Handleiding

Introductie Opdracht Software Engineering

A-mazing Challenge 20

OOSDD.msg

Challenge Software Engineering
Schooljaar: 2021-2022
Auteur: Thomas Boose e.a.

Opleiding: HBO-ICT, Hogeschool Windesheim

September 2022
Versie 4.0

Inhoud

Inleiding

Deze eerste week van het schooljaar gaan jullie ter introductie op het uitstroomprofiel Software engineering een challenge aan. Dit project kent te veel facetten om alles samen te vatten in één alinea. Wat jullie belangrijk vinden en wat jullie meenemen na deze week zal verschillen per student.

Belangrijk is wel dat we lol hebben, dat we zo veel mogelijk leren, dat we elkaar uitdagen en proberen te verslaan. We gaan niet uit van enige voorkennis. Interesse in programmeren en de werking van computers, een gezonde dosis verstand, doorzettingsvermogen en nauwkeurigheid is voldoende om deze challenge met een goed resultaat af te sluiten. We dwingen het gebruik van geen enkele programmeertaal of omgeving af zodat iedereen gelijke kansen heeft op een hoge score. Programmeren en automatisch verwerken van opdrachten kan jullie wel een voorsprong bezorgen, jullie zijn vrij om dit te doen en daarbij een eigen programmeeromgeving in te zetten.

Bij de start kun je regelmatig het gevoel hebben dat je informatie mist. Wees gerust, na de film, de kick-off, het lezen van deze handleiding, de colleges met de bijbehorende terugkoppeling van je groepsgenoten en de eerste ervaringen met de website, zal het aan het einde van de eerste dag voor iedereen duidelijk zijn wat er van jullie wordt verwacht.

Wat heb je nodig?

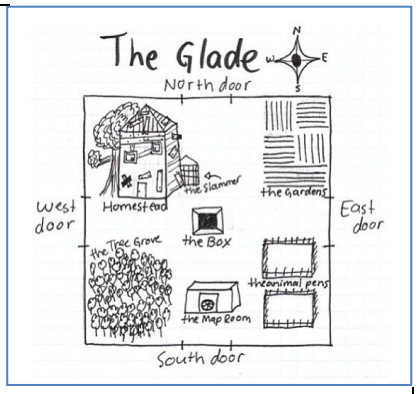
Om succesvol mee te kunnen doen aan deze challenge heb je toegang nodig en zullen jullie als team voorbereidingen moeten treffen. De volgende zaken hebben jullie ontvangen:

- Deze beschrijving en, heb je deze niet, dan weet je niet wat je mist :-)
- 4 tickets voor de film “The Maze Runner” te bekijken bij het Fraterhuis in Zwolle op maandag 6 september.
- 4 tickets voor toegang voor één student per college. De colleges gaan over de volgende onderwerpen:
  - Syntax diagrammen;
  - Structured programming;
  - Orde, Complexiteit en Kosten;
  - Visualiseren.
- Inlogcode voor de “zandbak-website” waar jullie kunnen oefenen met het oplossen van glades.

Synopsis

Deze challenge draait om the glade. The glade is een vierkant veld, omringd door een doolhof van hoge muren. Deze term "the glade" komt uit de film, The Maze Runner, die jullie zullen bekijken.

Het was relatief rustig in the glade totdat er een jongen, genaamd Thomas verscheen. Hij doodt een griever en daardoor is het doolhof van slag. Grievers zijn robots en deze moeten geprogrammeerd worden.



Voorheen gingen 's nachts alle openingen van de Maze dicht waardoor er een relatieve rust heerste in the glade. Maar sinds Thomas een griever doodde gaan de deuren niet meer dicht.

De bewoners van the glade besluiten om zich te beschermen tegen de grievers door obstakels op te werpen in the glade. Aangezien grievers robots zijn, en geprogrammeerd worden, hopen de bewoners op deze manier de belangrijkste voorzieningen, ofwel doelen, te kunnen beschermen tegen deze "relatief" domme apparaten.

Doelen mogen natuurlijk niet afgesloten worden. De runners en gladers moeten er nog wel bij kunnen komen. Alle doelen moeten bereikbaar blijven maar moeilijk bereikbaar zodat de domme grievers op afstand gehouden worden.

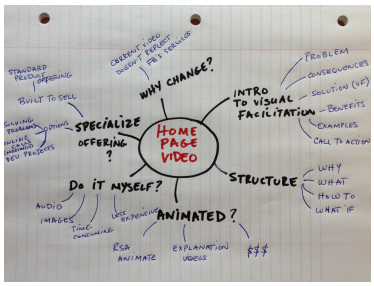
Dit is de context van deze challenge. Jullie spelen in deze challenge in groepen van 4 of 5. Eerst als bewoner van de glade en daarna de programmeur van een griever. Als bewoner is het belangrijk om the glade zo te (ver)bouwen dat het niet makkelijk voor een griever is om de doelen te bereiken. Er moet echter een beloopbaar pad zijn, langs alle doelen in de juiste volgorde, van start tot einde voor de runners en de gladers.

Als programmeur van een griever is het belangrijk om een programma te schrijven dat de griever efficiënt en effectief naar de bron(-nen) of doel(-en) van the glade leidt.

Wat moet je doen?

Om deze challenge aan te kunnen gaan, zullen jullie deze instructie moeten bestuderen. Ook is het belangrijk om de film “The Maze Runner” recentelijk te hebben gezien zodat jullie een gedeelde vocabulaire opbouwen.

Daarnaast bieden wij jullie in fase 1 voorbeeldopgaven aan in een ‘zandbak omgeving’ zodat jullie een beeld kunnen krijgen van de mogelijkheden die je als bewoner van de glade hebt en de mogelijkheden die een programmeur van een griever heeft. Het is belangrijk om deze omgeving te gebruiken om te oefenen met de programmeertaal 20.



Naast inhoudelijke colleges, bieden we begeleiding door jullie SB-ers. Ieder groepslid bezoekt slechts één college. Als iedereen hierbij aantekeningen maakt van belangrijke opmerkingen en tips dan kunnen jullie aansluitend deze kennis met elkaar delen.

Let op dat jullie tijdens het overleggen alle groepsleden de mogelijkheid bieden om zijn of haar ideeën kenbaar te maken. Het maken van een schets, een stukje pseudocode of een mindmap zijn effectieve manieren om ideeën met elkaar uit te wisselen.

Tijdens fase 2 raden we jullie aan om de ideeën eerst in klad uit te werken, te overleggen in de groep en daarna één persoon verantwoordelijk te maken voor het daadwerkelijk uitwerken van the glade of de programma’s op de website. Meer informatie over de website hebben jullie per brief ontvangen.

Leerdoelen

De challenge doen we niet alleen omdat het heel erg leuk is. We willen ook graag dat jullie er iets van leren. Naast algemene leerdoelen die voor HBO-ICT gelden, hebben we leerdoelen specifiek voor deze challenge beschreven.

- Rekenen: Er zal van jullie verwacht worden dat je veel berekeningen maakt. Met name bij het bepalen van de meest efficiënte strategie bij het programmeren van een griever wint de groep die het best zijn sommetjes uitwerkt. Het gaat hierbij niet om zeer complexe berekeningen maar eenvoudigweg; optellen, aftrekken, vermenigvuldigen en delen; maar dan wel zeer precies. Het kan helpen als je voor het rekenwerk tooling gebruikt; Excel, Python, C#, een Calculator maar dat hoeft niet. Het is hierbij zeker belangrijk ervoor te zorgen dat er niet te veel tijd gaat zitten in het programmeerwerk waardoor de uitvoering van de opdracht in gevaar komt.

- Nauwkeurigheid: Jullie zullen verleid worden om snel en doeltreffend te werken. Dat is natuurlijk goed. Het is ook belangrijk als software engineer om daarbij de nauwkeurigheid niet uit het oog te verliezen. Een spatie, puntkomma of = teken maakt op het oog niet veel uit maar in software kan het na het jaren, in een ongelukkige omstandigheid, ineens mensenlevens kosten of erger. Het zijn machines die straks jullie code gaan uitvoeren, zonder vragen te stellen. Deze nauwkeurigheid kun je bereiken door een hoge mate van concentratie, de discipline om je eigen werk regelmatig te controleren en testen en door elkaars werk te doorgronden en valideren voordat je het groepsproduct inlevert, opslaat of uitvoert.

- Uitdagen: Bij het bouwen van the glade zullen jullie als bewoner de programmeurs van grievers moeten uitdagen. Het is niet goed voor je eigen score als niemand jullie glade kan oplossen. Het aantal verschillende goede oplossingen dat andere groepen inleveren voor jullie glade bepaalt namelijk een groot deel van je cijfer. m.a.w:

    - Stel dat niemand jullie glade oplost, dan verdien je 0 van de 2 punten.
    - Zijn alle oplossingen van de groepen die jullie glade oplossen goed, maar verschillend in efficiëntie (verbruik), dan verdien je 2 van de 2 punten.
    - Is je eigen oplossing ondanks dat, het meest efficiënt, dan heb je nog eens 2 punten te pakken.

- Oplossen en Programmeren: Een deel van je cijfer hangt ervan af hoe efficiënt en effectief jullie zijn in het bepalen van een strategie om de uitdagingen van de andere groepen op te lossen. Natuurlijk is het oplossen niet genoeg, je moet de oplossing ook nog programmeren. Maar grievers zijn geen standaard apparaten. De taal waarin ze geprogrammeerd moeten worden is niet bepaald standaard.

- Visualiseren en vormgeven: Het laatste leerdoel betreft het visualiseren en vormgeven van jullie uitdaging / glade. Daarbij is het belangrijk dat je zo realistisch mogelijk, in tijd en vorm de oplossing van je eigen uitdaging / glade weergeeft. Dit moet opgeleverd worden in de vorm van een filmpje van: Lego blokken; in kleipoppetjes; in Minecraft of wat je maar kunt bedenken. Gebruik je fantasie! De oplevering betreft een film op Streams.

Eindeisen

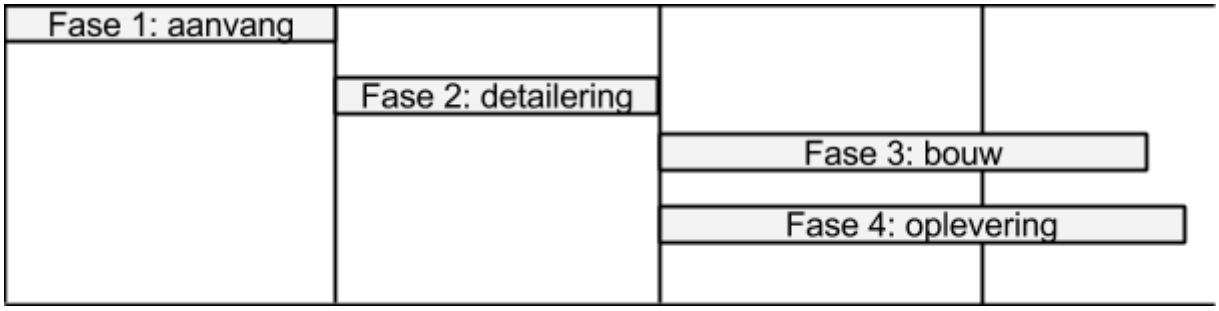
- Deel A (het maken van een uitdagende opdracht)
  de groep maakt een uitdagende opdracht, Glade, die een discriminerende werking heeft.
  de groep voorziet deze opdracht, Glade, van een werkende voorbeeldoplossing.
  de groep maakt geen vergissingen of rekenfout waardoor de eigen oplossing slechter scoort dan veel / het merendeel van de oplossingen van de collega groepen.
  de groep scoort zelf relatief hoog ten opzichte van de topscore van alle groepen.

- Deel B (het oplossen van de uitdagingen van andere groepen)
  Het aantal overwonnen uitdagingen is groot t.o.v. het totaal aantal aangeboden opdrachten.
  Het overgebleven deel van het beschikbare budget is hoog t.o.v. dat van de best presterende groep.

- Deel C (visualisatie)
  de groep visualiseert de eigen opdracht / uitdaging en de eigen oplossing.
  de groep geeft vorm aan de eigen opdracht.

Planning

Met het bekijken van de film “The Maze Runner”, of met het volgen van colleges, begint fase 1.



Fase 1 aanvang

In deze fase worden er colleges gegeven over Syntaxdiagrammen en Structured programming. Daarnaast zijn er, ter voorbereiding op fase 3, colleges over Visualiseren en Complexiteit. Het is verstandig om per student één college bij te wonen. Zorg ervoor dat je aantekeningen maakt en koppel het geleerde terug aan je groepsgenoten.

Aan de hand van deze colleges, deze beschrijving en jullie eigen gezonde verstand kunnen jullie in de zandbak-omgeving een aantal voorbeelduitdagingen proberen op te lossen door je griever te voorzien van een programma.

Fase 2 detaillering

Als fase 2 start ontvangen jullie de inlogcodes van de productieomgeving. De zandbakomgeving zal niet meer beschikbaar zijn.

Vervolgens ontwerpen jullie een eigen versie van the glade. Hiervoor is een editor beschikbaar, maar het is verstandig om eerst een ontwerp in klad te maken. Behalve dat jullie de obstakels, doelen en vallen plaatsen, bepalen jullie ook de kosten voor de verschillende hardware onderdelen, de kosten voor het programmeren en de kosten voor het verbruik van de griever door middel van het invullen van een kostenkaart.

Als proof of concept, bewijs dat jullie uitdaging opgelost kan worden, schrijven jullie een programma dat een runner kan volgen om de uitdaging op te lossen. Dit programma mag maximaal 20 regels code bevatten en dient binnen budget alle doelen te bereiken. Jullie kunnen maximaal 20 keer je eigen code proberen op jullie eigen opdracht. Ook het opslaan van de glade, en de kostenkaart nemen een poging in beslag. Wees dus zuinig met de knop “Opslaan”.

De meeste werkzaamheden, met name het denkwerk, van fase 2 kunnen jullie al gedaan hebben voordat jullie de inloggegevens van de productieomgeving ontvangen. Als jullie dit goed gedaan hebben, dan is het invoeren een kleinigheid.

Zodra jullie tevreden zijn met jullie glade, de kostenkaart en jullie programma, publiceren jullie deze en starten direct de fases 3 en 4. Het publiceren kan vanaf woensdag 8 september om 13:30 uur en moet uiterlijk op donderdag 9 september om 13:00 uur gedaan zijn. Zo niet, dan eindigt de challenge en hebben jullie een onvoldoende.

Fase 3 bouw

Tijdens fase 3 programmeren jullie voor iedere uitdaging een griever. Gedurende de woensdag middag en donderdag ochtend gaan steeds meer groepen hun glade publiceren.

Voor alle uitdagingen kun je maximaal drie opleveringen doen. Wees dus zuinig op je pogingen en probeer bij de derde oplevering geen nieuwe trucjes. Een deel van jullie cijfer wordt bepaald door het budget dat jullie over hebben en het aantal uitdagingen dat jullie weten op te lossen.

Fase 4 oplevering

Fase vier start direct na fase twee. In deze fase maken jullie een visualisatie van jullie eigen glade en het programma dat jullie zelf hebben gemaakt voor de runners.

Hierbij is het belangrijk dat er sprake is van een vloeiende lijn, dat de tijd voor het lopen van één stap en het draaien gelijk is aan één seconde, zoals deze opdracht voorschrijft en dat jullie glade waarheidsgetrouw en aantrekkelijk gevisualiseerd is.

Er is een bonus te verdienen als jullie naast de visualisatie ook de stappen door het programma laten zien, zoals “Step into” in de debugger van Eclipse. Ook kan het wisselen van camerastandpunt (runner view, bird view, target view) op extra waardering rekenen.

Jullie publiceren de film op Streams voor de uiterlijke deadline: vrijdag 10 september 13:00 uur. Aansluitend is er een gelegenheid voor jullie coaches om jullie werk te bekijken en te scoren.

Voorbeelddefinitie glade

Om een beeld te krijgen van de werking van het spel volgt hier een voorbeeld van een glade.

Een glade is een matrix van 20 bij 20 cellen die allen van een bepaald type zijn. Iedere cel is dus een obstakel, een gekleurde tegel waarover een griever of runner kan lopen, een bom, een draaiende plaat, een doel, een startpunt of een bonus veld waar extra geld te verdienen is.

Bij een glade hoort een lijst met kosten voor de aanschaf van hardware, het programmeren van programmacode (software) en het verbruik van resources, tijd en ruimte. Het startkapitaal van een runner of griever is altijd 2020.

Daarnaast dient een glade voorzien te worden van een voorbeeldprogramma waarmee het minimaal binnen budget opgelost wordt. Oplossen betekent hier dat het uitvoeren van het programma, gegeven de door jullie ontworpen glade, ertoe leidt dat alle doelen op de juiste volgorde worden geraakt. Bedenk wel dat jullie veel tijd kunnen nemen voor het maken van jullie glade en het programma voor de runners terwijl een programmeur van een griever slechts beperkt de tijd heeft om hem op te lossen.

Beschrijving van de taal 20

Een voorbeeldprogramma waarmee een runner alle doelen binnen een glade kan bereiken, is onderdeel van de oplevering van een glade. Hier volgt een beknopte beschrijving van de taal 20. Verderop staat een uitgewerkt programma inclusief de bijbehorende rekensom om tot een score te komen. Als bijlage vind je een formele specificatie in de vorm van syntaxdiagrammen.

- De verschillende vergelijkingsoperatoren die je mag gebruiken zijn: `==`, `<`, `>`, `!=` .
- Deze vergelijkingsoperatoren zijn onderdeel van een vergelijking van 2 expressies.
- Er bestaan geen boolean variabelen, literals (true of false) of unaire operatoren (! of not).
- Een vergelijking kun je inzetten bij een "Zolang" of een "Als" regel.

- De rekenkundige operatoren die je kunt gebruiken zijn `+`, `-`, `*`, `%` en `/`.
- Er bestaan enkel integer variabelen, hele getallen.
- De operator `%` geeft de restwaarde van een breuk en `/` het hele deel:

```
5 / 3 = 1
5 % 3 = 2
```

In tegenstelling tot wat je op school hebt geleerd worden alle operaties op volgorde verwerkt.
De expressie: `2 + 3 * 7 % 4` *wordt verwerkt alsof er staat:* `(((2 + 3) * 7) % 4)`

- Een geldig programma in de taal 20 heeft eerst, eventueel, een initialisatieblok, gevolgd door een programmablok.
- Variabelen hebben een naam van 1 kleine letter uit het alfabet [a .. z].
- Variabelen en hardware dienen geïnitialiseerd te worden om ze te kunnen gebruiken.
- Witruimte is een combinatie van 1 of meer spaties en/of tabs, witruimte mag niet worden genegeerd.
- Een programmablok wordt altijd voorafgegaan door een newline "/n", hierop is slechts 1 uitzondering. Deze uitzondering is af te leiden uit de syntaxdiagrammen in de bijlage.

In bijlage 2 vind je een formele definitie van de syntax van de taal 20

Lijst van hardware-, software- en verbruikskosten

Dit is een voorbeeld van een kostenoverzicht bij een glade. De items staan vast, de kosten worden door de bewoners van the glade vastgesteld:

Kostenkaart

Let op: Een gemarkeerde cel betekend dat die waarde is aangepast door de makers, en dus afwijkt van de standaard kostenkaart!

Hardware (aanschaf)		Verbruik		Software (per geschreven statement)	
kompas	100	stapVooruit	1	zolang (lus)	50
zwOog (alles is wit behalve zwart)	50	stapAchteruit	1	als (keuze)	40
kleurOog	100	draaiLinks	5	opdracht	20
variabele (a .. z)	30	draaiRechts	5	toekenning	10
		zwOog	3		
		kleurOog	5		
		kompas	15		
		duw obstakel (schade)	100		
		toewijzing (a = 1) (*het uitvoeren van een toekenning*)	2		
		operatie (+, 0, *, %, /)	2		
		vergelijking (==, <, >, !=)	2		
Start kapitaal	2020				

Voorbeelden:

Compile / design time:

- Het initialiseren van een variabele, of je hem gebruikt of niet, kost bij deze opdracht 30 euro.
- Het schrijven van een zolang lus kost 50 euro, ook als de voorwaarde direct onwaar is.
- Het toekennen van een waarde “a = kleurOog” kost eenmalig 10 euro

Runtime:

- Het berekenen van de som 3 * 3 kost 2 euro
- Het toewijzen van de waarde aan een variabele “a = kleurOog” kost telkens 5 euro voor het aflezen van het kleurOog en 2 euro voor het plaatsen van de waarde in de variabele a.

De tegels van the glade

Een glade is een matrix van 20 bij 20 van tegels van een bepaald type. Dit type wordt aangeduid met 2 tekens. Deze wereld kan worden gedefinieerd d.m.v. 20 regels met ieder 40 tekens:

Bom: [B0, B1 .. B8]
Een bom is een tegel die explodeert. Als de griever of een runner zich op de tegel bevindt tijdens het exploderen dan is de griever of runner kapot. Een `B0` explodeert direct bij betreden. Een [`B1 .. B8`] explodeert 1, 2 .. 8 seconden na het betreden door een griever of runner. Zowel het draaien van een griever of runner als het zetten van een stap kost 1 seconde. Op een B1 kun je dus niet draaien maar je kunt er wel overheen stappen. Na de explosie wordt een `Bx` veld een `O0` veld (puin). De kleur van de bom is zwart(0)

Kleur: [C0..C8]
Een kleur-tegel is een tegel waarop een griever of runner kan staan. Met een "oog" kan de kleur van de vloer afgelezen worden. Een zwart/wit oog ziet alles wit behalve de zwarte tegels.

	Zwart wit oog	Kleuren oog
Wit	1	8
Grijs	1	7
Rood	1	6
Oranje	1	5
Geel	1	4
Groen	1	3
Blauw	1	2
Paars	1	1
Zwart	0	0

Doel: [D1 .. D9]
Een doel moet bereikt worden. Meerdere doelen moeten op volgorde worden bereikt. Je mag over een doel heen lopen en er later terugkeren als het aan de beurt is. Er zijn per glade maximaal 9 doelen: D1 .. D9. Ieder doel komt hooguit 1 keer voor en er is minimaal 1 doel gedefinieerd. Als je op het laatste doel stapt, nadat alle doelen op volgorde zijn bereikt, ben je klaar. Het programma hoeft niet zelf te "stoppen" bij het laatste doel. De kleur van een doel is geel (4)

Bonus: [E1 .. E9]
Bij een bonusveld kan een griever of runner extra geld verdienen. Een "`Ex`" veld levert `2^x` extra euro op. Bijvoorbeeld: een `E2` veld levert 4 euro op en een `E8` veld 256 euro. De kleur van een bonusveld is geel (4). Nadat de bonus is opgepikt blijft er een lege gele tegel achter. Per glade kan iedere bonus [E1 .. E9] maar 1 keer voorkomen.

Obstakel: [O0 .. O3]
Tegen een obstakel kun je aanlopen. De griever of runner blijft dan staan maar duwen kost wel geld. Er zijn verschillende soorten obstakels: `O0=Puin,O1=Heg,O2=Steen,O3=Hout`

Draai: [R0 .. R3]
Als een griever of runner op een draaivlak staat dan draait hij. Bij R1 draait de griever of runner 90 graden met de klok mee, R2 180 graden met de klok mee, R3 270 graden per seconde met de klok mee. Als een griever of runner op een R0 veld komt draait hij een willekeurig aantal graden (0,90,180 of 270). Je heb een kompas nodig om na een willekeurige draai te weten in welke richting je staat.

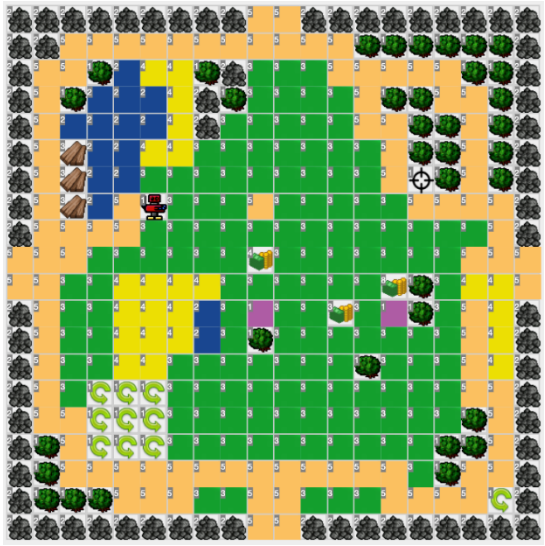
	kompas
Noord	0
Oost	1
Zuid	2
West	3

Als je op een R1 veld stapt en er direct weer afstapt ben je 90 graden gedraaid maar als je zelf een draaibeweging maakt op een R1 veld dan draait de schijf een 2e keer voordat je er weer af bent. Een draaibeweging van een griever of runner kost immers ook 1 seconde net als een stap. De kleur van een draaischijf is blauw (2).

Start: [S0 .. S3]
Er is in iedere glade 1 veld waarop de griever of runner start. De griever of runner kan in vier richtingen worden opgesteld: Noord = 0, Oost = 1, Zuid = 2 en West = 3. De kleur van het startveld is zwart (0).

Voorbeeld van het ontwerp van een glade

20 regels van ieder 20 cellen met ieder een letter en een cijfer zou er als volgt uit kunnen zien:



O2	O2	O2	O2	O2	O2	O2	O2	O2	C5	C5	O2	O2	O2	O2	O2	O2	O2	O2	O2
O2	O2	C5	C5	C5	C5	C5	C5	C5	C5	C5	C5	C5	O1	O1	O1	O1	O1	O1	O2
O2	C5	C5	O1	C2	C4	C4	O1	O2	C3	C3	C3	C5	C5	C5	C5	C5	O1	O1	O2
O2	C5	O1	C2	C2	C2	C4	O2	O1	C3	C3	C3	C3	C5	O1	O1	C5	C5	O1	O2
O2	C5	C2	C2	C2	C2	C4	O2	C3	C3	C3	C3	C3	C5	C5	O1	O1	C5	O1	O2
O2	C5	O3	C2	C2	C4	C4	C3	C3	C3	C3	C3	C3	C3	C5	O1	O1	C5	O1	O2
O2	C5	O3	C2	C2	C3	C3	C3	C3	C3	C3	C3	C3	C3	C5	D1	O1	C5	O1	O2
O2	C5	O3	C2	C5	S1	C3	C3	C3	C5	C3	C3	C3	C3	C3	C5	C5	C5	C5	O2
O2	C5	C5	C5	C5	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C5	O2
C5	C5	C5	C3	C3	C3	C3	C3	C3	E4	C3	C3	C3	C3	C3	C3	C3	C5	C5	C5
C5	C5	C3	C3	C4	C4	C4	C4	C3	C3	C3	C3	C3	C3	E8	O1	C3	C4	C4	C5
O2	C5	C3	C3	C4	C4	C4	C2	C3	C1	C3	C3	E3	C3	C1	O1	C3	C5	C4	O2
O2	C5	C3	C3	C4	C4	C4	C2	C3	O1	C3	C3	C3	C3	C3	C3	C3	C5	C4	O2
O2	C5	C3	C3	C4	C4	C3	C3	C3	C3	C3	C3	C3	O1	C3	C3	C3	C5	C4	O2
O2	C5	C3	R1	R1	R1	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C5	C5	O2
O2	C5	C5	R1	R1	R1	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	O1	C5	O2
O2	O1	C5	R1	R1	R1	C3	C3	C3	C3	C3	C3	C3	C3	C3	C3	O1	O1	C5	O2
O2	O1	C5	C5	C5	C5	C5	C5	C5	C5	C5	C5	C5	C5	C5	C5	C3	O1	C5	C5	O2
O2	O1	O1	O1	C5	C5	C5	C3	C3	C5	C5	C3	C3	C3	C5	C5	C5	C5	R1	O2
O2	O2	O2	O2	O2	O2	O2	O2	O2	C5	C5	O2	O2	O2	O2	O2	O2	O2	O2	O2

Je ziet hier een uitwerking uitgeschreven en de manier waarop de website de glade zal weergeven.

Voorbeeldprogramma

Bij een ontwerp van een glade moet ook een programma / oplossing worden geleverd die werkt. Deze oplossing moet een runner alle doelen laten bereiken zonder dat het budget op raakt. Het voorbeeldprogramma mag niet meer dan 20 regels code omvatten. Het voorbeeldprogramma is niet zichtbaar voor de tegenstanders maar wel een voorwaarde voor publicatie van jullie opdracht.

Hieronder rekenen we twee voorbeeldoplossingen na:

Programma	**Softw**	**Hardw**	**Verbruik**	**Bonus**
gebruik kleurOog		-100		
zolang 1 == 1 {	-50		5 * -2	
stapVooruit	-20		5 * -1	
stapVooruit	-20		4 * -1	
stapVooruit	-20		4 * -1	+24
stapVooruit	-20		4 * -1	
als kleurOog == 5 {	-40		4 * -(2 + 5)	
draaiRechts	-20		2 * -5	
}				
als kleurOog == 1 {	-40		4 * -(2 + 5)	
draaiLinks	-20		2 * -5	
stapVooruit	-20		2 * -1	+256
}				
}				
	====	====	======	====
	-270	-100	-105	+280
	_____/			
			\/	
			= **- 195**	
	Startkapitaal		**+2020**	
			========	
	Score		**+1825**	

Feedback

```
start kapitaal: 2020
kosten zolang: 1 * 50
kosten als: 2 * 40
kosten opdracht: 7 * 20
kosten toekenning: 0 * 10
kosten voor declaratie 'kleurOog': 100
kosten voor het vergelijken: 2
kosten voor stap vooruit: 1 naar: [7,6]
kosten voor stap vooruit: 1 naar: [7,7]
kosten voor stap vooruit: 1 naar: [7,8]
kosten voor stap vooruit: 1 naar: [7,9]
kosten voor gebruik kleur-oog: 5
kosten voor het vergelijken: 2
kosten voor het draaien naar rechts: 5
kosten voor gebruik kleur-oog: 5
kosten voor het vergelijken: 2
kosten voor het vergelijken: 2
kosten voor stap vooruit: 1 naar: [8,9]
kosten voor stap vooruit: 1 naar: [9,9]
bonus van 16 gepakt.
kosten voor stap vooruit: 1 naar: [10,9]
kosten voor stap vooruit: 1 naar: [11,9]
kosten voor gebruik kleur-oog: 5
kosten voor het vergelijken: 2
kosten voor gebruik kleur-oog: 5
kosten voor het vergelijken: 2
kosten voor het draaien naar links: 5
kosten voor stap vooruit: 1 naar: [11,10]
kosten voor het vergelijken: 2
kosten voor stap vooruit: 1 naar: [11,11]
kosten voor stap vooruit: 1 naar: [11,12]
bonus van 8 gepakt.
kosten voor stap vooruit: 1 naar: [11,13]
kosten voor stap vooruit: 1 naar: [11,14]
kosten voor gebruik kleur-oog: 5
kosten voor het vergelijken: 2
kosten voor gebruik kleur-oog: 5
kosten voor het vergelijken: 2
kosten voor het draaien naar links: 5
kosten voor stap vooruit: 1 naar: [10,14]
bonus van 256 gepakt.
kosten voor het vergelijken: 2
kosten voor stap vooruit: 1 naar: [9,14]
kosten voor stap vooruit: 1 naar: [8,14]
kosten voor stap vooruit: 1 naar: [7,14]
kosten voor stap vooruit: 1 naar: [6,14]
kosten voor gebruik kleur-oog: 5
kosten voor het vergelijken: 2
kosten voor het draaien naar rechts: 5
kosten voor gebruik kleur-oog: 5
kosten voor het vergelijken: 2
kosten voor het vergelijken: 2
kosten voor stap vooruit: 1 naar: [6,15]
doel [6, 15] bereikt.
eind kapitaal :1825
Doel bereikt binnen budget
```

De kolom “Feedback” is de feedback die jullie te zien krijgen als je het betreffende programma zou opslaan.

Een veel eenvoudiger programma is te maken, zonder oog, door gewoon de stappen te zetten en de draaibewegingen te maken die nodig zijn om het pad af te lopen. Is dit een 'goedkopere' oplossing?:

Programma	Softw	Hardw	Verbruik	Bonus
stapVooruit	-20		-1	
stapVooruit	-20		-1	
stapVooruit	-20		-1	
stapVooruit	-20		-1	
draaiRechts	-20		-5	
stapVooruit	-20		-1	
stapVooruit	-20		-1	+16
stapVooruit	-20		-1	
stapVooruit	-20		-1	
draaiLinks	-20		-5	
stapVooruit	-20		-1	
stapVooruit	-20		-1	
stapVooruit	-20		-1	+8
stapVooruit	-20		-1	
stapVooruit	-20		-1	
draaiLinks	-20		-5	
stapVooruit	-20		-1	+256
stapVooruit	-20		-1	
stapVooruit	-20		-1	
stapVooruit	-20		-1	
stapVooruit	-20		-1	
draaiRechts	-20		-5	
stapVooruit	-20		-1	
	====	====	======	====
	-460		-39	+280
	_____		\/ _____	/
			= **- 219**	
	Startkapitaal		**+2020**	
			========	
	Score		**+1801**	

Feedback

```
start kapitaal: 2020
kosten zolang: 0 * 50
kosten als: 0 * 40
kosten opdracht: 23 * 20
kosten toekenning: 0 * 10
kosten voor stap vooruit: 1 naar: [7,6]
kosten voor stap vooruit: 1 naar: [7,7]
kosten voor stap vooruit: 1 naar: [7,8]
kosten voor stap vooruit: 1 naar: [7,9]
kosten voor het draaien naar rechts: 5
kosten voor stap vooruit: 1 naar: [8,9]
kosten voor stap vooruit: 1 naar: [9,9]
bonus van 16 gepakt.
kosten voor stap vooruit: 1 naar: [10,9]
kosten voor stap vooruit: 1 naar: [11,9]
kosten voor het draaien naar links: 5
kosten voor stap vooruit: 1 naar: [11,10]
kosten voor stap vooruit: 1 naar: [11,11]
kosten voor stap vooruit: 1 naar: [11,12]
bonus van 8 gepakt.
kosten voor stap vooruit: 1 naar: [11,13]
kosten voor stap vooruit: 1 naar: [11,14]
kosten voor het draaien naar links: 5
kosten voor stap vooruit: 1 naar: [10,14]
bonus van 256 gepakt.
kosten voor stap vooruit: 1 naar: [9,14]
kosten voor stap vooruit: 1 naar: [8,14]
kosten voor stap vooruit: 1 naar: [7,14]
kosten voor stap vooruit: 1 naar: [6,14]
kosten voor het draaien naar rechts: 5
kosten voor stap vooruit: 1 naar: [6,15]
doel [6, 15] bereikt.
eind kapitaal :1801
Doel bereikt binnen budget
```

Behalve dat deze oplossing uiteindelijk minder euro over laat dan de vorige is er nog een belangrijk probleem. Als ontwerper van een glade mag je maar 20 regels code gebruiken. Dit programma heeft 23 regels en mag dus alleen door de tegenstanders gebruikt worden.

Kunnen jullie een programma schrijven waarmee je meer dan 1825 euro over houdt?

Bijlage 1: Beoordelingsmodel A-mazing Challenge 20

Dit beoordelingsmodel bestaat uit 3 delen.

Deel A en B, beide ter waarde van 4 punten, zullen automatisch gescoord worden door het “a-Mazing” systeem dat jullie gedurende deze week gebruiken om je Glade te publiceren en de oplossingen in te leveren. De stand is continue te volgen op deze site. Meer informatie over de website vind je in de inlogbrief. Publicaties en oplossingen van andere groepen kunnen invloed hebben op deze score. Streef dus naar een ruime marge om teleurstellingen aan het einde van de challengeweek te voorkomen.

Deel C, ter waarde van 2 punten, zal op de laatste dag van de challenge door jullie SB-er beoordeeld worden.

Een voldoende, 5.5 of hoger voor deze beoordeling telt mee als een deeltoets voor de module BAI.

Onderdeel	Score toelichting	Score
Deel A: Eigen glade		Index
Maken werkende glade, kostenkaart en voorbeeld oplossing	Go / No go	0
Discriminerende werking	Aantal verschillende scores / totaal aantal scores	1
Toernooi prestatie	Ranking van de eigen oplossing / totaal aantal scores	2
Relatieve prestatie	Eigen score / top score eigen glade	3
Eindscore Deel A	als score[0] == “no go” dan 0 anders (score[1] * 2) + score[2] + score[3]	

Onderdeel	Score toelichting	Score
Deel B: glades oplossen		Index
Oplossen	Aantal werkende oplossingen / totaal aantal glades	1
Overgebleven budget	(Som (overgebleven budget) + som (achievements)) / topscore	2
Eindscore Deel B	(score[1] * 2) + (score[2] * 2)	

Onderdeel	Score toelichting	Score
Deel C: Visualisatie		Index
Vormgeving eigen glade	Examinator Grade [0,1]	1
Visualisatie eigen glade	Examinator Grade [0,1]	2
Eindscore Deel C	score[1] + score[2]	

Onderdeel	Score toelichting	Score
		Index
Deel A Totaal eigen glade	Verplicht > 1 geen compensatie mogelijk	0
Deel B Totaal wedstrijd	Verplicht > 1 geen compensatie mogelijk	1
Deel C Totaal visualisatie	Compensatie is mogelijk	2
Eindscore	som(score[0], score[1], score[2])	

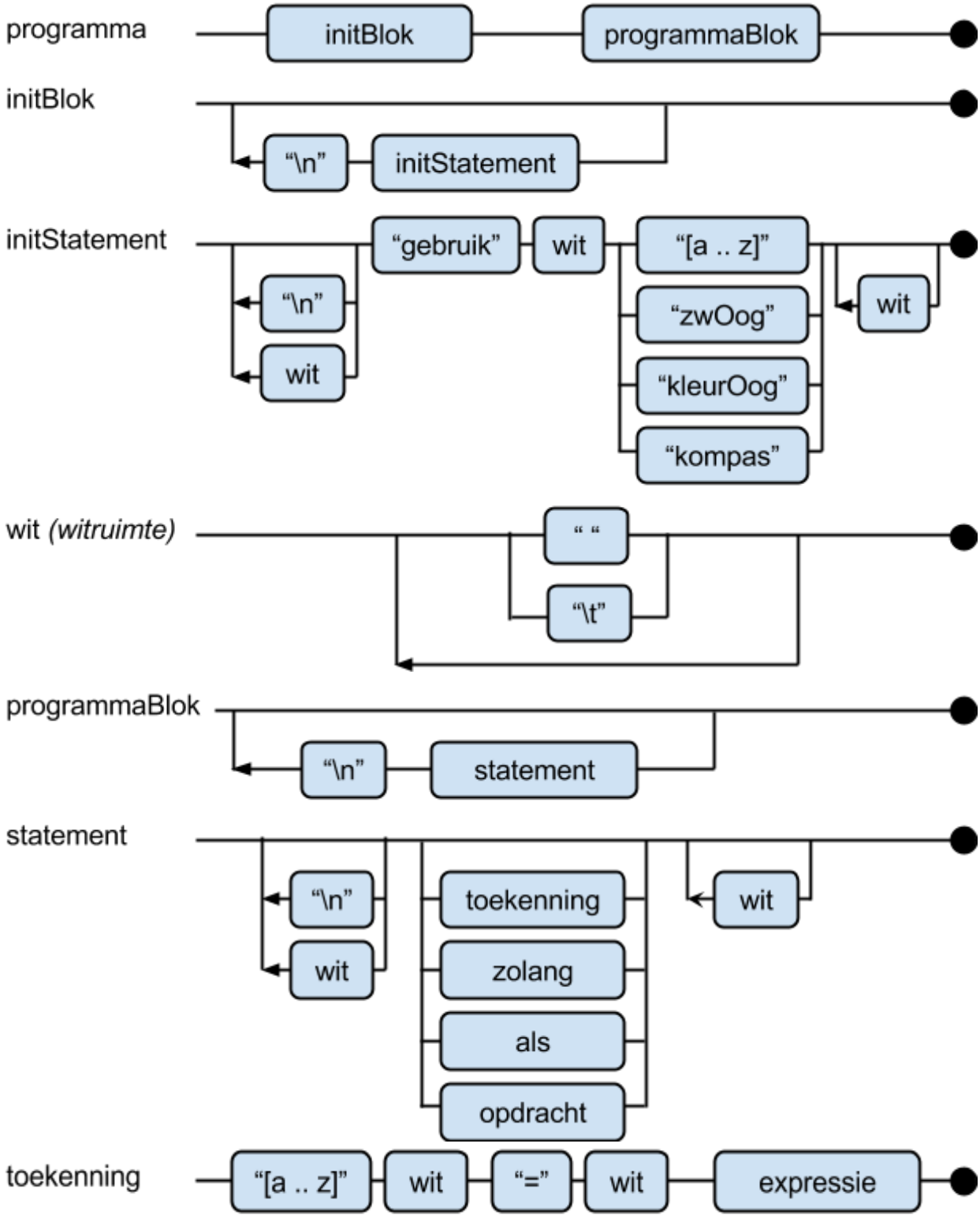
Nadat de eindscore door de examinator bekent gemaakt is kunnen jullie vragen om een motivatie. De examinator zal dan kort uitleggen hoe het cijfer tot stand is gekomen. Als jullie vinden dat de argumentatie niet klopt of als jullie een beoordeling per individuele student wensen dan kunnen jullie dit direct kenbaar maken door te vragen om een herkansing.

Deze herkansingen zullen plaatsvinden in de onderwijsweken die volgen op de challengeweek in de vorm van een assessment. De examinator zal jullie individueel of per groep vragen om op gesprek te komen. Tijdens dit gesprek zal de examinator a.d.h.v. de antwoorden die je / jullie geven op de vragen bepalen of de individuele beoordelingen verschillend moeten zijn en of jullie misschien wel aan de eindeisen voldoen maar geen voldoende hebben gekregen omdat de andere groepen excelleerden.

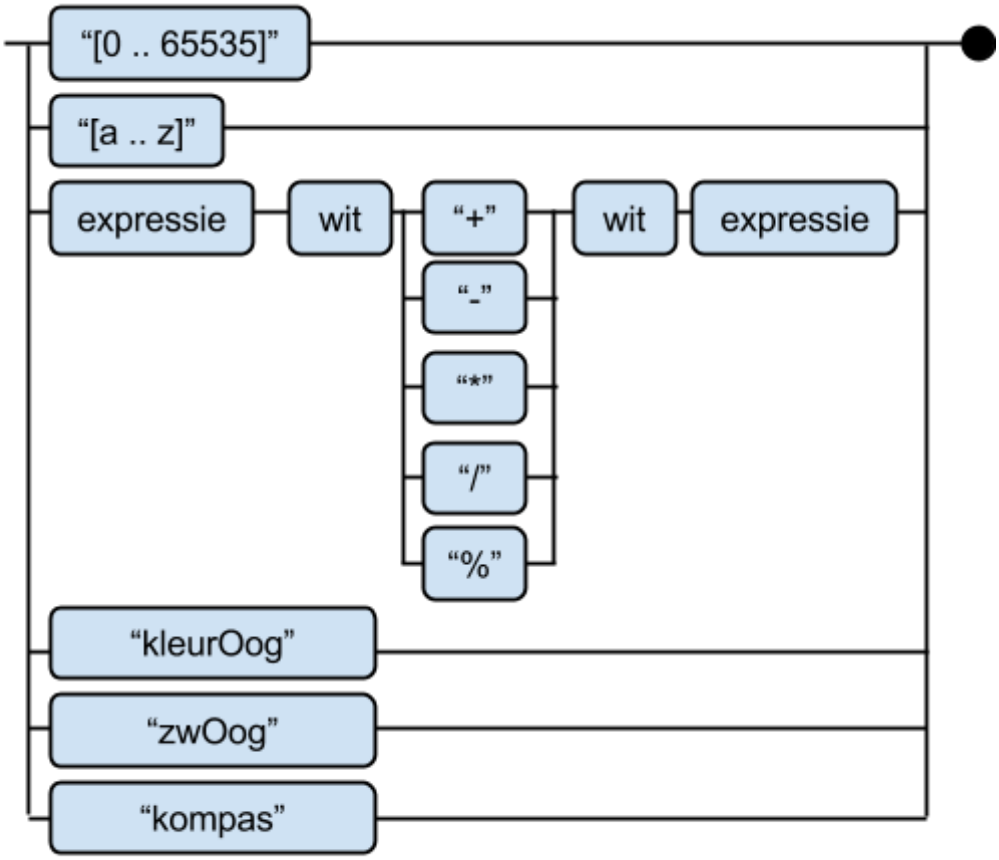
Je kunt je voorbereiden op de herkansing door:

- De film te bekijken op Pathé thuis.
- Deze handleiding aandachtig door te nemen, inclusief bijlagen.
- De website te bestuderen waarop de beste oplossingen bij de verschillende opdrachten staan.
- De aantekeningen van de colleges op te vragen bij je groepsgenoten, en deze te bestuderen.

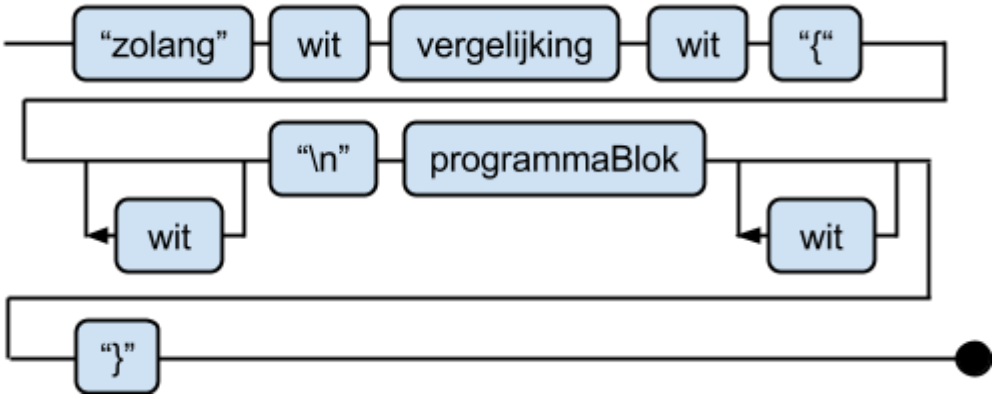
Bijlage 2: Syntax specificatie 20



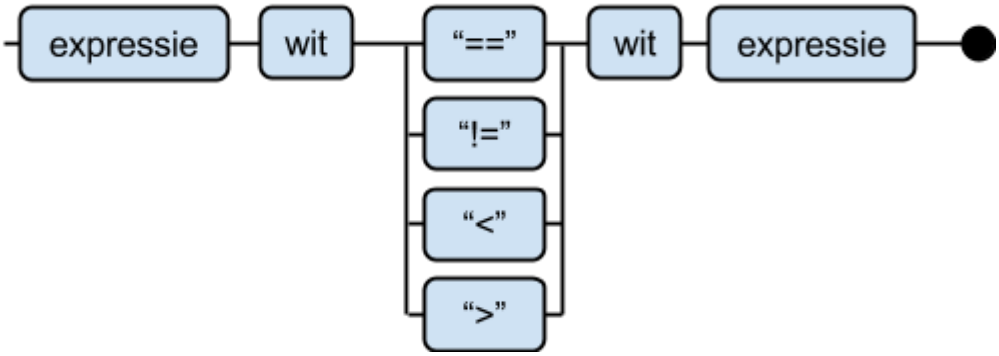
expressie



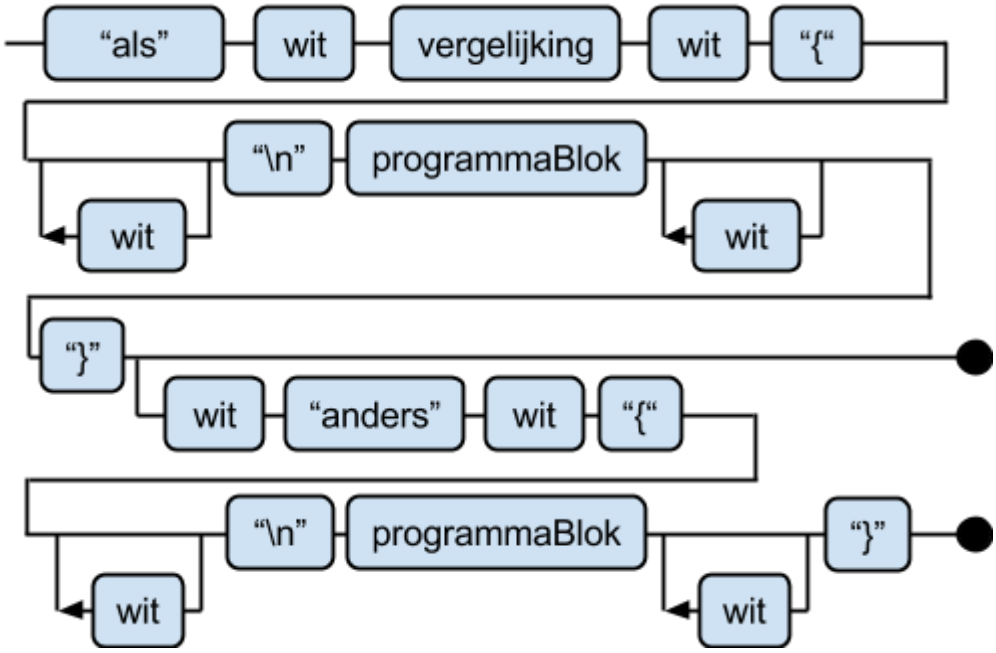
zolang



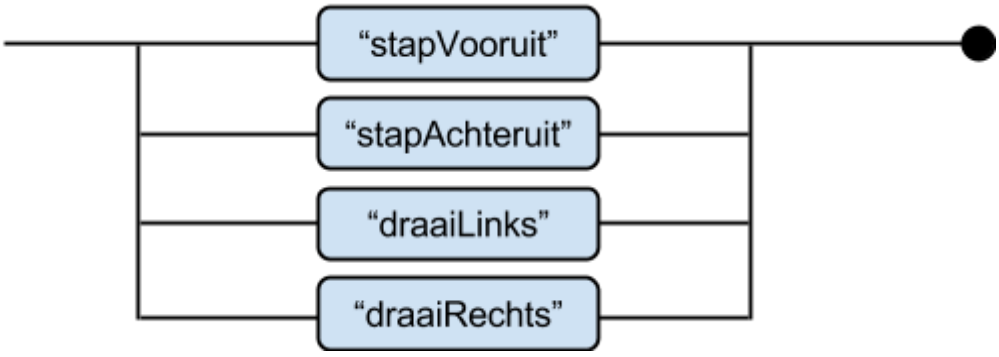
vergelijking



als



opdracht



Bijlage 3: (Veel)gestelde vragen

Naar aanleiding van de evaluatie van vorige jaren hebben we een aantal vragen beantwoord en hier verzameld. Het is verstandig om minimaal één groepslid te vragen om deze vragen en antwoorden door te nemen, zodat er bij het voeren van een discussie ook het perspectief van het onderwijs ontwerp kan worden benoemd.

Vraag: Zou het mogelijk zijn om eerder tijdens deze week een voorbeeld van een Glade te laten zien?

Deze vraag zou kunnen verwijzen naar pagina 13 van deze handleiding: **Voorbeelddefinitie glade.** Ook kan het zijn dat de student die deze vraag stelt niet heeft gekeken naar de voorbeelden in de zandbakomgeving. De belofte die wij doen is dat iedereen na dag 1 een beeld heeft van de challenge. Mocht dit niet zo zijn neem dan aan het begin van dag twee contact op met de docenten in bij de Balie in de lobby van gebouw XT

Vraag: Mag ik meer gelegenheden om mijn pogingen op te slaan?

Deel van de uitdaging is dat je maar 20 keer een eigen oplossing mag opslaan en maar 3 keer een oplossing voor iemand anders zijn opdracht. Wil je meer pogingen dan zul je zelf een programma moeten maken die je hiertoe in de gelegenheid stelt. De taal 20 is eenvoudig en compleet gedocumenteerd. Je kunt de opdrachten downloaden vanaf de website, vertalen naar je eigen datastructuur (b.v. een array van 20 bij 20) en vervolgens je programma testen door zelf de taal te interpreteren.

Vraag: Willen jullie de volgende keer duidelijker communiceren dat je slechts 20 regels code mag gebruiken voor je eigen Glade?

Dat willen we graag. Onderdeel van je eigen Glade is een programma waarmee je deze zelf oplost. Voor dit programma mag je maar **20 regels code** gebruiken. Je 'tegenstanders', de andere groepen, mogen bij het oplossen van jullie opdracht zoveel regels code schrijven als de kostenkaart toelaat.

Vraag: Hoe zijn de groepjes gemaakt?

De groepen zijn vooraf samengesteld. Er is dit jaar gekozen voor groepjes van 4 of 5 studenten. Deze groepen blijven intact gedurende het semester. Per klas zijn er 6 groepen.

Maar: Een nieuw team, t.o.v. vorig jaar, en competitie staat op gespannen voet met elkaar. Je kunt makkelijker samenwerken met mensen die je al kent.

Dat klopt, in de beroepspraktijk kun je ook niet altijd zelf je team samenstellen. Goede professionele vaardigheden maken het makkelijker voor je om samen te werken met diverse teamgenoten. Als software engineer ben je meestal bezig met het oplossen van problemen van iemand anders, althans als je met je werk geld wilt verdienen. Empathie, begrijpen voordat je begrepen wilt worden, luisteren en vertalen zijn als vaardigheid minstens zo belangrijk als programmeren, ontwerpen, advies geven, analyseren en het beheer van je software.

Vraag: Kunnen jullie de regels nog duidelijker uitleggen?

Nee, doe zelf een voorstel voor een betere uitleg, dan kunnen we kijken of deze 'beter' is.

Vraag: Kunnen we de volgende keer de zandbak overslaan, dat was een beetje saai?

Het zou korter kunnen maar dan zullen sommige groepen te weinig tijd hebben om zich voor te berijden en te begrijpen wat er van ze wordt verwacht. Als de zandbak saai is, programmeer dan een eigen interpreter voor de taal 20, b.v. in C# als vingeroefening voor de komende weken, Deze interpreter zal er straks voor zorgen dat je oneindig veel pogingen kan doen en daarmee heb je een groot voordeel.

Ben je al klaar met je interpreter? Train dan een deep neural network dat zelf een optimaal programma schrijft gegeven een Glade en kostenkaart. Dan kun je straks achteroverleunen en toekijken hoe jou AI de rest van de wereld verslaat.

Vraag: Een taal in het Nederlands (als, zolang), dat is toch vervelend?

Klopt. Dat zou niemand ooit doen en daarom doen wij het wel. Een leerdoel is nauwkeurigheid, door Nederlandse termen te verwerken in de taal ben je gedwongen om heel nauwkeurig te werken. Ben je niet zo nauwkeurig maar wil je voorkomen dat je veel pogingen verspeeld door eenvoudige syntaxfouten (spaties, newline, puntkomma's, if's en while's)? Dwing dan een code review af of schrijf een aantal regex regels die je programma kunnen testen voordat je ze probeert. Dit geeft je een groot voordeel ten opzichte van de groepen die geen controle uitvoeren.

Vraag: De foutmelding in het geval van een vergeten white space is onduidelijk. Kan de interpreter niet gewoon "Je mist een spatie" teruggeven?

Nauwkeurigheid is 1 van de leerdoelen van deze challenge. Als je niemand in de groep hebt die over genoeg concentratie beschikt om dit soort syntax fouten snel en betrouwbaar te kunnen herkennen, dan kun je met een aantal eenvoudige regex regels je code laten controleren voordat je deze opslaat. Dit zal veel fouten voorkomen en levert jullie een groot voordeel ten opzichte van de andere groepen.

Vraag: Waarom programmeren we in 20 en niet in C#?

Je mag natuurlijk programmeren in C#. Bouw tooling om je te helpen tijdens de wedstrijd. Een eenvoudige syntaxcontrole, een interpreter voor de taal 20 of een Neuraal netwerk dat voor jou de beste oplossing verzint. Er zijn tal van mogelijke tools die het doen van deze challenge kunnen vergemakkelijken het is aan jullie groepje om te kiezen welke aanpak haalbaar en succesvol kan zijn.

Vraag: Wat heeft deze challenge met mijn studie te maken?

Het begrijpen van taal, taalconstructies, turing compleetheid, syntax v.s. semantiek, willekeur en algoritmiek is fundamenteel voor de vaardigheid van het programmeren. Niet langer kun je leunen op de intelligentie van de ontvanger om te 'begrijpen' wat je wilt duidelijk maken, je zult precies en nauwkeurig gebruik moeten maken van de taal van het instrument om het te laten doen wat je opdrachtgever voor ogen heeft.

Als software engineer moet je loskomen van een specifieke taal zodat je flexibel inzetbaar bent en in staat om je te schikken in zo veel mogelijk verschillende contexten. Met minimale middelen moet je in staat zijn om complexe problemen op te lossen door deze te ontbinden in programmeerbare deelproblemen.