# Stap 1:

Begin met het simuleren van een kaartspel. Een kaartspel heeft 52 kaarten, 4x de waarden 1 t.e.m 13.  
Momenteel houden we de beeldjes op een integer waarde.  
Maak een array kaartspel en zorg ervoor dat hier 4x de waarden 1 t.e.m 13 in zitten.  
Gebruik hiervoor een dubbele for loop.  
controleer of er 52 waardes in het kaartspel zitten aan de hand van de functie count().  
gebruik de functie array\_count\_values() om na te gaan of iedere waarde 4x voorkomt.  
doe een var\_dump van kaartspel en kijk of iedere waarde 4x voorkomt.

Var\_dump(array\_count\_values($kaartspel)) moet volgende output geven:  
array(13) {

[1] => int(4)   
 [2] => int(4)  
 … (3 tem 12)  
 [13] =>int(4)  
 }

Type aantal elementen  
index/key value/waarde

Tip: de functie array\_count\_values() zal de hele array aflopen en bijhouden hoeveel keer iedere waarde voorkomt. Deze functie geeft een associatieve array terug met als key de waarde en als value hoevaak deze voorkomt in de array kaartspel.

# Stap 2:

Sorteer het kaartspel met de sort() functie.  
Maak een variabele kaart en laat deze een random\_int() bevatten. De argumenten voor deze functie zijn de eerste en laatste index(key) van het kaartspel (doe var\_dump() van het kaartspel als je niet zeker bent).  
Selecteer een random kaart met behulp van de random\_int(0, 51) functie en geef deze aan de speler.  
Verwijder nu de geselecteerde kaart uit het kaartspel met behulp van de unset() functie.  
print de waarde van de kaart.  
Doe nu een var\_dump() van het kaartspel. Let erop dat de kaart die je getrokken hebt niet meer in het kaartspel aanwezig is. (indien kaart 10 is, mag er geen index 10 meer aanwezig zijn in het kaartspel).  
sorteer het kaartspel met sort() en doe nogmaals een var\_dump() van kaartspel. Let nu op dat index 10 opnieuw aanwezig is, maar index 51 niet meer.   
De kaarten zijn opgeschoven om het gat te vullen, let wel op dat de sort functie ook gaat sorteren op waarde. Daarom zullen we in stap 4 een andere functie gebruiken.  
Doe nu een var\_dump() van speler. Deze moet nu 1 kaart hebben met de waarde gelijk aan kaart/4 afgerond naar boven. Indien kaart/4 0 is, moet de waarde van de kaart van de speler gelijk zijn aan 1.

Doordat we het kaartspel eerst gesorteerd hadden zaten alle 1 vooraan, gevolgd door alle 2,.. en alle 13 achteraan. Wanneer we dan kaart (index) 10 trokken is dit een 3 (10/4 afgerond naar boven).

# Stap 3:

Tijd om de kaarten te verdelen.   
Maak een array speler en een array computer.  
Geef de speler 26 kaarten. Doe dit door steeds 1 random kaart uit het kaartspel te nemen en deze aan de speler te geven.  
Let op: deze kaart moet je uit het kaartspel verwijderen (gebruik hiervoor de unset() functie, vervolgens moet je het kaartspel opnieuw sorteren met de sort() functie.  
De overblijvende kaarten mag je zo aan de computer geven.  
Controleer met de count() functie dat zowel speler als computer 26 kaarten hebben.  
Controleer met de array\_count() functie dat je nog steeds een geldig kaartspel hebt.

Het is nu mogelijk dat computer of speler geen 13 verschillende kaarten meer heeft (mogelijks heeft deze dan 4x dezelfde van een andere kaart).

Tip: Gebruik het aantal kaarten-1 in het kaartspel als bovengrens voor het selecteren van een kaart.

# Stap 4:

Tijd om te spelen.  
Maak een loop zodat speler en computer elk 3x de eerste kaart uit hun hand nemen.  
De hoogste kaart wint.  
Wanneer beide kaarten gelijk zijn, keren ze terug naar het eigen hand, achteraan de stapel.  
De winnaar mag zijn kaart én de kaart van de verliezer opnieuw achteraan in zijn kaartspel aansluiten. Voeg eerst de eigen kaart toe daarna de kaart van de verliezer.  
Na 3 rondes wordt de winnaar bekend gemaakt, de winnaar is diegene met het meeste kaarten.  
Op het einde van het spel zorg je ervoor dat je output er als volgt uit ziet:  
X wint van Y met a kaarten tegen b kaarten

(X is de winnaar, Y is de verliezer, a is het aantal kaarten van de winnaar, b is het aantal kaarten van de verliezer).

Indien beide nog evenveel kaarten hebben moet de output er als volgt uitzien:  
Er is geen winnaar Speler heeft nog a kaarten en Computer heeft ook nog b kaarten.

(a is het aantal kaarten van Speler, b is het aantal kaarten van Computer)

Tip: Wanneer de speler zijn kaart trekt, verwijder deze uit zijn. Doe dit ook voor de computer.   
Let wel op dat je de juiste kaart verwijdert.  
Indien nodig controleer je met de var\_dump() functie.

Tip: om het gat in de array te dichten zonder te sorteren op waarde, gebruik de array\_values() functie en wijs de uitkomst hiervan opnieuw toe aan de array. Doe dit telkens vooraleer een nieuwe kaart te trekken.

Tip: het totaal aantal kaarten moet altijd 52 zijn.

# Stap 5:

Zorg er nu voor dat het spel blijft duren tot de speler de waarde ‘q’ ingeeft.  
Vraag hiervoor telkens of de speler wilt verder spelen.  
Wanneer de speler q ingeeft, moet het spel stoppen en de output van stap 4 geven.

# Stap 6: uitbreiding

Het spel en de spelregels veranderen niet.  
Wat wel verandert is de manier waarop we de winnaar berekenen.  
In vorige stap was de winnaar diegene die het meeste kaarten had.  
Zorg er nu voor dat de winnaar diegene is die de grootste waarde aan kaarten heeft.  
(Ookal heeft computer alle kaarten van 1 tot en met 9 zal de speler winnen)

Alle kaarten van 1 tot en met 9 zijn 36 kaarten met een totale waarde van 180.  
de kaarten van 10 tot en met 13 zijn16 kaarten met een totale waarde van 184.

Na afloop van het spel moet de output als volgt zijn  
X wint van Y met a kaarten en een totale waarde van c tegen b kaarten en een totale waarde van d.

(X is de winnaar, Y de verliezer. a is het aantal kaarten van de winnaar en c de totale waarde van de kaarten van de winnaar. b is het aantal kaarten van de verliezer en d de totale waarde van de kaarten van de verliezer)

Tip: de totale waarde moet altijd 364 zijn.

# Stap 7: uitbreiding

Waardes 1, 11, 12 en 13 worden voortaan benoemd als ‘Aas’, ‘Boer’, ‘Dame’, ‘Heer’ of zoals jij deze kaarten benoemd.

Vanaf nu houden we ook rekening met de verschillende soorten of kleuren (“Schoppen” “Harten”, “Klaveren”, “Koeken”)  
Gebruik daarom een associatieve array bij het aanmaken van het kaartspel en het bijhouden van de kaarten van de speler en de computer.

De verschillende soorten krijgen ook een waarde verhoging als volgt:

* Helaas voor koeken, deze blijft de gewone waarde.
* Alle waardes van klaveren worden vermenigvuldigd met 1.2
* Alle waardes van harten worden vermenigvuldigd met 1.5
* Alle waardes van Schoppen worden vermenigvuldigd met 2.

Als gevolg verliest Koeken koning (waarde 13) van Schoppen 7 (waarde 14)  
Waardes worden naar boven afgerond (klaveren 2 wordt waarde 3 in plaats van 2.2).

Wanneer 2 kaarten gelijk zijn in waarde neemt iedere partij een 2e kaart en een 3e kaart.  
Het zijn de 3e getrokken kaarten die opnieuw vergeleken worden.  
Zijn deze weer gelijk trekken beide partijen opnieuw 2 kaarten en worden de laatste opnieuw vergeleken tot dat er een winnaar is.  
Heeft 1 van de partijen geen kaarten meer, wind de andere partij automatisch alle kaarten.

Het spel blijft duren tot dat de speler q ingeeft of een van de partijen geen kaarten meer heeft.

Tip: print bij iedere beurt de vergelijking uit  
Computer trok Harten 7 met een waarde van 11, Speler trok Koeken 10 met een waarde van 10 – Computer wint.

Tip: maak verschillende helper functies om je code overzichtelijker te houden.  
Tip: de totale waarde moet altijd 528 zijn.

Op het einde van het spel wordt de output gegenereerd zoals in Stap 6.