Programmeren OEFENTENTAMEN2 nakijkvel (29 punten)

(5 punten) Menselijke interpreter

0. (1 punt) c
1. (1 punt) c
2. (1 punt) c
3. (1 punt) a
4. (1 punt) b

(4 punten) Comprehend this

5. (1 punt) b
6. (1 punt) a
7. (1 punt) b
8. (1 punt) c

(2 punten) Woordenboeken

9. (1 punt) a
10. (1 punt) d

(9 punten) Functies

Syntactische fouten slaan we hier grotendeels over. Behalve als je door zo'n fout niet kan beoordelen of de gewensde kennis aanwezig is. Bijvoorbeeld: een missende) gewoon negeren, een loop met compleet andere opbouw ala `for (i = 0 to 9)` is fout.

Als het werkt is het goed! We kijken hier niet naar design ;).

Let op, niet alle onderstaande oplossingen hoef je te kennen. Ze zijn enkel ter referentie.

11. Oplossingen zijn hier bijvoorbeeld:

```
# deze
def filter(words):
    l = []
    for word in words:
        shouldAdd = True
        for letter in word:
            if letter.isupper():
                shouldAdd = False
        if shouldAdd:
            l.append(word)
    return l

# maar ook
def filter(words):
    return [w for w in words if not any([l.isupper() for l in w])]

# of deze
def filter(words):
    return [w for w in words if w.islower()]
```

    (a) (1 punt) Idee is correct (een dubbele loop (of gebruik islower()), een conditie om te checken op hoofdletters, er wordt een woord toegevoegd aan een nieuwe lijst)

    (b) (1 punt) Er wordt een nieuwe lijst aangemaakt, de oude blijft onveranderd.

    (c) (1 punt) Code is correct (runt minus syntactische fouten, en zou de juiste uitkomsten genereren)

12. Oplossingen zijn hier bijvoorbeeld:

```
# deze
def weighted_average(grades, weights):
    out = 0
    for i in range(len(grades)):
        out += grades[i] * weights[i]
    return out

# maar ook
def weighted_average(grades, weights):
    return sum([weights[i] * grades[i] for i in range(len(grades))])

# of deze
def weighted_average(grades, weights):
    return sum(w * g for w, g in zip(weights, grades))
```

   (a) (1 punt) Idee is correct (een loop over indices (of gebruik zip), een variabele om in op te tellen, weight * grade, en return aanwezig)

   (b) (1 punt) Correct gebruik gemaakt van indices, de juiste grades * de juiste weights.

   (c) (1 punt) Code is correct (runt minus syntactische fouten, en zou de juiste uitkomsten genereren)

13. Oplossingen zijn hier bijvoorbeeld:

```
# deze
def flip(dict):
    out = {}
    for k in dict:
        out[dict[k]] = k
    return out

# maar ook
def flip(dict):
    return {dict[k] : k for k in dict}
```

   (a) (1 punt) Idee is correct (een loop over dict, een omwisseling van key en value, een return aanwezig)

   (b) (1 punt) Er wordt een nieuwe dictionary gemaakt, de oude blijft ongewijzigd. Er wordt correct gebruik gemaakt van de dictionary (keys worden ingevuld en niet indices).

   (c) (1 punt) Code is correct (runt minus syntactische fouten, en zou de juiste uitkomsten genereren)

(6 punten) Whoops!

We kijken ook hier niet naar design. Als het probleem wordt opgelost is het goed!

14. Een oplossing is hier bijvoorbeeld:

    ```
    x is een float en deze mag je niet gebruiken als index voor een string.
    Regel 2 veranderen naar x = int(4 / 2) zou het probleem verhelpen.
    ```

    (a) (1 punt) Correct gespotte fout + uitleg
    (b) (1 punt) Correcte oplossing

15. Een oplossing is hier bijvoorbeeld:

    ```
    Door een dictionary heen loopen geeft je telkens de key.
    Dus niet de value die je hier wilt optellen.
    Regel 4 veranderen naar y += d[x] zou het probleem verhelpen.
    ```

    (a) (1 punt) Correct gespotte fout + uitleg
    (b) (1 punt) Correcte oplossing

16. Een oplossing is hier bijvoorbeeld:

    ```
    Uiteindelijk komt er een error welke je verteld dat je geen strings en
    lijsten bij elkaar kan optellen. Op regel 3 wordt per ongeluk een lijst
    ge-returned.
    Regel 3 veranderen naar return "" zou het probleem verhelpen.
    ```

    (a) (1 punt) Correct gespotte fout + uitleg
    (b) (1 punt) Correcte oplossing

(3 punten) Lastig lezen en begrijpen

17. (1 punt) Hier zijn ontzettend veel goede antwoorden mogelijk, een kleine incomplete lijst:
    (a) Haal regel 7 weg, deze is geheel overbodig.
    (b) Gebruik één oplossing voor het probleem, bijvoorbeeld of module en gedeeld door of een while loop.
    (c) Verander regel 2 naar `n = -1`, dan kan je regel 3 schrappen en heb je geen duplicate code.
    (d) Introduceer een loop en een functie voor het tellen van muntjes, scheelt een heel hoop code.

18. (2 punten) Hier zijn ontzettend veel goede antwoorden mogelijk, een kleine lijst:
    (a) Let op spaties rondom operatoren.
    (b) Verzin betere variabele namen dan `foo`, `n` en `c`.
    (c) Gebruik comments.
    (d) Gebruik een witregel om blokken code te scheiden.