogelijk het eerste antwoord weg te laten (moeilijker).
- Je kunt de werking van de muizenval uitgebreid beschrijven, waardoor leerlingen alleen de diagramtaal moeten kennen om de zaken goed neer te zetten (makkelijker)</td></tr>
</table>

7. Domein B4 Grammatica's

Er zijn 3 voorbeeldtoetsvragen:

1. B4.1 Grammatica's lezen.
2. B4.2 Grammatica's uitbreiden eenvoudig.
3. B4.3 Grammatica's uitbreiden lastiger.

Naam	B4.1 Grammatica's lezen
Leerdoel	17.2 De kandidaat kan bij een gegeven grammatica en gegeven woorden vaststellen of de woorden aan de grammatica voldoen.
Vraag	De programmeertaal Yuk kent eenvoudige variabelen. Er zijn slechts twee verschillende datatypen: int en string. De waarden van de variabelen kunnen bestaan uit letters, cijfers en enkele symbolen. Een variabeledeclaratie wordt afgesloten met een puntkomma. De volgende grammaticale regels in BNF geven weer aan welke eisen een variabeledeclaratie in de programmeertaal Yuk moet voldoen. <var_declaratie> ::= <type> <naam> = <waarde>; <type> ::= int | string <naam> ::= <tekenreeks> <waarde> ::= <tekenreeks> <tekenreeks> ::= <teken> | <teken><tekenreeks> <teken> ::= <letter> | <cijfer> | <symbool> <letter> ::= a | b | .. | z | A | .. | Z <cijfer> ::= 0 | 1 | 2 | .. | 9 <symbool> ::= _ | " Zijn de volgende variabeledeclaraties toegestaan in de programmeertaal Yuk? Geef per variabeledeclaratie aan: juist/onjuist. 1. int temperatuur = 30; 2. string postcode = "3343PL"; 3. string woonplaats = Rotterdam"; 4. int huisnummer = 2B;

	5. double korting = 33.33; 6. totaalbedrag = 277; 7. int prijsKroket 2; 8. string email = "john.doe@gmail.com"; 9. int aantal_stenen = 20; 10. string afstand"sGravenhage = "80km"; 11. string 1945 = einde_WO2; 12. string adres = "Kerkstraat 2"; 13. int _winkelwagen = 3; 14. int trap_Treden = 13; 15. string datum = 3_april_2020;
Beoordelingsmodel	[3pt als het antwoord helemaal correct is, 1 pt aftrek voor elke fout.] • Juist: 1, 3, 4, 6, 8, 9, 10, 11, 13, 15 • Onjuist: 2, 5, 7, 12, 14
Toelichting	Deze grammatica is een sterk vereenvoudigde weergave van de werkelijkheid. Zo zijn er variabeledeclaraties volgens deze grammatica juist, die volgens geen enkele programmeertaal juist zijn.
Variatiemogelijkheden	De lijst van 15 variabelen kan uitgebreider, of minder uitgebreid.

Naam	B4.2 Grammatica's uitbreiden eenvoudig
Leerdoel	17.3: De kandidaat kan een gegeven grammatica aanpassen, bijvoorbeeld naar aanleiding van een uitbreiding van de taal.
Vraag	Gebruik het volgende grammatica van de programmeertaal Yuk. <var_declaratie> ::= <type> <naam> = <waarde>; <type> ::= int | string <naam> ::= <tekenreeks> <waarde> ::= <tekenreeks> <tekenreeks> ::= <teken> | <teken><tekenreeks> <teken> ::= <letter> | <cijfer> | <symbool> <letter> ::= a | b | .. | z | A | .. | Z

	<cijfer> ::= 0 | 1 | 2 | .. | 9 <symbool> ::= _ | " De ontwikkelaars van Yuk hebben een nieuwe versie van de programmeertaal gemaakt. Er zijn onder andere nieuwe mogelijkheden bij het declareren van variabelen. Verwerk de volgende update in de grammatica: *"Er komt een nieuw datatype beschikbaar: boolean. De waarde van een variabele van het type boolean moet ten minste true of false kunnen zijn."*
Beoordelingsmodel	[1pt] Een extra datatype toevoegen: <type> ::= int | string | boolean.
Toelichting	Het uitbreiden van een grammatica is alleen onderdeel van het vwo-programma volgens de voorbeeldspecificaties. De waarden true en false kun je met de huidige <tekenreeks> maken. Een leerling kan er voor kiezen om expliciet true en false op te nemen. Dat is niet fout, wel overbodig. <waarde> ::= true | false | <tekenreeks>
Variatiemogelijkheden	Het toevoegen van een extra datatype zoals in deze vraag is een eenvoudige uitbreiding. Een andere eenvoudige uitbreiding zou een extra symbool of extra symbolen kunnen zijn.

Naam	B4.3 Grammatica's uitbreiden lastiger
Leerdoel	17.3: De kandidaat kan een gegeven grammatica aanpassen, bijvoorbeeld naar aanleiding van een uitbreiding van de taal.
Vraag	Gebruik het volgende grammatica van de programmeertaal Yuk. <var_declaratie> ::= <type> <naam> = <waarde>; <type> ::= int | string <naam> ::= <tekenreeks> <waarde> ::= <tekenreeks> <tekenreeks> ::= <teken> | <teken><tekenreeks> <teken> ::= <letter> | <cijfer> | <symbool> <letter> ::= a | b | .. | z | A | .. | Z <cijfer> ::= 0 | 1 | 2 | .. | 9 <symbool> ::= _ | " De ontwikkelaars van Yuk zijn fan van de programmeertaal PHP. Ze nemen bij een volgende update van hun taal een aantal syntaxeigenschappen van PHP over. Verwerk de volgende update in de grammatica: a) "De naam van een variabele begint altijd met een #." b) "Na het # moet er eerst een letter of symbool komen. Cijfers zijn niet toegestaan als eerste teken na het #."
Beoordelingsmodel	a) [1pt] Toevoegen van hekje bij de omschrijving van <naam> <naam> ::= #<tekenreeks> b) Een nieuwe tekenreeks maken voor de naam. Die start met een <letter><tekenreeks> of <symbool><tekenreeks>. Hiermee dwing je af dat het eerste teken een letter of symbool is. Daarna zijn er wel cijfers toegestaan.

	[1pt] <naam> ::= #<tekenreeks_naam> [1pt] <tekenreeks_naam> ::= <letter><tekenreeks> | <symbool><tekenreeks>
Toelichting	Het uitbreiden van een grammatica is alleen onderdeel van het vwo-programma volgens de voorbeeldspecificaties. Als het item <tekenreeks> aangepast wordt volgens beoordelingsmodel, dan is het ook verplicht dat de waarde van een variabele met een letter of symbool begint. Dat is niet juist. Er moet dus een nieuwe <tekenreeks> gemaakt worden, specifiek voor de variabele naam.
Variatiemogelijkheden	De grammatica moet op meerdere punten worden aangepast. Een vergelijkbare aanpassing is bijvoorbeeld dat je aangeeft dat vanaf nu de waarde van een variabele altijd tussen aanhalingstekens moet staan.

8. BIJLAGE A: Toetsvragen opbouw algoritmen

Versie 11 juli 2020

8.1 Inleiding

Onderstaande uitwerking laat zien hoe je op basis van één algoritme een reeks van vragen van verschillend niveau kunt genereren. Deze aanpak kun je ook toepassen op andere algoritmen. Daarmee wordt het eenvoudiger om nieuwe toetsvragen te ontwikkelen van min of meer hetzelfde niveau voor domein B1: Algoritmen. Ook is het mogelijk om twee varianten te ontwikkelen, bijvoorbeeld een voor havo en een voor vwo.
Het centrale algoritme in deze uitwerking heeft als doel: *bepaal of een woord een palindroom is*.
Op basis hiervan zijn zeven typen vragen uitgewerkt, van eenvoudig tot moeilijk.

- Type 1: lezen en uitvoeren van het algoritme.
- Type 2: debuggen van het algoritme.
- Type 3: algoritme opstellen door drag&drop.
- Type 4: stellingen over een algoritme.
- Type 5: zelf een algoritme opstellen op basis van gegeven oplossingsrichting.
- Type 6: zelf een algoritme opstellen op basis van probleem.
- Type 7: recursief algoritme opstellen.

Aan het eind van het document staan nog twee voorbeelden van algoritmen die op basis van deze zeven typen kunnen worden gebruikt om toets vragen te genereren:

- Is een woord een anagram?
- Winstmaximalisering voor Bitcoins.

Natuurlijk zijn er nog veel meer algoritmes te bedenken of vinden die als basis voor de zeven type vragen kunnen dienen.
De stroomdiagrammen in dit document zijn gemaakt met:
http://course.cs.ru.nl/greenfoot/flowchart/flowcharttool.html

8.2 Type 1 Lezen en uitvoeren van het algoritme

Bij dit type opdracht moeten leerlingen een algoritme weergegeven in een stroomdiagram lezen en uitvoeren met gegeven input.

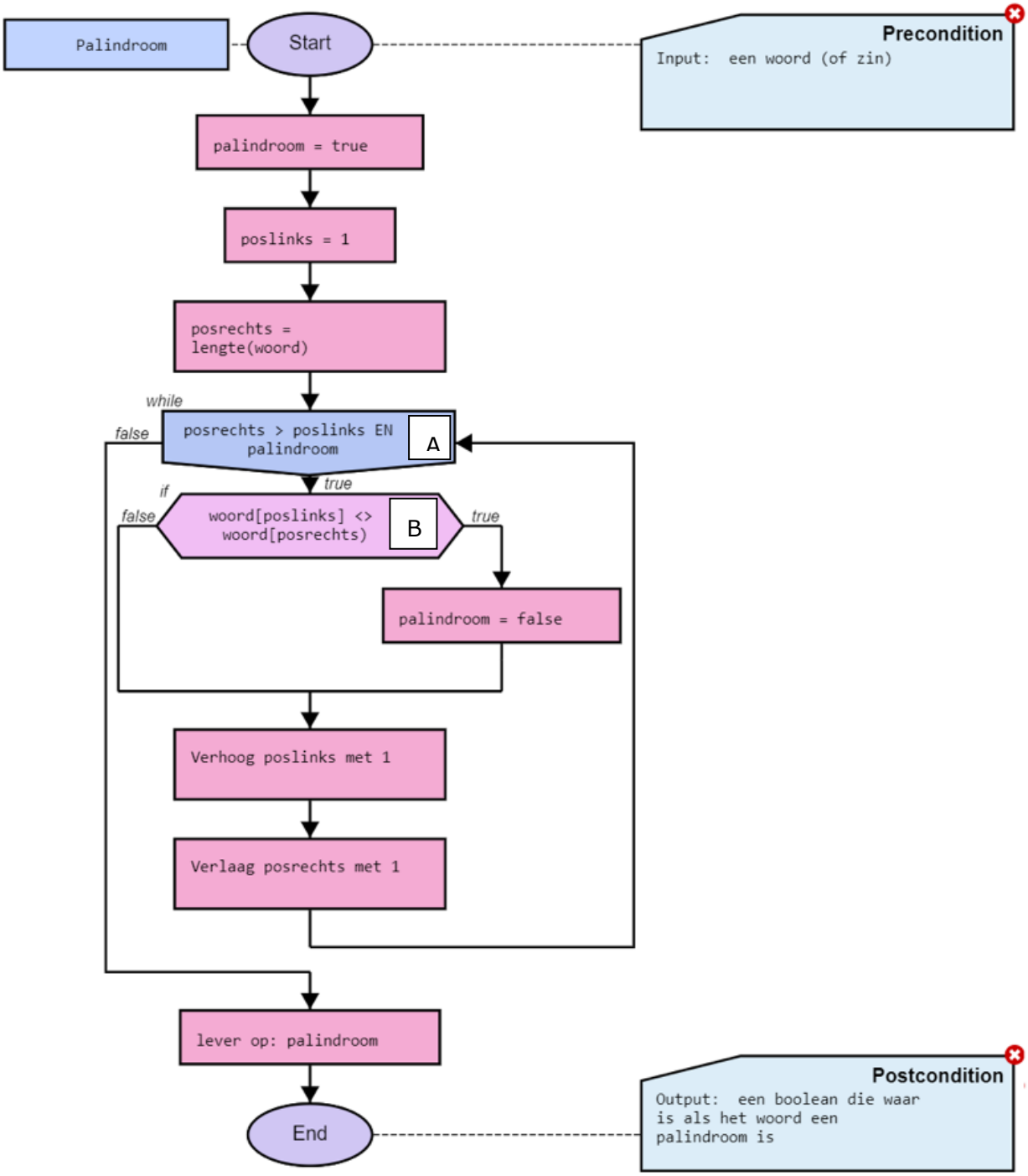
De opdracht

Een palindroom is een woord dat van achter naar voren gelezen precies hetzelfde is als van voor naar achter. RAAR, LEPEL en PARTERRETRAP zijn palindromen, ROER niet.

Hieronder zie je het stroomdiagram van een algoritme dat controleert of een woord een palindroom is. In het stroomdiagram zijn twee blokjes gemarkeerd met de letters A en B. Deze blokjes bevatten voorwaarden.

Voer het algoritme uit voor het woord **LESTABEL.** Vul hieronder in wat de voorwaarden bij A en B worden, met de waarden ingevuld en geef aan of er *true* of *false* uitkomt. Hieronder is de eerste regel alvast ingevuld.

	Waarde posrechts	**Waarde poslinks**	**Waarde palindroom**	**Waarde A**	**Waarde B**
Iteratie 1	9	1	True	true	false
Iteratie 2					
Iteratie 3					
…					


Palindroom
Start
Precondition
Input: een woord (of zin)
palindroom = true
poslinks = 1
posrechts = lengte(woord)
while
false
posrechts > poslinks EN palindroom
A
true
if
false
woord[poslinks] <> woord[posrechts)
B
true
palindroom = false
Verhoog poslinks met 1
Verlaag posrechts met 1
lever op: palindroom
End
Postcondition
Output: een boolean die waar is als het woord een palindroom is

Uitwerking

	Waarde posrechts	**Waarde poslinks**	**Waarde palindroom**	**Waarde A**	**Waarde B**
Iteratie 1	9	1	True	true	false
Iteratie 2	8	2	True	true	false
Iteratie 3	7	3	True	true	true
Iteratie 4	6	4	False	false	nvt
...					

Het algoritme stopt met resultaat *false.*

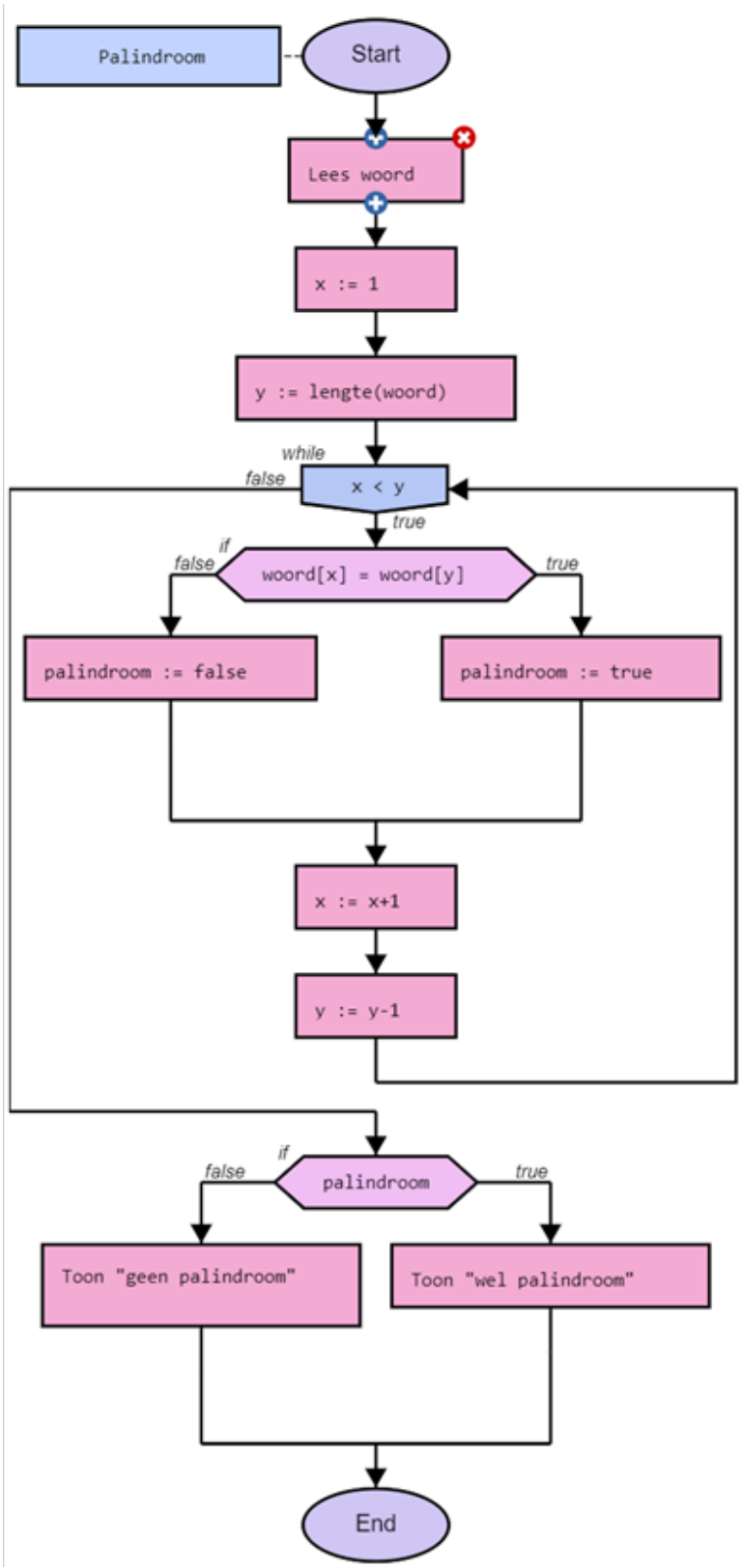
Toelichting
Het is mogelijk om de eerste regel uit de tabel weg te laten om de vraag moeilijker te maken. Ook is het mogelijk om alleen naar de waarde van A en B te vragen, de vraag moeilijker wordt, je biedt dan immers geen *scaffold* (ondersteuning).
Een manier om het makkelijker te maken is door in tekst een toelichting te geven van hoe het algoritme werkt. In de vraag van type 2 hieronder vind je daar een voorbeeld van. Bij type 5 vind je als toelichting twee concrete voorbeelden.

8.3 Type 2 Debuggen van het algoritme

Bij dit type opdracht moeten leerlingen een algoritme waar een fout in zit debuggen en verbeteren.

De opdracht
Een palindroom is een woord dat van achter naar voren gelezen precies hetzelfde is als van voor naar achter. RAAR, LEPEL en PARTERRETRAP zijn palindromen, ROER niet.
Hieronder zie je een stroomdiagram van een programma dat de gebruiker een woord laat invoeren en vervolgens bepaalt of het ingevoerde woord een palindroom is. Dit wordt gedaan door de eerste letter te vergelijken met de laatste letter. Deze moeten gelijk zijn, anders is het geen palindroom. Vervolgens wordt de tweede letter van het woord vergeleken met de een-na-laatste letter. Ook deze moeten gelijk zijn. Daarna wordt de derde letter in het woord vergeleken met de twee-na-laatste letter. Zo gaat het programma verder totdat het in het midden van het woord is uitgekomen. Het programma toont dan of het woord wel of geen palindroom is.

a) Helaas zit er nog een fout in dit stroomdiagram. Geef een voorbeeld van een woord dat geen palindroom is, maar volgens dit stroomdiagram wel een palindroom is. Het hoeft geen bestaand woord te zijn.
b) De fout kan worden hersteld door één blokje te verplaatsen. Geef aan welk blokje en waar het blokje naar verplaatst moet worden.
c) Nadat je het foutje er hebt uitgehaald werkt het programma op zich prima. Wanneer je er nog eens kritisch naar kijkt kom je er achter dat het programma in een aantal gevallen, waarbij het woord geen palindroom is, onnodig veel opdrachten uitvoert, bijvoorbeeld bij KANDELAAR. Hoe zou je het programma met een kleine aanpassing kunnen verbeteren zodat het minder opdrachten uitvoert?


Palindroom
Start
Lees woord
x := 1
y := lengte(woord)
while
false
x < y
true
if
false
woord[x] = woord[y]
true
palindroom := false
palindroom := true
x := x+1
y := y-1
if
false
palindroom
true
Toon "geen palindroom"
Toon "wel palindroom"
End

Uitwerking

a) Het woord KAPPER levert als resultaat *true* op, omdat de letters in het midden gelijk zijn.
b) Het blokje 'palindroom := true' moet uit de herhaling worden verwijderd en worden toegevoegd aan het begin, bijvoorbeeld gelijk na 'Lees woord'.
c) Zodra blijkt dat het geen palindroom is, mag het algoritme stoppen. Dat kan bijvoorbeeld door het blokje 'x < y' te vervangen door 'x < y EN palindroom'.

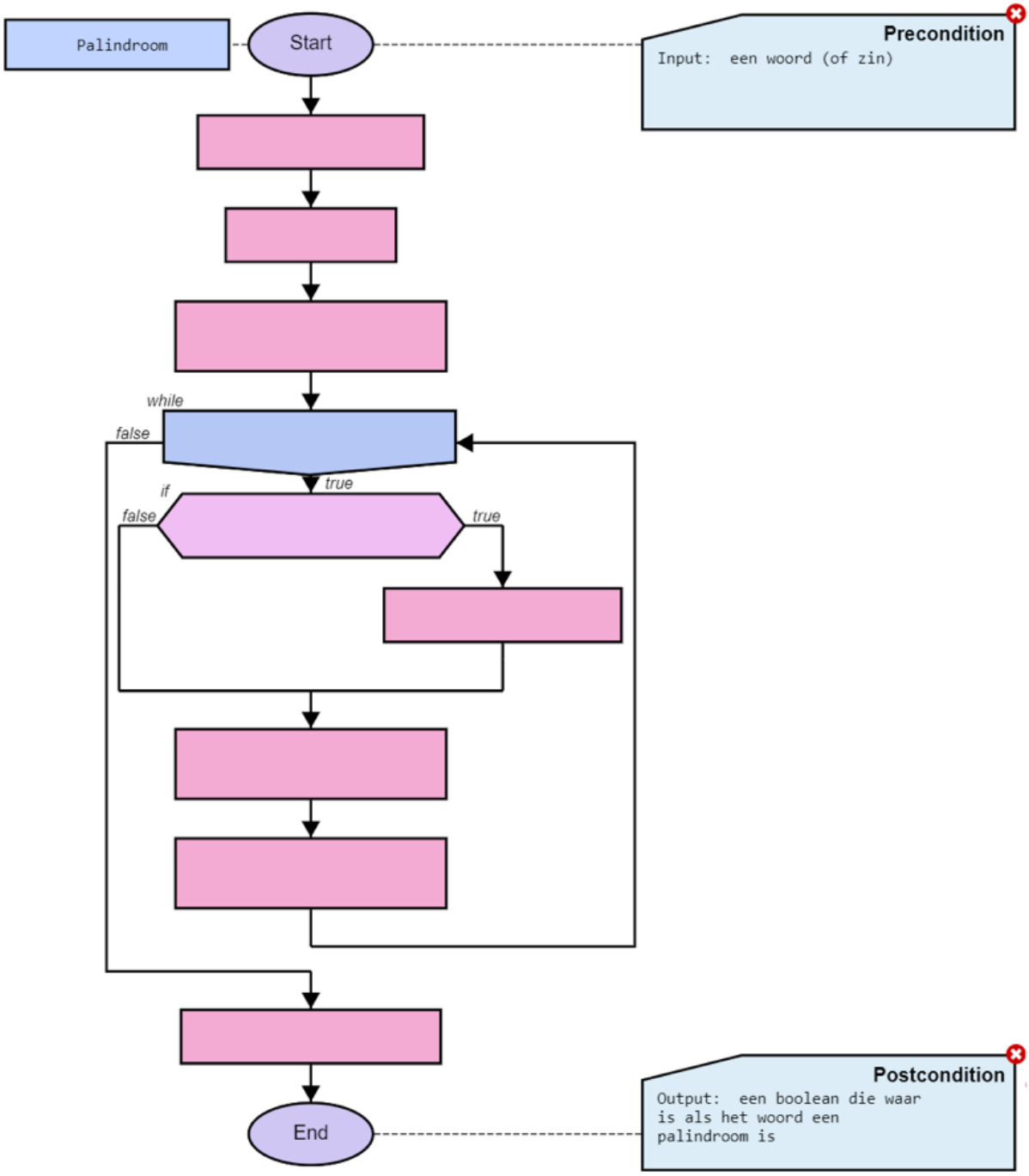
8.4 Type 3 Algoritme opstellen door drag&drop

Bij dit type opdracht moeten leerlingen een algoritme opstellen op basis van gegeven blokjes met instructies.

De opdracht

Een palindroom is een woord dat van achter naar voren gelezen precies hetzelfde is als van voor naar achter. RAAR, LEPEL en PARTERRETRAP zijn palindromen, ROER niet.
Hieronder zie je het stroomdiagram van een algoritme dat controleert of een woord een palindroom is. De blokjes zijn alleen niet ingevuld. Hieronder vind je alle instructies. Neem het stroomdiagram over en zet de instructies op de juiste plek door de bijbehorende letter (A t/m I) in het blokje te schrijven. Alle instructies worden precies één keer gebruikt.

A	lever op: palindroom
B	palindroom = false
C	verhoog poslinks met 1
D	posrechts > poslinks EN palindroom
E	verlaag posrechts met 1
F	palindroom = true
G	poslinks = 1
H	posrechts = lengte(woord)
I	woord[poslinks] <> woord[posrechts]



Uitwerking

Zie de vraag bij type 1.

Toelichting

Je kunt het makkelijker maken door in tekst toe te lichten hoe het algoritme werkt. In de vraag van type 2 hierboven vind je daar een voorbeeld van. Bij type 5 vind je als toelichting twee concrete voorbeelden, ook dat kan.

Daarnaast kun je enkele blokjes alvast invullen, waardoor het makkelijker wordt.
Je kunt het ook moeilijker maken door enkele afleiders toe te voegen bij de instructies. In de opdracht moet dan worden aangegeven dat niet alle instructies gebruikt hoeven worden. Afleiders kunnen bijvoorbeeld zijn:

J	posrechts <= poslinks
K	woord[poslinks] == woord[posrechts]

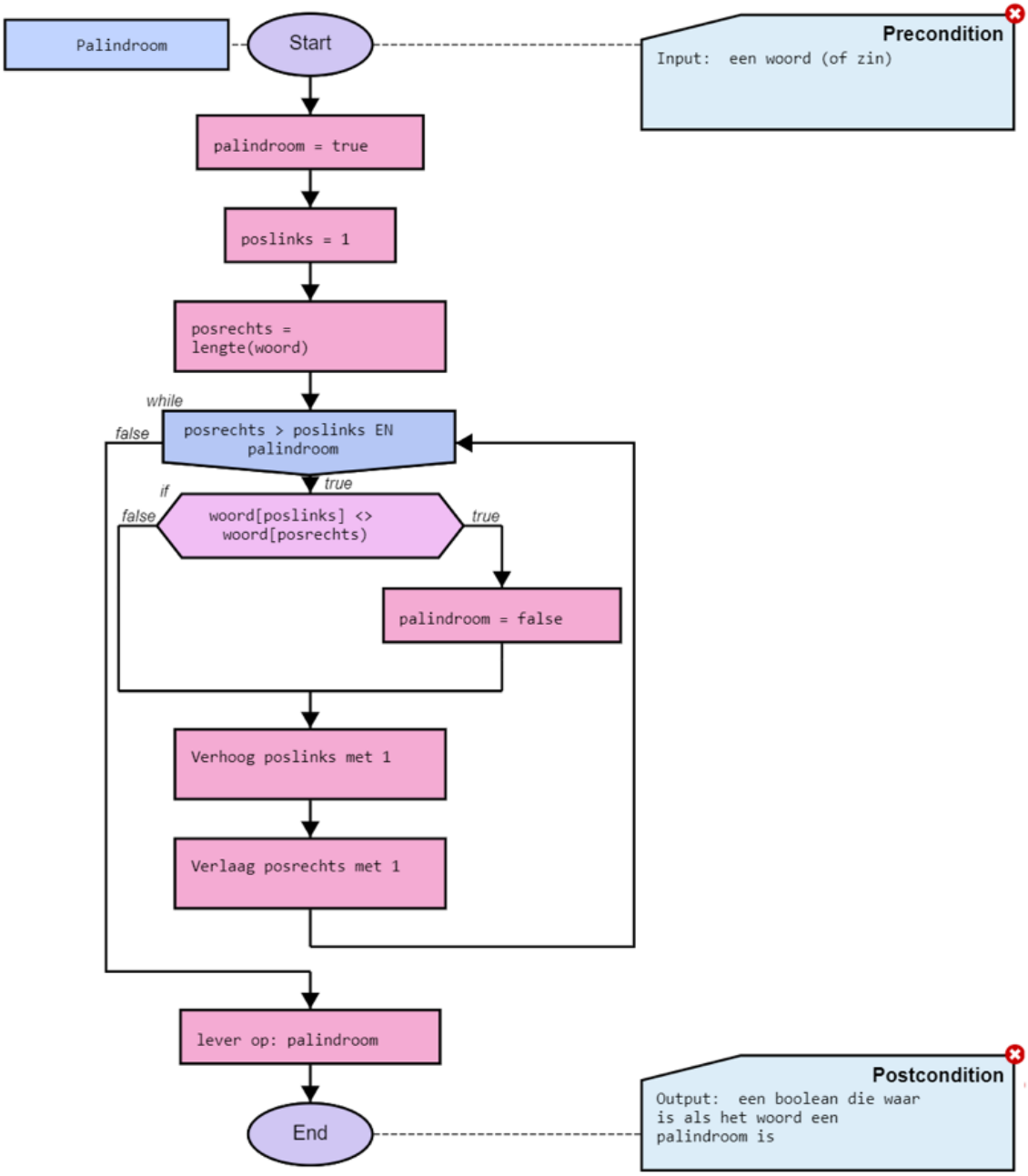
8.5 Type 4 Stellingen over een algoritme

Stellingen zijn een goed didactisch middel om leerlingen te laten nadenken over de werking van een algoritme, maar kunnen ook gebruikt worden in een toets. Bij dit type opdracht moeten leerlingen aangeven of een stelling klopt en eventueel toelichten.

De opdracht

Een palindroom is een woord dat van achter naar voren gelezen precies hetzelfde is als van voor naar achter. RAAR, LEPEL en PARTERRETRAP zijn palindromen, ROER niet. Hieronder zie je het stroomdiagram van een algoritme dat controleert of een woord een palindroom is.

a) Iemand beweert dat bij een woord van 1 letter, het resultaat altijd *true* is. Klopt dat? Zo ja, geef aan waarom. Zo nee, geef een tegenvoorbeeld.
b) In het algoritme zit een herhaling (een while-loop). Het doorlopen van zo'n herhalingslus heet een iteratie. Iemand beweert: het aantal iteraties is gelijk aan het aantal letters van het woord gedeeld door 2, afgerond naar beneden. Klopt dat? Zo ja, geef aan waarom. Zo nee, geef een tegenvoorbeeld.



Uitwerking

a) Ja, dat klopt. Het resultaat van 'posrechts > poslinks EN palindroom' is dan altijd *false* en dus blijft palindroom altijd *true.*

b) Nee, dat klopt niet. Het algoritme stopt immers als het geen palindroom is. Dus bij het woord KAPPER zijn er slechts 2 iteraties en niet 6/2=3.

8.6 Type 5 Zelf een algoritme opstellen op basis van gegeven oplossingsrichting

De leerling moet zelf een algoritme opstellen door een stroomdiagram te maken. De oplossingsrichting wordt daarbij gegeven op basis van enkele voorbeelden.

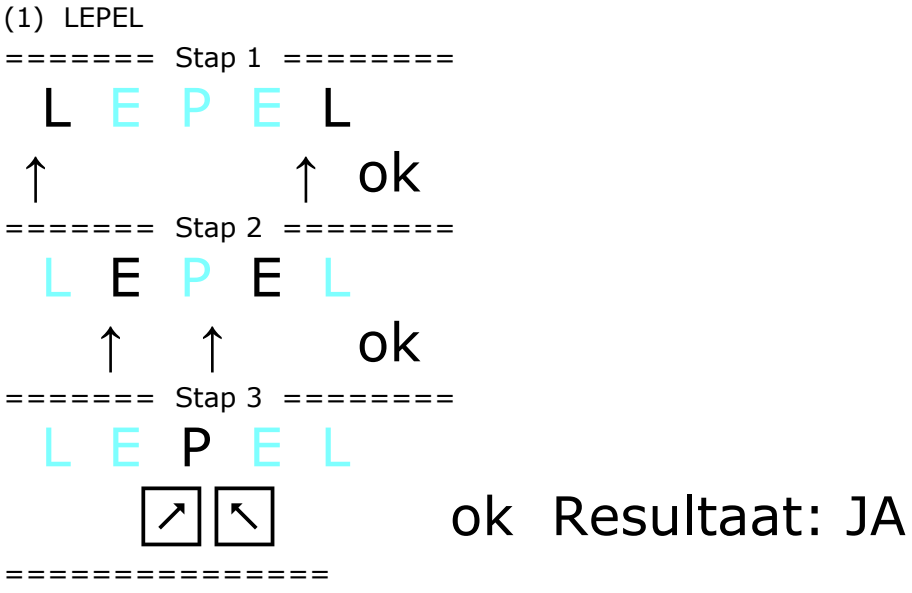
De opdracht

Een palindroom is een woord dat van achter naar voren gelezen precies hetzelfde is als van voor naar achter. RAAR, LEPEL en PARTERRETRAP zijn palindromen, ROER niet. Hieronder zie je het stroomdiagram van een algoritme dat controleert of een woord een palindroom is.

Er is een algoritme dat test of een woord een palindroom is. Je geeft een woord als input, en het algoritme geeft 'JA!' terug als het een palindroom is en 'NEE' als het geen palindroom is. Hieronder zie je het algoritme toegepast op twee woorden: 'LEPEL' en 'LETSEL'.

Stel een stroomdiagram op van de werking van dit algoritme.

(1) LEPEL



(2) LETSEL

Uitwerking
Zie de vraag bij type 1.

Toelichting
Het antwoord kan verschillende abstractieniveaus hebben. Het kan bijvoorbeeld in natuurlijke taal, zonder expliciet variabelen te noemen. In dat geval wordt het vaak wel lastiger te beoordelen of het correct is. Of het kan meer richting een programma gaan, zoals in de uitwerking bij type 2. Dan wordt het beoordelen meestal makkelijker.
Ook is de vraag hoe je moet omgaan met randgevallen: een woord van 0 letters of 1 letter. Door dat expliciet te benoemen kan de vraag wat moeilijker worden gemaakt.
Het is ook mogelijk om naar pseudocode te vragen.
Het is ook mogelijk om de oplossingsrichting te beschrijven zoals bij type 2.

8.7 Type 6 Zelf een algoritme opstellen op basis van probleem

De leerling moet zelf een algoritme opstellen door een stroomdiagram te maken. Alleen het probleem wordt gegeven, de oplossingsrichting moet de leerling zelf bedenken.

De opdracht
Een palindroom is een woord dat van achter naar voren gelezen precies hetzelfde is als van voor naar achter. RAAR, LEPEL en PARTERRETRAP zijn palindromen, ROER niet. Stel een stroomdiagram op dat aangeeft of een woord een algoritme is. Het algoritme krijgt als invoer een woord bestaande uit 1 of meer letters. Het algoritme geeft ‘JA!’ terug als het een palindroom is en ‘NEE’ als het geen palindroom is.

Uitwerking
Zie de vraag bij type 1.

Toelichting
Ook hier kun je variëren zoals bij type 5.
Het zelf bedenken van een oplossingsrichting staat niet beschreven als leerdoel in het eindexamen. Het is een opdracht voor leerlingen die meer uitdaging aankunnen.

8.8 Type 7 Recursief algoritme opstellen

In deze opdracht moeten leerlingen een recursief algoritme formuleren, waarbij ze een beetje op weg geholpen worden doordat de fasen van het opstellen apart worden gevraagd.

De opdracht
Een palindroom is een woord dat van achter naar voren gelezen precies hetzelfde is als van voor naar achter. RAAR, LEPEL en PARTERRETRAP zijn palindromen, ROER niet.
Een *recursief* algoritme roept zichzelf aan. In veel gevallen maak je dan instructies om een stap, een stukje van het probleem, op te lossen en roep je het algoritme weer aan op 'wat er over is'. In deze opdracht pas je dat toe op palindromen.
Stel een recursief algoritme op dat aangeeft of een woord een algoritme is. Het algoritme krijgt als invoer een woord bestaande uit 1 of meer letters. Het algoritme geeft *true* terug als het een palindroom is en *false* als het geen palindroom is. Je mag alle instructies in natuurlijke taal formuleren. Doe dit op basis van de volgende vragen.

a. Wat is de stap waarmee je begint bij de bepaling of een woord een palindroom is?
b. Wat is de recursieve aanroep? Geef aan hoe je probleem hier kleiner geworden is.

Een recursief algoritme kan oneindig doorgaan omdat het zichzelf steeds aanroept. Daarom moet je voorwaarden inbouwen om het algoritme te laten stoppen.

c. Welke voorwaarde kun je bij de stap aangeven om het algoritme te stoppen?
   Geef ook aan wat dan de uitkomst is (*true*/*false*).
d. Welke voorwaarde kun je bij de recursieve aanroep aangeven om het algoritme te stoppen?
   Geef ook aan wat dan de uitkomst is (*true*/*false*).
e. Maak nu het algoritme in pseudocode.

Uitwerking

a. Kijk of de eerste en laatste letter gelijk zijn.
b. Roep het algoritme aan op de rest van het woord: het oorspronkelijke woord minus eerste en laatste letter. Het probleem is nu kleiner omdat het woord twee letters korter is.
c. Het algoritme stopt met uitkomst *false* als de letters bij a verschillend zijn.
d. Het algoritme stopt met de uitkomst *true* als de lengte van het woord 1 of 0 is, bij de recursieve aanroep in b.
e. Zie hieronder.

```
boolean isPalindroom(String woord)
   if (lengte woord groter dan 1)
      if  (eerste letter gelijk aan laatste)
         return isPalindroom (woord-minus-eerste-en-laatste-letter)
      else
         return false
   else
      return true
```

Of, alternatief geformuleerd met staartrecursie (dit is de kortste variant):

```
boolean isPalindroom(String woord)
   if (lengte woord kleiner dan 2)
      return true
   if (eerste letter ongelijk aan laatste)
      return false
   return isPalindroom (woord-minus-eerste-en-laatste-letter)
```

Voor wie bovenstaande te informeel vindt, een moeilijke versie met indices en hulpmethode:

```
boolean isPalindroom (String woord)
      isPalindroom (String woord, 1, lengte(woord))

boolean isPalindroom (String woord, int first, int last)
      if (last <= first)
         return true
      if (woord[first] == woord[last])
         return false
```

```
    return isPalindroom (woord, first+1, last-1)
```

Toelichting
In de voorbeeldspecificaties van domein B staat dat recursieve algoritmen alleen onderdeel zijn van het vwo-programma.
De vragen zijn bedoeld als vorm van ondersteuning (scaffolding) om de leerling op weg te helpen. Het is ook mogelijk om deze vragen weg te laten en daarmee de opdracht moeilijker te maken.
Het is natuurlijk mogelijk om bij de vragen van type 1, 2, 3, 4 en 5 te kiezen voor een recursief algoritme, om daarmee het niveau te verhogen.

8.9 Algoritme: Anagram

Het onderstaande algoritme over het analyseren van een anagram kan als basis dienen voor de 7 type vragen.
"Een anagram is een woord of zin, gevormd uit de letters van een ander woord of een andere zin maar in een andere volgorde." (bron: Wikipedia)

Maak een algoritme dat twee woorden inleest en vervolgens aangeeft of deze twee woorden elkaars anagram zijn. Je mag er bij deze opdracht van uitgaan dat een woord volledig uit hoofdletters bestaat en geen spaties of andere leestekens bevat.

Voorbeelden:
woord1: KLORKESTEIN
woord2: KLEINORKEST
anagram: Ja

woord1: PIETMONDRIAN
woord2: IPAINTMODERN
anagram: Ja

woord1: STOTTER
woord2: TROTSER
anagram: Nee

woord1: ZEKER
woord2: KEIZER
anagram: Nee

8.10 Algoritme: Winstmaximalisatie met bitcoins

Het onderstaande algoritme over het analyseren van een anagram kan als basis dienen voor de 7 type vragen.

De bitcoin is een digitale munteenheid waar sinds 2009 volop in wordt gehandeld. Als je de koers van de bitcoin vooraf zou kunnen voorspellen dat weet je precies wanneer je bitcoins moet kopen:

- Bitcoin is morgen meer waard is dan vandaag: kopen.
- Bitcoin is morgen minder waard is dan vandaag: verkopen.

Je komt in contact met een financieel expert met voorspellende gaven. Hij weet zeker wat de koers van de bitcoin gaat zijn voor de komende dagen en hij is bereid om jou dit te vertellen. Je moet hem wel beloven dat je nooit meer dan één bitcoin tegelijk in je bezit zal hebben. Nadat je hiermee hebt ingestemd vertelt hij het aantal dagen dat hij gaat voorspellen en vervolgens geeft hij per dag de koers door:

Aantal dagen: 10
Koers dag 1: 5
Koers dag 2: 11
Koers dag 3: 4
Koers dag 4: 2
Koers dag 5: 8
Koers dag 6: 10
Koers dag 7: 7
Koers dag 8: 4
Koers dag 9: 3
Koers dag 10: 6

Je schrijft een computerprogramma om te bereken hoeveel winst je maximaal kan behalen. Het programma vergelijkt voor ieder dag de koers met de koers van de volgende dag. Als de koers van de volgende dag hoger is, dan moet een bitcoin worden gekocht. Als de koers van de volgende dag lager is, zal een bitcoin juist verkocht moeten worden. Uiteraard kun je alleen maar een bitcoin verkopen als je er een in je bezit hebt. Je mag alleen een bitcoin kopen als je er nog geen in je bezit hebt.
Het programma begint met het bedrag in het begin op 0 te zetten. Het programma leest daarna het aantal dagen dat er wordt voorspeld en de koers de eerste dag (vandaag). Vervolgens herhaalt het programma steeds de volgende stappen: de koers van morgen wordt ingelezen. Als de koers van vandaag groter is dan de koers van morgen en als er een bitcoin in het bezit is, dan wordt de bitcoin verkocht: de koers van de bitcoin wordt opgeteld bij het bedrag. Als de koers van vandaag lager is dan de koers van morgen en als er nog geen bitcoin in het bezit is, dan wordt er een bitcoin gekocht: het bedrag

wordt verlaagd met de koers van de bitcoin. De koers van vandaag krijgt daarna de waarde van de koers van morgen. Deze stappen worden net zolang herhaald ook de koers van de laatste dag is ingelezen. Uiteindelijk controleert het programma of ook de laatste bitcoin is verkocht. Als dit niet het geval is zal de bitcoin alsnog moeten worden verkocht.
Maak een stroomdiagram voor het programma dat de maximale winst berekent aan de hand van de gegevens die de financieel expert achtereenvolgens geeft.

9. BIJLAGE B: Misconcepten

9.1 Waarom op misconcepten toetsen

Onder *misconcepten* verstaan we begripsproblemen die voortkomen uit kennisoverdracht of kennisaanbod. Vaak gaat het om onderwerpen die complex of moeilijk zijn. Omdat leerlingen nieuw kennis toevoegen aan bestaande kennis, kunnen misvattingen, blijven voortbestaan en hardnekkig zijn om later aan te passen. Er is een relatie tussen de kennis dat een docent heeft over de verkeerde antwoorden die leerlingen gegeven, en hoeveel leerlingen leren (Sadler et. al, 2013).

Uit programmeer onderzoek blijkt dat de meeste fouten voortkomen uit een klein aantal misconcepten (Sirkia en Sorva, 2012). Het toetsen op deze misconcepten:

1) geeft leerling en docent feedback op de aanwezigheid van specifieke misconcepten, zodat daar passend pedagogisch vervolg aan gegeven kan worden;
2) heeft een onderscheidend vermogen tussen wie het wel echt snapt en wie met het concept worstelt.

9.2 Voorbeelden van misconcepten

We geven hieronder enkele voorbeelden van bekende misconcepten. Zie ook http://www.pd4cs.org/ voor meer voorbeelden.

9.3 Misconcepten bij programmeren: toekenning en controlestructuren

Omgekeerde toekenning	eerste = tweede De leerling wijst de waarde van de variabele aan de linkerzijde toe aan de variabele aan de rechterzijde, in plaats van andersom.
Verkeerde tak	IF deler == 0: Ook al evalueert de voorwaarde naar False, de student springt naar de 'THEN'-tak.
Verkeerde False	IF NOT informatie_ok(locatie, afstand): return False Als de voorwaarde False evalueert, gaat de student verder met het retourneren van False uit de functie.
Voorwaarde aan variabele toekennen	IF noemer == 0: Na het evalueren van de uitdrukking, kent de leerling de waarde toe aan noemer.

Voorwaarde in loop control variabele	WHILE i < 7: Na evaluatie van de uitdrukking, kent de leerling de waarde toe aan i.
Grenswaarde	Verkeerde grenswaarde bij itereren van een lijst (laatste element missen), of '<' in plaats van '<='
Optimalisatie	in een stroomdiagram taken weghalen die overbodig lijken, of juist incorrect samenvoegen
Controle stroom	pijlen weglaten in een stroomdiagram, print/return verwarren, denken dat na een return nog iets uitgevoerd kan worden
Condities	Condities: controlestructuur niet goed begrijpen, bijvoorbeeld zowel IF en ELSE worden uitgevoerd. Boolean statements: verschil tussen AND / OR, combinaties met 'NOT'
Variabelen	(tussenresultaat)waarden niet opslaan, denken dat een variabele kan gelijktijdig twee waarden bevatten
Natuurlijke taal	Sommige studenten denken misschien dat de IF-statement altijd wacht tot de Booleaanse conditie waar is, net zoals we het woord "als" gebruiken in natuurlijke conversaties (Pea 1986).

9.4 Misconcepten bij programmeren: Functies

Functie uitvoeren in plaats van definiëren	def ask_euros(): Het def-commando definieert een nieuwe functie: de student wordt verondersteld de functie (object) in het geheugen op te slaan. In plaats daarvan begint de student met het uitvoeren van de functie.
Niet-geëvalueerde parameters	result = calculate(result, result + 1) De student probeert de functieaanroep te starten voordat zij/hij de som heeft geëvalueerd.
Parameter in het verkeerde blok	def calculate(first, second): De student creëert parametervariabelen in het blok van de aanroepende functie, niet in dat van de aangeroepen functie.
Return waarde toekennen aan variabele	return intermediate * intermediate De leerling kent het resultaat van de vermenigvuldiging weer toe aan de variabele intermediate.
Vergeten de return waarde op te slaan	invoer = ask_euros()

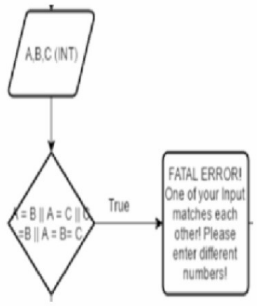
	Na het retourneren van een waarde, slaat de student de retourwaarde niet op, maar in plaats daarvan roept de functie opnieuw aan.

9.5 Bekende misconcepten bij stroomdiagrammen/algoritmen

Moeilijkheden met strategische kennis:

- Wanneer het oplossen van een probleem het tussenvoegen van twee of meer schema's of plannen vereist, voegen beginners vaak alleen de plannen samen en "omzeilen" het tussenvoegen. [6]
- Lusconstructie kiezen in een specifieke context. [7][8][9]
- Een controle gebruiken, zoals controleren op deling door nul. [7][9][10]
- De grootste uitdaging bij het debuggen van beginners is meestal niet het oplossen van de fout, maar het begrijpen van het programma en het lokaliseren van de fout. [11]

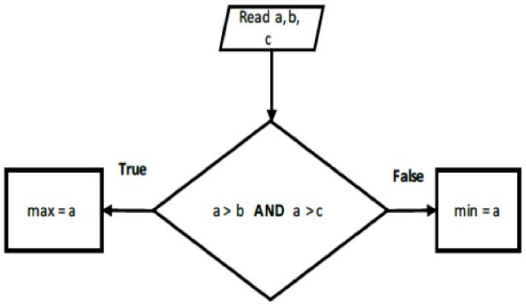
Teken een **stroomdiagram** die, gegeven een lijst van getallen [*a*,*b*,*c*,...], de kleinste waarde print.



Speciale waarden niet afvangen (unexpected case problem)

Veelvoorkomend:
- Lege lijsten
- Waarde gelijk aan 0
- Negatieve waarden
- Gelijke waarden

Radboud Universiteit

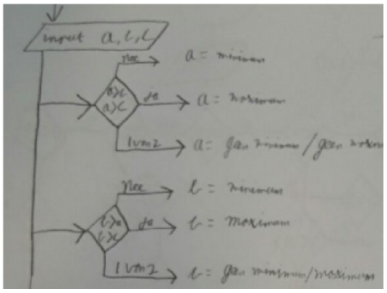


Verkeerd aanpassen van een 'standaard' oplossing (previous experience problem)

Veelvoorkomend:
- Min plan naar max omzetten
- Grensgevallen

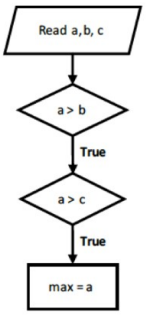
Radboud Universiteit

Bron: Rahimi (2017)



Conditie met 3-mogelijkheden

Radboud Universiteit

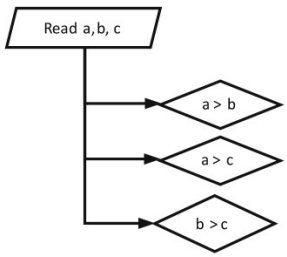


Condities: Ontbreken van paden (bv. False)

Veelvoorkomend:
- Vergissen While/If
- True/False verwisselen

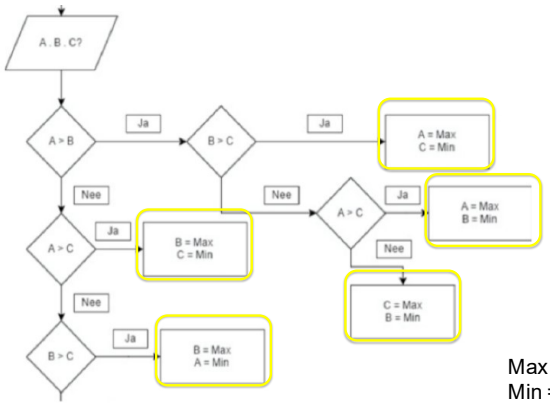
Radboud Universiteit

Bron: Rahimi (2017)



Meerdere operaties worden gelijktijdig uitgevoerd

Radboud Universiteit



Omgekeerde toekenning

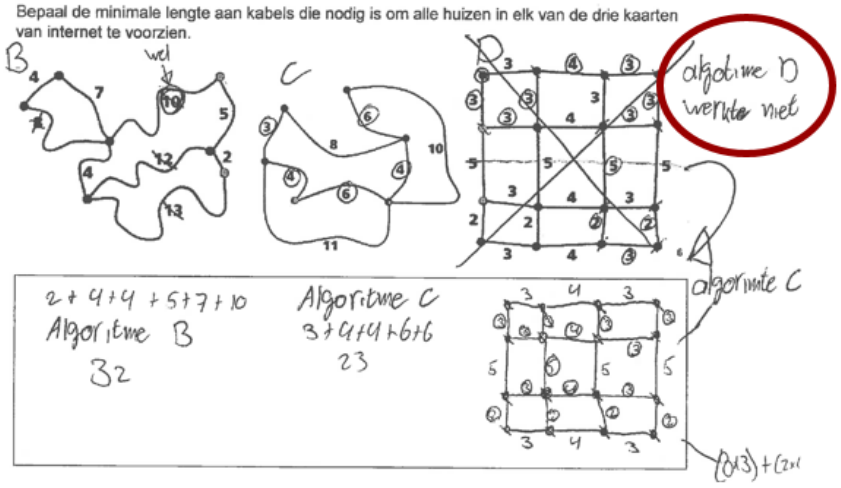
Radboud Universiteit

Bron: Rahimi (2017)

Graaf. Grondslagen: traceren van algoritme

Misconcepten:

- Een algoritme 'kan niet' uitgevoerd worden in bepaalde situatie (Kruskal vs. Prim)
- Verschillende algoritmen hebben verschillende uitvoer



Efficiëntie

Misconcepten:

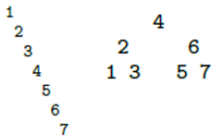
- Looptijd met een stopwatch
- Kortere code is altijd efficiënter dan langere code

9.6 Misconcepten bij datastructuren

Misconcepten bij datastructuren

Misconcepten:

- Een binair boom is uit zichzelf altijd gebalanceerd
- Een combinatie van 'sliert' en 'bosje' kan niet, ook geen zigzag
- Elke binaire boom is een **zoek**boom



Moeilijkheden:

- Wat te doen als je iets wilt toevoegen dat al in de boom staat

9.7 Misconcepten bij toestandsdiagrammen

1. Een schakeling zien als een natuurkundige schakeling. Neem bijvoorbeeld een hotelschakeling in huis. Daar komt geen microcontroller bij aan te pas. Bij physical computing gaan we uit van een microcontroller die het gedrag van het

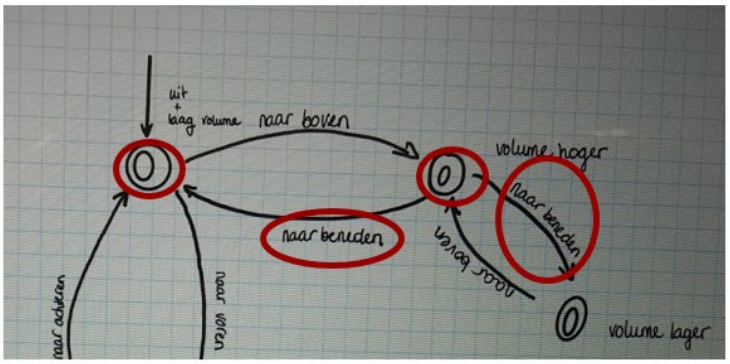
systeem bepaalt op basis van een programma. Dat betekent dat alle sensoren en actuatoren gekoppeld zijn aan de microcontroller en niet direct aan elkaar zoals bij een natuurkunde schakeling.

2. De leerlingen hebben soms geen goed beeld van wat voor sensoren er zijn. Vraag bijvoorbeeld naar sensoren voor een grasmaaierrobot, en wellicht noemen ze een sensor die de hoogte van het gras kan meten. Op zich zou het mogelijk moeten zijn om iets te ontwikkelen dat de hoogte van het gras meet. Dit zal dan waarschijnlijk uit een combinatie van diverse sensoren en software zijn. Het gaat dus om het abstractieniveau.

3. Bij het maken van toestandsdiagrammen zijn we verschillende soorten fouten tegengekomen. Het maken van een toestandsdiagram is overigens niet eenvoudig, het vergt de nodige oefening.

- Het toestandsdiagram wordt gezien als een soort stroomdiagram (flow chart) dat sequentieel werkt. Belangrijk bij het werken met toestandsdiagrammen is na te denken over de toestand waarin een systeem kan verkeren. Toestandsdiagrammen en stroomdiagrammen zijn wezenlijk anders in de manier waarop ze moeten worden geïnterpreteerd.
- Objecten zoals een afstandsbediening worden soms gemodelleerd als een toestand.
- Signalen tussen twee onderdelen, bijvoorbeeld een afstandsbediening en een tv, worden gemodelleerd als een toestandsovergang.
- Soms zijn leerlingen terughoudend in het zelf toevoegen van extra toestanden, terwijl dat wel nodig is om het systeem goed te modelleren.
- Soms worden toestandsovergangen gemaakt zonder specifieke gebeurtenis. De overgang verloopt als het ware automatisch. Een toestandsovergang moet altijd gekoppeld zijn aan een gebeurtenis.
- Niet alle gebeurtenissen worden gekoppeld aan alle toestanden (zie Maak een tabel met alle mogelijke toestandsovergangen).
- De begintoestand wordt gauw vergeten.
- Toestanden en toestandsovergangen worden met elkaar verward of weggelaten.

Toestandsdiagram MP3 speler



Misconcepten:

- Een toestand kan vaker voorkomen
- Vanuit één toestand zijn meerdere zelfde toestandsovergangen mogelijk (niet deterministisch)

10. Bronnen

[1] Sirkia, T., Sorva, J. (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, pp. 19–28. ACM.

[2] Rahimi, Ebrahim, Barendsen, E., Henze, I. (2017). *Identifying students' misconceptions on basic algorithmic concepts through flowchart analysis.* International Conference on Informatics in Schools: Situation, Evolution, and Perspectives. Springer, Cham.

[3] Smetsers-Weeda, R., Smetsers, S. (2017). *Problem solving and algorithmic development with flowcharts.* Proceedings of the 12th Workshop on Primary and Secondary Computing Education.

[4]Docentenhandleiding. Veelgemaakte (denk)fouten. https://maken.wikiwijs.nl/136757/Docentenhandleiding#!page-5379807

[5] Professional Development for CS Principles Teaching. http://www.pd4cs.org/

[6] Ginat, D., Menashe, E., & Taya, A. (2013, February). Novice difficulties with interleaved pattern composition. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 57-67). Springer, Berlin, Heidelberg.

[7] De Raadt, M. (2008). *Teaching programming strategies explicitly to novice programmers* (Doctoral dissertation, University of Southern Queensland).

[8] Fisler, K., Krishnamurthi, S., & Siegmund, J. (2016, February). Modernizing plan-composition studies. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 211-216).

[9] Simon, "Soloway's Rainfall Problem Has Become Harder," *2013 Learning and Teaching in Computing and Engineering*, 2013, pp. 130-135, doi: 10.1109/LaTiCE.2013.44.

[10] Fisler, K. (2014, July). The recurring rainfall problem. In *Proceedings of the tenth annual conference on International computing education research* (pp. 35-42).

[11] McCauley, R., Fitzgerald, S., Lewandowski, G., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: a review of the literature from an educational perspective. *Computer Science Education*, *18*(2), 67-92.

[12] Rahimi, E., Barendsen, E., & Henze, I. (2017, November). Identifying students' misconceptions on basic algorithmic concepts through flowchart analysis. In *International Conference on Informatics in Schools: Situation, Evolution, and Perspectives* (pp. 155-168). Springer, Cham.

Als landelijk kenniscentrum leerplanontwikkeling richt SLO zich op de ontwikkeling van het curriculum in het primair, speciaal en voortgezet onderwijs in Nederland. We werken met het onderwijsveld aan de doelen, kaders en instrumenten waarmee scholen hun opdracht vanuit een eigen visie kunnen vervullen.

We brengen praktijk, beleid, maatschappelijke ontwikkelingen en onderzoek samen en stellen onze expertise beschikbaar aan onderwijs en overheid, bijvoorbeeld in de vorm van leerplannen, tools, voorbeeldlesmaterialen, conferenties en rapporten.

slo

Bezoekadres
Stationsplein 15
3818 LE Amersfoort

Postadres
Postbus 502
3800 AM Amersfoort

T +31 (0)33 484 08 40
E info@slo.nl
W www.slo.nl

company/slo
SLO_nl