Praktische Opdracht

Mens Erger Je Niet

Informatica havo en vwo 4 periode 4

Harold Prins
Versie 1, voorjaar 2025

Inhoud

Inleiding 2
Programmeeropdracht week 1 - opdracht 1 Speelbord 3
Programmeeropdracht week 1 - opdracht 2 Dobbelsteen 4
Vragenblad week 1 5
Programmeeropdracht week 2 - opdracht 3 Ben ik er al? 7
Programmeeropdracht week 2 - opdracht 4 Voorbij positie 39 8
Programmeeropdracht week 2 - opdracht 5 Voorbij positie 39 9
Vragenblad week 2 10
Programmeeropdracht week 3 - opdracht 6 Spelfeedback 12
Programmeeropdracht week 3 - opdracht 7 De andere kleuren 13
Programmeeropdracht week 3 - opdracht 8 De andere kleuren – efficiënter 14
Vragenblad week 3 15
Programmeeropdracht week 4 - opdracht 9 De wachtrij en de eindrij 16
Programmeeropdracht week 4 - opdracht 10 De wachtrij met 1 pion 17
Programmeeropdracht week 4 - opdracht 11 De wachtrij met 4 pionnen 18
Programmeeropdracht week 4 - opdracht 12 Met 4 pionnen lopen 19
Vragenblad week 4 20
Programmeeropdracht week 5 - opdracht 13 De eindrij 21
Programmeeropdracht week 5 - opdracht 14 Elke positie 1 pion 22
Programmeeropdracht week 5 - opdracht 15 Puntjes op de i 23
Programmeeropdracht week 5 - opdracht 16 Weergave 24

Inleiding

Hier volgt de praktische opdracht MensErgerJeNiet. In principe moet je de opdrachten op volgorde maken. Bij elke opdracht staat voor welke week die in de periode is bedoeld.

Je kan per (deel)opdracht maximaal 5 punten krijgen. Er zijn 25 inlevermomenten. Sommige opdrachten lever je in het geheel in en andere opdrachten lever je in per deelopdracht in. Deze zijn goed zichtbaar op maken.mijn-in.nl. Totaal zijn er dus 125 punten te verdienen.

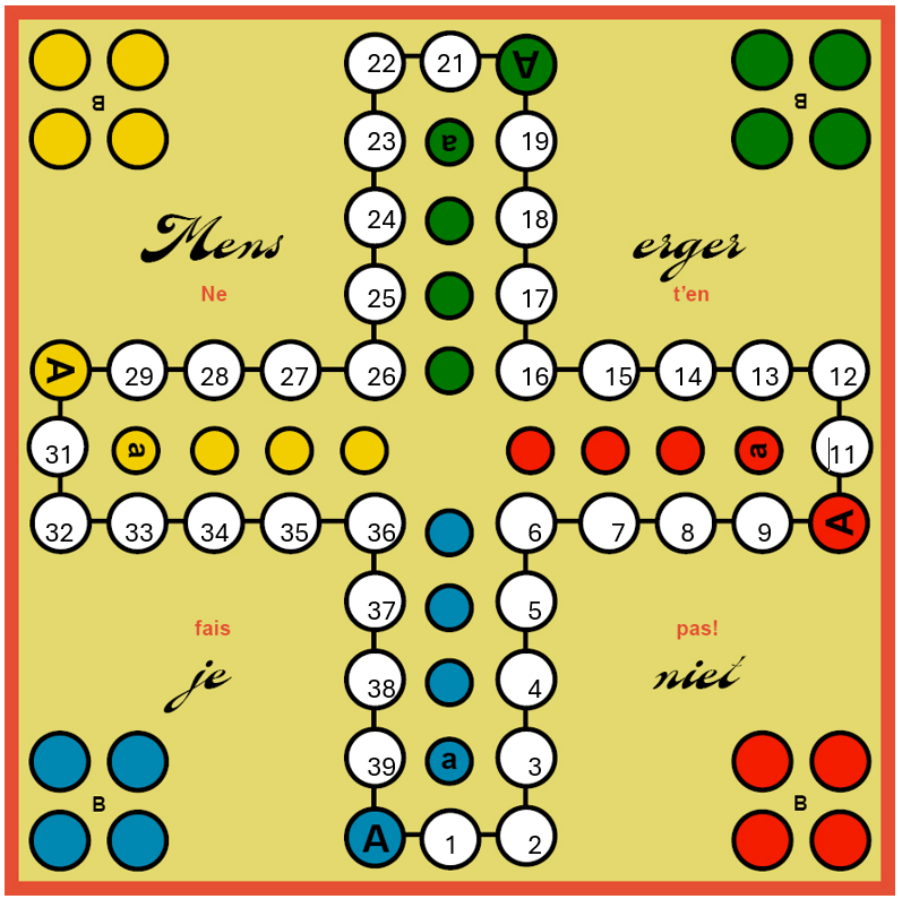
Je krijgt ook punten voor het op tijd inleveren. Op maken.mijn-in.nl kan je per opdracht zien wanneer ik die ingeleverd wil hebben. Ben je te laat dan kost je dat 1 van de 5 punten. Zorg dus dat je direct goed meedoet. Je mag eventueel later wel opdrachten opnieuw inleveren om dingen te verbeteren. Let op dat de eerste inlevering dan wel van voldoende kwaliteit is.

Op maken.mijn-in.nl laat ik ook al een cijfer zien. Die begint bij een 1 en bij elk punt dat je haalt gaat het cijfer omhoog. Dit cijfer is voor nu een indicatie. Deze opdracht is nog nieuw en ik moet zelf ook even ervaren hoe deze opdracht gaat werken.

Voor de opdracht heb ik ook wijzigingen aangebracht aan de software van maken.mijn-in.nl. Loop je tegen dingen aan die niet goed werken laat het dan zo snel mogelijk weten.

Heel veel succes!

Programmeeropdracht week 1 - opdracht 1 Speelbord



Hier zie je het spelbord van Mens erger je niet. Zoals je ziet kan je elke positie een nummer geven. Op die manier kan je het bord ook goed voor de computer digitaliseren. De blauwe A is positie 0 en dan ga je met de klok mee het bord rond. Jij gaat een functie maken die het bord tekent:

```
function toonSpelbord($blauw, $rood, $groen, $geel) {}
```

In de variabelen rood, blauw, groen en geel geef je aan waar de pion van die kleur zich op dat moment bevindt. Voor het gemak gaan we ervanuit dat er maar 1 kleur van elke pion op het bord aanwezig mag zijn en dat die pion er altijd is. De functie roepen we bij het starten van het spel als volgt aan:

```
toonSpelbord(0, 10, 20, 30);
```

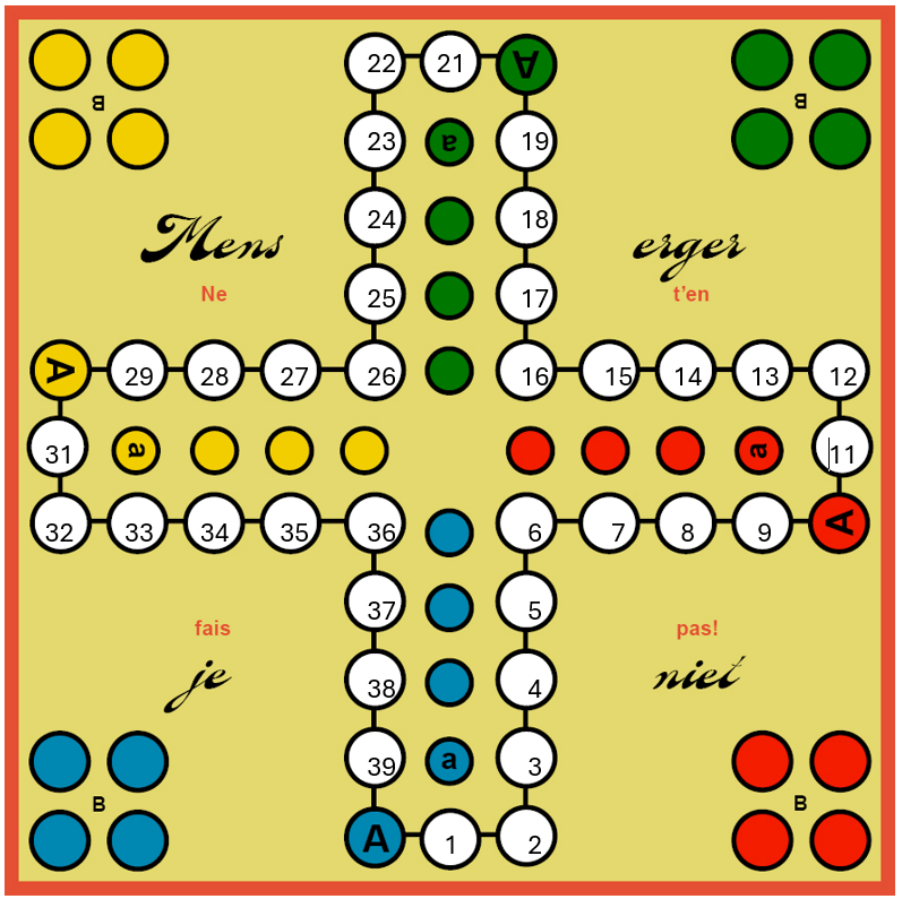
Het spelbord ziet er dan zo uit:

```
================================================================================
B . . . . . R . . . . . G . . . . . Y . . . . . .
================================================================================
```

De letters staan voor de 4 kleuren (we maken van de Engelse namen gebruik om dubbele letters te voorkomen).

Opdracht A: Zorg dat de functie toonSpelbord werkt. Het resultaat moet gelijk zijn aan bovenstaande voorbeeld en er moet gebruik gemaakt worden van de 4 variabelen voor de posities van de pionnen. Lever de functie in.

Programmeeropdracht week 1 - opdracht 2 Dobbelsteen



We gaan verder met de opdracht van het spel Mens erger je niet. Zodra alle pionnen op de startpositie staan moet de eerste speler (voor het gemak speler blauw) met de dobbelsteen gooien én dan zoveel stappen als gegooid vooruit plaatsen.

We gaan eerst de functie maken voor de dobbelsteen. Deze functie heet gooiDobbelsteen. Deze functie geeft een willekeurig getal terug tussen de 1 en 6. PHP heeft daarvoor een functie. Ga op zoek naar deze functie en pas die op de juiste manier toe.

Opdracht A: Zorg dat deze gooiDobbelsteen functie werkt en een willekeurig getal tussen de 1 en 6 teruggeeft. Roep de functie een aantal keren aan om te testen. Lever de functie met de test aanroepen in.

Opdracht B: Maak de mainloop van het spel. (We gaan alleen de blauwe pion bewegen).

1. Maak 4 variabelen voor elke startpositie van een kleur
2. Start de mainloop (stop de loop als de blauwe pion voorbij positie 39 is)
3. Toon het spelbord
4. Gooi met de dobbelsteen
5. Verhoog de positie van de blauwe pion met het aantal gegooide ogen
6. Hier herhalen we de loop en gaan terug naar stap 3

Lever de mainloop in.

Vragenblad week 1

1. Welke variabelen heb je gedeclareerd?

Variabele naam	Regelnummer	Scope

2. Welke keuze algoritmes start je, geef daarvan een korte beschrijving

3. Zitten er ook herhaalalgoritmes in je code? Geef ook daar dan een korte beschrijving van.

4. Op welke regels zit de output van je code, geef per regel aan waar de output over gaat.

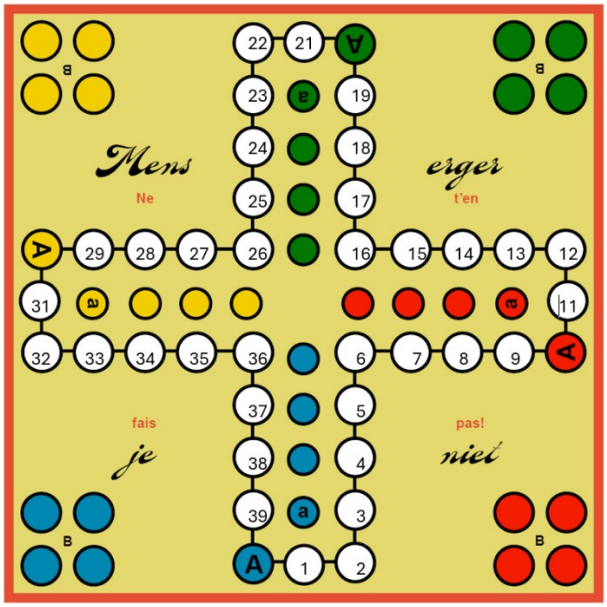
5. Zit er ook input in je code?

6. Wat vond je lastig?

7. Wat ging goed?

8. Controleer of je commentaar in de code helder is en pas deze zo nodig aan.

Programmeeropdracht week 2 - opdracht 3 Ben ik er al?



We gaan er eerst even vanuit dat de eindpositie de startpositie + 1 is. Dus blauw eindigt op 1. Daarmee kunnen we dus 41 stappen op het bord zetten.

Om te controleren of de worp geldig is maken we een nieuwe functie:

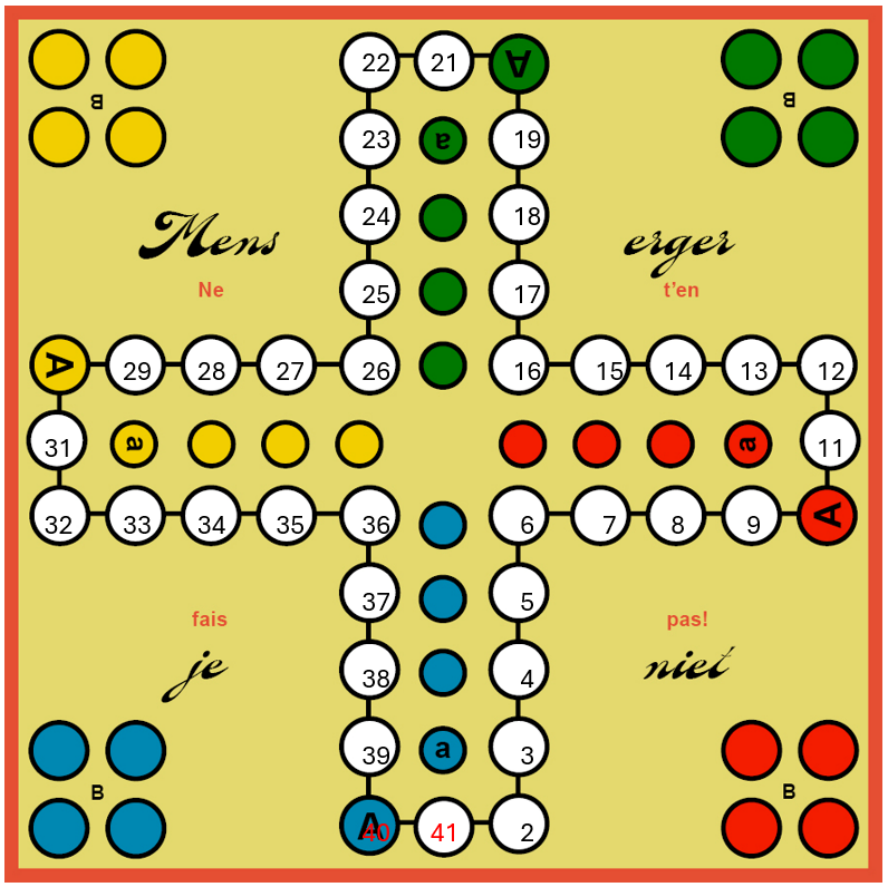
```
function kanBewegen($gezetteStappen, $dobbelsteen) {}
```

De functie krijgt mee het aantal al gezette stappen en de waarde van de dobbelsteen. De functie geeft een true terug als het aantal stappen is toegestaan en een false als dat niet zo is. Zie onderstaande tabel waarmee je je functie kan testen. Bedenk zelf ook nog een aantal testwaarden.

gezetteStappen	dobbelsteen	Toegestaan?
36	6	
36	5	
39	2	
39	4	

Opdracht A: Maak de functie kanBewegen en test deze goed. Lever de functie met de testaanroepen in.
Je kan echo niet gebruiken voor de output, omdat deze niets laat zien bij een false (bij true is het een 1). Gebruik in plaats daarvan de functie vardump().

Programmeeropdracht week 2 - opdracht 4 Voorbij positie 39



Het lastige is dat positie 0 en 1 zowel bij de start als aan het einde worden gebruikt. Dit kunnen we oplossen door deze posities dus ook de waardes 40 en 41 te geven. Bij het tonen van de positie kunnen we dit corrigeren. Daarom hebben we van alle kleuren minimaal 2 gegevens nodig:

1. De startpositie
2. Het aantal gezette stappen

Wat we nu moeten bereken is de huidige positie van de pion op basis van bovenstaande waarden.

start	gezetteStappen	huidigePositie
0	39	
0	40	
0	41	
10	39	
10	40	
10	41	
30	39	
30	41	

Opdracht A: Maak de functie huidigePositie om de huidige positie te berekenen op basis van de start positie en de gezette stappen en test deze goed. Lever de functie met de testaanroepen in.
Gebruik hier ook de functie vardump() voor de testresultaten.

Programmeeropdracht week 2 - opdracht 5 Voorbij positie 39

In opdracht 2B punt 7 stond de volgende opmerking: 'Stop de loop als de blauwe pion voorbij positie 39 is'. Dat is zoals je in opdracht 4 en 5 ziet geen juiste benadering. Alle pionnen moeten voorbij positie 39 kunnen komen. We moeten dus de mainloop aanpassen. We kunnen tegelijk dan ook de functies die je in opdracht 3 en 4 hebt gemaakt daaraan toevoegen.

Opdracht A: Pas de mainloop van het spel aan. (Eerst alleen de blauwe pion). Voor de helderheid heb ik het hele (aangepaste) stappenplan hier herhaalt

1. Maak 4 variabelen voor elke startpositie van een kleur
2. Maak nu ook 4 variabelen voor het aantal gezette stappen van elke kleur
3. Start de mainloop (stop deze nu als aantal gezette stappen van blauw 41 is)
4. Bereken de huidige positie van blauw
5. Toon het spelbord, vervang de variabele van de blauwe positie door het antwoord van de vorige stap
6. Gooi met de dobbelsteen
7. Controleer met de functie kanBewegen of dit is toegestaan
8. Indien toegestaan verhoog gezetten stappen van de blauwe pion met het aantal gegooide ogen
9. Hier herhalen we de loop en gaan terug naar stap 4

Lever de nieuwe mainloop in.

Opdracht B: Stappen 6 t/m 8 lenen zich ook goed voor een eigen functie, vooral omdat je dit ook voor de andere 3 kleuren nog moete doen. Maak de functie speelBeurt($kleur, $gezetteStappen) en neem daarin deze stappen op. In de variabele $kleur kan je de waarde "blauw", "rood", "groen" of "geel" meegeven. Deze gaan we later gebruiken.

Lever de functie speelBeurt in.

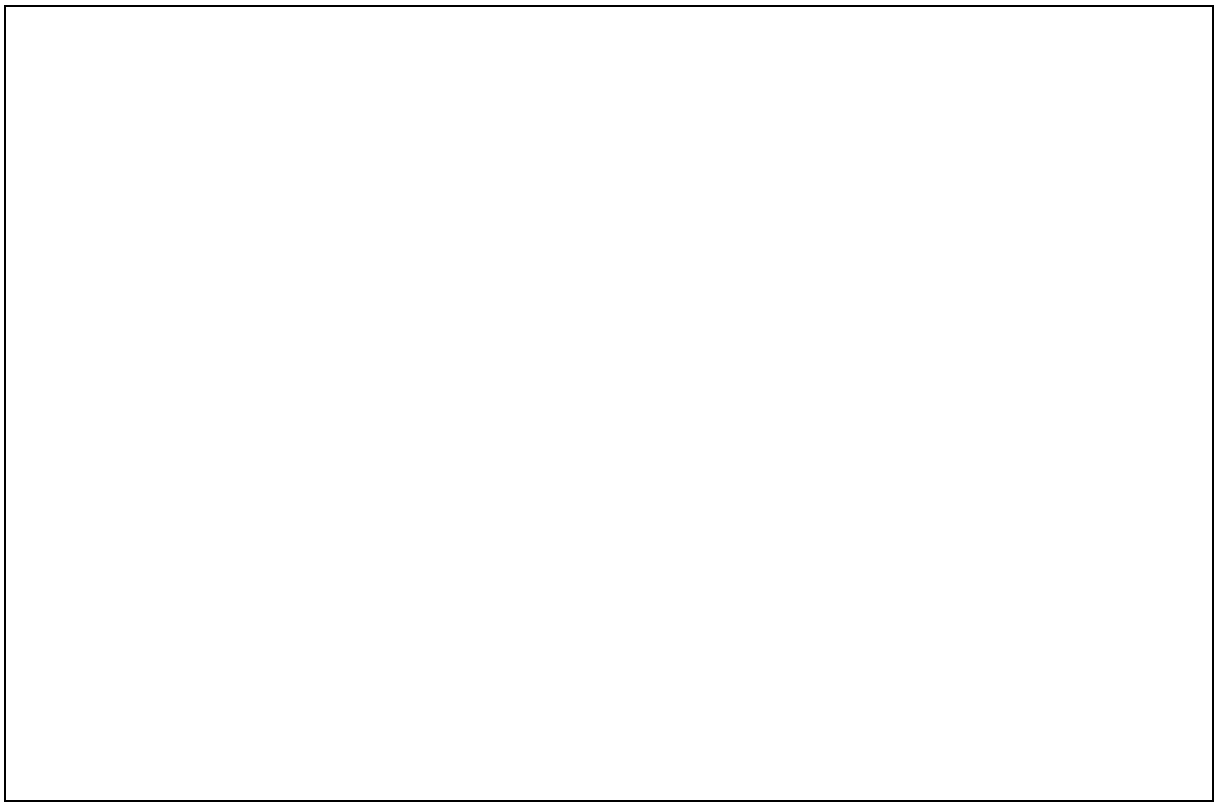
Opdracht C: Pas de mainloop aan door de functie speelBeurt te gebruiken. Nog steeds alleen de kleur blauw.

Lever de nieuwe mainloop in.

Vragenblad week 2

Deze vragen gaan over de functies speelBeurt() en kanBewegen() en over de mainloop.

1.Maak een stroomdiagram van de functie speelBeurt()



2. Maak een stroomdiagram van de functie kanBewegen()

3. Leg uit hoe het kan dat de mainloop niet de laatste stand van het speelbord laat zien.

3. Wat vond je lastig?

4. Wat ging goed?

5. Controleer of je commentaar in de code helder is en pas deze zo nodig aan.

Programmeeropdracht week 3 - opdracht 6 Spelfeedback

Het spel toont op dit moment alleen het speelbord. Andere feedback geven we niet. Het zou wel mooi zijn om dit ook zichtbaar te maken in het spel. Voeg daarom de volgende output toe aan het script:

1. Wat er is gegooid met de dobbelsteen.
2. Een foutmelding als het verplaatsen van de pion niet is toegestaan.

Met de volgende code kan je het cli script vertragen en het scherm (min of meer) leeg maken (let op dit stukje algoritme kan van platform tot platform verschillen):

```
// Clear cli
sleep(1);
echo "\033[2J\033[H";
```

Opdracht A: Voeg de meldingen op een logische plek toe.

Opdracht B: Controleer de uitkomsten van je script nu goed. Is alle uitvoer zoals je verwacht?

Opdracht C*: Het zou nog mooi zijn om op het spelbord aan te geven waar de startposities van de 4 kleuren zich bevinden.

Lever het totale script wat je nu hebt in.

Programmeeropdracht week 3 - opdracht 7 De andere kleuren

Tot nu hebben we alleen de blauwe pion verzet. Maar dit willen we natuurlijk ook met de andere pionnen. Het éénvoudigste is om de code in de mainloop te kopiëren en ook de andere kleuren eraan toe te voegen. Hieronder volgt wat we hadden als we alleen de blauwe pion verzetten:

1. Maak 4 variabelen voor elke startpositie van een kleur
2. Maak nu ook 4 variabelen voor het aantal gezette stappen van elke kleur
3. Start de mainloop (stop deze nu als aantal gezette stappen van blauw 41 is)
4. Bereken de huidige positie van blauw
5. Toon het spelbord, vervang de variabele van de blauwe positie door het antwoord van de vorige stap
6. speelBeurt() en verhoog de waarde van het aantal gezette stappen van blauw
7. wacht 1 seconde en maak daarna de cli leeg
8. Hier herhalen we de loop en gaan terug naar stap 4

Op het vragenblad van week 2 werd al de vraag gesteld hoe het kan dat deze loop de laatste stap van blauw in het spel niet laat zien. De reden is dat we eerst het spelbord laten zien en daarna de beurt spelen. Het is beter om dat aan te passen. Het gevolg is wel dat nu de eerste beurt (de startposities dus) niet worden getoond. Dat kan je oplossen door het speelbord te tonen voordat de mainloop start. Als het goed is kan je er nu dit van maken (in geel de wijzigingen, let op dat 5 al in je script stond maar is verplaatst):

1. Maak 4 variabelen voor elke startpositie van een kleur
2. Maak nu ook 4 variabelen voor het aantal gezette stappen van elke kleur
3. Toon het spelbord met de startposities
4. Start de mainloop (stop deze nu als aantal gezette stappen van blauw 41 is)
5. speelBeurt() en verhoog de waarde van het aantal gezette stappen van blauw
6. Bereken de huidige positie van blauw
7. Toon het spelbord, vervang de variabele van de blauwe positie door het antwoord van de vorige stap
8. wacht 1 seconde en maak daarna de cli leeg
9. Hier herhalen we de loop en gaan terug naar stap 4

Het toevoegen van de andere kleuren is nu ook eenvoudig door het onderste toe te voegen na regel 8:

9. speelBeurt() en verhoog de waarde van het aantal gezette stappen van **rood**
10. Bereken de huidige positie van **rood**
11. Toon het spelbord gebruik de juiste variabelen
12. wacht 1 seconde en maak daarna de cli leeg
13. … en doe dit ook voor de kleuren groen en geel

Je zult nu veel regels zien die op elkaar lijken. Hoe we dat anders kunnen doen zien we in de volgende opdracht.

Opdracht A: Volg de aanwijzingen zoals hierboven aangegeven en wijzig de mainloop en voeg de andere 3 kleuren ook toe. Lever de nieuwe mainloop in.

Programmeeropdracht week 3 - opdracht 8 De andere kleuren – efficiënter

Begin alleen aan deze opdracht als je zeker weet dat je code uit opdracht 7 goed werkt. Zoals je nu in je code kan zien herhaal je in de loop telkens deze regels:

1. speelBeurt() en verhoog de waarde van het aantal gezette stappen van **kleur**
2. Bereken de huidige positie van **kleur**
3. Toon het spelbord gebruik de juiste variabelen
4. wacht 1 seconde en maak daarna de cli leeg
5. … en doe dit ook voor de kleuren groen en geel

Regel 3-5 zullen telkens gelijk zijn, maar dat geldt niet voor de regels 1 en 2. Daarin gebruik je immers telkens een andere kleur.

Oplossing 1

Er is een optie om in PHP een variabele te gebruiken voor een variabele naam. Je hebt bijvoorbeeld de volgende variabele:

```
$startBlauw = 0;
```

Je kan voor de kleurnaam ook een variabele gebruiken door de naam van de variabele tussen {} (accolades) te zetten:

```
$kleur = 'Blauw';
${'start' . $kleur} = 0;
```

Dit geeft de mogelijkheid om door lijstje met alle kleuren heen te gaan met foreach:

```
foreach([' Blauw', ' Rood', 'Groen', 'Geel'] as $kleur) {
     // hier wordt de code voor elke kleur uitgevoerd
}
```

Oplossing 2 (eigenlijk een betere oplossing)

Er is nog een betere oplossing door niet met losse variabelen te werken voor de startposities en het aantal gezette stappen, maar met een lijst waarin je in plaats van getallen de kleurnamen als indexen gebruikt.

```
$startPosities = ['blauw' => 0, 'rood' => 10, 'groen' => 20, 'geel' => 30];
```

Je kan nu de startPositie van blauw op de volgende manier vinden:

```
echo $startPositie['blauw'];
```

Of in plaats van de letterlijke naam kan je ook een variabele gebruiken:

```
echo $startPositie[$kleur];
```

En dat geeft weer de mogelijkheid om de foreach te gebruiken uit oplossing 1.

Opdracht A: Maak een keuze uit bovenstaande oplossingen en voer die uit. Lever de nieuwe mainloop in.

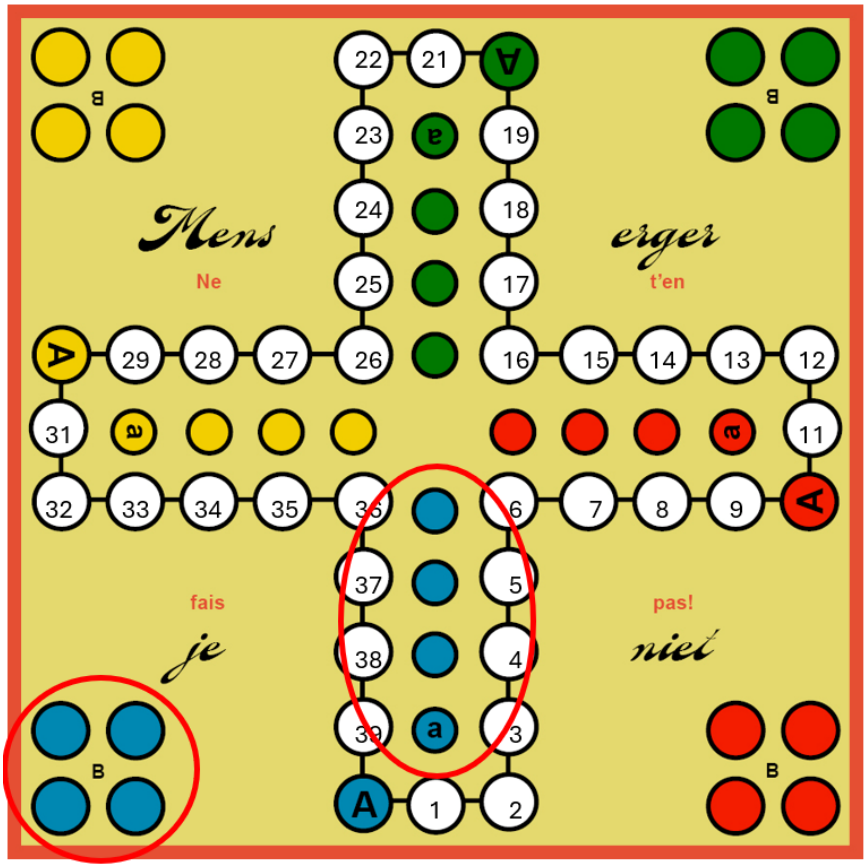
Vragenblad week 3

1. Leg uit hoe je nu goed kan testen of het spel op het juiste moment eindigt.

2. In opdracht 7 heb je de volgorde van de mainloop gewijzigd. Leg uit hoe het kan dat je het tonen van het bord 1 keer buiten de mainloop moet doen.

3. Opdracht 8 is best een lastige opdracht. Voor welke oplossing heb jij gekozen, leg uit waarom je daarvoor hebt gekozen.

Programmeeropdracht week 4 - opdracht 9 De wachtrij en de eindrij



Deze week gaan we de wachtrij (linksonder) en de eindrij toevoegen. In deze eerste opdracht beginnen we met de visuele weergave. Het is tijdens het ontwikkelen misschien wel prettig om de wachttijd en het wissen van de cli tijdelijk uit te schakelen.

Het eindresultaat van deze opdracht ziet er zo uit:

```
==========================================================================
B . . . . . R . . . . . G . . . . . Y . . . . .
B _         R _         G _         Y _
B _         R _         G _         Y _
B _         R _         G _         Y _
B _         R _         G _         Y _

==========================================================================
```

Er staan 4 pionen van elke kleur in de wachtrij visueel weergegeven door de letter van de kleur. En de eindrij is leeg, visueel weergegeven door de _ (underscore). Op het bord staat ook nog een pion die halen we later weg.

Opdracht A: Wijzig de functie toonSpelbord. Voeg de wachtrij en de eindrij aan de weergave toe.

Programmeeropdracht week 4 - opdracht 10 De wachtrij met 1 pion

Het visueel toevoegen van de wachtrij en eindrij is uiteraard niet voldoende. Er gebeurt in het programma nu nog niets met deze rijen. Tot nu toe werkten met 4 pionnen. Voor elke kleur één. Nu gaan we met 4 pionnen werken per kleur. Dat betekent dus 16 pionnen totaal. Voor elke pion moeten we bijhouden wat de positie is. Daarvoor gebruiken we nu variabelen voor het aantal gezette stappen en we rekenen dan uit wat de huidige positie is.

Ook het spelen van de beurt wat complexer. Je moet namelijk nu per speler gaan bepalen welke pion er verzet moet worden of dat er een nieuwe pion op het bord geplaatst moet worden.

Dat laatste is ook nieuw, tot nu toe plaatsen we de pion gewoon op het bord. Nu moeten we dat pas doen als er een keer 6 wordt gegooid.

In deze opdracht gaan we nog even door met 1 pion per kleur die pas op het bord komt als we 6 hebben gegooid. En we bereiden de code alvast voor om te kunnen werken met 16 pionnen.

Opdracht A: Wijzig de initialisatie (= variabelen die we voor de mainloop hebben gedeclareerd) van gezette stappen van de waarde 0 naar -1. We gebruiken -1 om aan te geven dat de pion nog in de wachtrij zit.

Opdracht B: Wijzig de functie speelBeurt. Er is een extra controle nodig of het aantal gezette stappen nog -1 is. Dan is verplaatsen alleen toegestaan als er 6 wordt gegooid. Op dat moment moet het aantal gezette stappen op 0 worden gezet.

Opdracht C: Ook de functie huidigePositie moet worden aangepast. Als het aantal gezette stappen namelijk -1 is moet deze functie ook -1 retourneren.

Opdracht D: En de functie toonSpeelbeurt moet daar dan ook op worden aangepast. Als het goed is gaat het tonen van het spelbord al goed, omdat die geen positie -1 kent. Maar in het tonen van de wachtrij moet je de pion weergeven (letter van de kleur) of een lege positie (underscore). Je moet dus controleren of de pion zich nog in de wachtrij bevindt. (Voor het gemak kan je dat nu nog doen voor alle pionnen in de wachtrij.)

Opdracht E: Controleer visueel of je algoritme nu klopt. Controleer vooral ook je startpositie aan het begin van het spel. Mogelijk dat deze nog moet worden aangepast. Wijzig eventueel de inleveringen bij A t/m D.

Programmeeropdracht week 4 - opdracht 11 De wachtrij met 4 pionnen

In deze opdracht gaan we van 4 naar 16 pionnen. Het nadeel van het huidige script is dat we zowel het aantal gezette stappen als de huidige positie bijhouden in de mainloop. Als we dat zo laten moeten we dat dus voor 16 pionnen doen. Dat is zeker een optie. In deze opdracht kies ik ervoor om de berekening van de huidige positie te verplaatsen naar de functie toonSpeelbord en in de mainloop alleen nog het aantal gezette stappen per pion bij te houden. We gaan daarvoor per kleur een lijstje bijhouden. Als je in opdracht 8 voor optie 2 hebt gekozen dan moet je dit net een beetje anders uitwerken. In de uitleg ga ik ervan uit dat er sprake is van optie 1 uit opdracht 8. Dit is een opdracht met wat uitdaging. Daarom stapsgewijs.

Opdracht A: We gaan de mainloop aanpassen:
1. We initiëren alleen nog de variabele voor het aantal gezette stappen, maar in plaats van alleen een -1 nemen we nu een lijst op voor 4 pionnen: [-1,-1,-1,-1]
2. Uit de mainloop wissen we de functie voor het berekenen van de huidige positie. En we geven aan de functie toonSpeelbord nu de variabele voor het aantal gezette stappen mee.
3. De while loop voorwaarde moet je nu ook aanpassen. Het eenvoudigste is nu dat je een index 0 aan de variabelen toevoegt.

Opdracht B: We gaan de functie toonSpeelbord ingrijpend wijzigen.
1. De start positie uit opdracht A1 (zie hierboven) van de kleuren plaatsen we nu bovenin deze functie.
2. Maak een algoritme waarin je alle pionPosities berekent, zie het uitgeschreven voorbeeld onderaan dit blad.
3. Pas nu de loop aan waarin je de posities laat zien, let op dat je daar dus nu de variabele pionPosities kan gebruiken en je dit dus niet meer per kleur hoeft te doen.
4. De loop van de beginrij pas je nu eenvoudig aan door bij de variabele ook de index die al in de loop zit toe te voegen.

Opdracht C: We zullen ook de speelBeurt functie moeten aanpassen omdat deze nu een lijst als invoer krijgt. We laten het spelen met 4 pionnen voor een volgende opdracht, dus we voegen nu eenvoudig gewoon overal een index 0 toe voor de eerste pion. Om het spel te testen kan je hier de index ook wijzigen in 1, 2 of 3 om te zien dat je nu met elke pion kan lopen. Daarom is het handig om de index met een variabele toe te voegen. (let op dat je dit dan in de while loop van de mainloop ook moet aanpassen.)

Algoritme voor pionPosities = []:

Loop door alle 4 kleuren ['R' => 'Rood', 'B' => 'Blauw', 'G' => 'Groen', 'Y' => 'Geel'], je kan in een foreach ook de index (hier de letters) meegeven.
    Loop door alle pionnen (4 pionnen per kleur)
        Bereken de huidige positie van de pion met de functie die je daarvoor al had gemaakt.
        Sla de positie op in pionPosities (bijv: $pionPosities[$huidigePos] = "$key ";).
    Einde loop
Einde loop

Programmeeropdracht week 4 - opdracht 12 Met 4 pionnen lopen

In deze opdracht gaan we alle 4 pionnen gebruiken. Daarvoor moeten we de functie speelBeurt aanpassen. Deze krijgt 2 variabelen mee: kleur en gezetteStappen. De focus in deze opdracht ligt op de laatste 'gezetteStappen'. Bij de eerste beurt van een kleur heeft deze de initialisatiewaarde. Dat is een lijstje voor 4 pionnen met de waarde -1 per pion. Dat ziet er dan zo uit: [-1,-1,-1,-1].

Hieronder heb ik het algoritme van de functie beschreven:

1. Als er 6 wordt gegooid moeten we in het lijstje kijken of er nog een pion is met -1. Als dat zo is dan moeten we deze pion op het spelbord zetten. Anders moet er een keuze gemaakt worden welke pion we vooruit willen zetten.
2. Anders
    a. We maken een lijst van alle pionnen die een gezette stappen hebben van 0 t/m 40.

    b. Als deze lijst groter is dan 1 dan vragen we aan de speler welke pion hij wil verzetten van de pionnen die op het spelbord staan.
    c. Als deze lijst precies gelijk is aan 1 dan gaan die pion verzetten.
    d. Anders verzetten we geen pion en laten we een melding zien dat het helaas niet mogelijk is om een pion te verplaatsen.

    e. We nemen de gekozen pion en controleren met de functie kanBewegen of het mogelijk is om deze te verplaatsen.
    f. Anders tonen we een melding dat het niet mogelijk was om deze pion te verplaatsen.
3. We geven de lijst met gezette stappen weer terug.

Opdracht A: Werk de functie gezetteStappen bij volgens bovenstaande beschrijving.

Opdracht B: In de mainloop wordt in de controle nog steeds gekeken naar alleen de eerste pion. Kun je deze ook zo aanpassen dat hij kijkt of alle pionnen zijn aangekomen. Het is nu wel een extra uitdaging om je script te testen. Bedenk hoe je zonder dat je het hele spel hoeft te spelen toch kan controleren of je while loop op het juiste moment stopt.

Vragenblad week 4

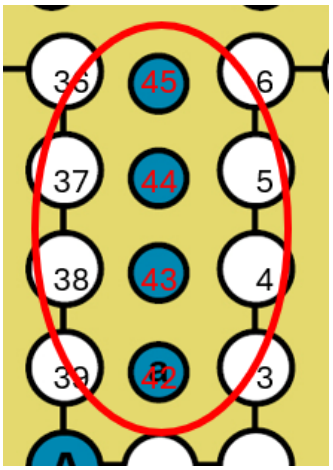
1. Leg je oplossing uit die je hebt gekozen voor vraag 12B.

2. Leg met de huidige code uit hoe je goed kan testen of het spel met meerdere pionnen goed werkt en op het juiste moment eindigt.

3. Is het nog nodig om je code in de mainloop te pauzeren? Leg je antwoord uit.

4. We hebben de eindrij nog niet geprogrammeerd. Als je nu naar de code kijkt wat zou jij dan ongeveer aanpassen om de eindrij ook te laten werken.

Programmeeropdracht week 5 - opdracht 13 De eindrij



Het meest logische is om het bord te vergroten en ook de eindhokjes een nummer te geven. Tot nu toe stopten we als we op vakje 41 kwamen. Maar de eigenlijke regel is dat een pion aan het einde ook kan teruglopen (alleen dan mag je achteruit). Een pion die in de eindrij staat mag in een volgende beurt niet meer verplaatst worden. We negeren nog even dat pionnen niet op hetzelfde vakje mogen staan.

Opdracht A: Wat moet je aanpassen in de functie kanBewegen? En is deze eigenlijk nog wel nodig?

Wat moet je aanpassen in de functie huidigePositie?

Wat moet je aanpassen in de functie speelBeurt?

En wat moeten we aanpassen om te controleren of de mainloop moet stoppen?

Je zal nu ook de functie toonSpelbord moeten aanpassen.

Programmeeropdracht week 5 - opdracht 14 Elke positie 1 pion

We moeten nu nog de spelregels helemaal juist gaan toepassen. Dat betekent dat we een aantal dingen nog niet hebben geregeld. Op elk vakje mag namelijk maar 1 pion staan. Als er al een pion van een andere speler staat dan moet die weer terug naar de startrij. Als je eigen pion er staat dan mag je die verplaatsing niet doen.

Opdracht A: Pas deze spelregel op de juiste wijze toe in het algoritme.

Opdracht B: Kun je het nu ook zo aanpassen dat de gebruiker geen foute keuzes kan maken of dat hij die kan corrigeren?

Programmeeropdracht week 5 - opdracht 15 Puntjes op de i

Er zijn nu nog wat kleine puntjes die we moeten oplossen om helemaal aan de spelregels te voldoen.

Opdracht A: Als je 6 gooit mag je kiezen wat je doet (pion op het bord plaatsen of met een andere pion lopen) en je mag nog een keer gooien. Het is hier even de vraag wat je moet doen als je nog een keer 6 gooit.

Programmeeropdracht week 5 - opdracht 16 Weergave

Opdracht A: Als laatste kunnen we het bord en de weergave natuurlijk nog sterk verbeteren zodat er een echt spelbord ontstaat.