Faculteit der Natuurwetenschappen, Wiskunde en Informatica

Tentamen:	Programmeren 1
Code:	NWI-NP033B
Docent:	H. Engelkamp
Telefoonnummer docent:	53367 / 0649848387
Datum:	1 maart 2017
Tijd:	18:00
Tentamenduur:	3 uur
Rekenmachine toegestaan:	Nee
Aantal vragen:	2
Beantwoord in:	Nederlands of Engels
Aantal pagina's:	5

Opgaven inleveren na afloop!

Het tentamen bestaat uit twee opgaven met elk vier onderdelen. Bij elk onderdeel staat aangegeven hoeveel punten je kunt halen. In totaal kun je 100 punten halen.

Schrijf leesbaar! Niet leesbaar = fout.

Start opgave 2 op een nieuw vel. Voorzie alle vellen van je naam en studentnummer.

Je mag bij dit tentamen behalve schrijfgerei geen hulpmiddelen gebruiken.

1. Diverse vragen

Hieronder staan een aantal vragen over algoritmen. Geef voor iedere vraag een beknopt, gemotiveerd antwoord.

1.1 (max. 10 punten)

Bestudeer de volgende functies f1 en f2:

```
def f1(n):
    while False:
        for i in range(n):
            print (i)

def f2(n):
    resultaat = 0
    for i in range(n):
        for j in range(i):
            resultaat = resultaat + i**2 + j
    return resultaat
```

Leg uit wat er wordt bedoeld met "orde van complexiteit", en bepaal deze voor de functies f1 en f2.

1.2 (max. 10 punten)

Bestudeer de functies f3 en f4:

```
def f3(n):
    if n < 10:
        print (n)
    else:
        print (n % 10)
        f3(n // 10)

def f4(n):
    if n < 10:
        print (n)
    else:
        f4(n // 10)
        print (n % 10)
```

Opdracht: Geef voor de onderstaande aanroepen aan wat het resultaat zal zijn:

```
f3(314)
f4(314)
```

1.3 (max. 10 punten)

Beschouw onderstaande regels Python-code (in willekeurige volgorde):

```
1. zeef(100)
2. def zeef(n):
3. num[s] = False
4. print ('De priemgetallen tot en met {0:d} zijn:'.format(n))
5. for p in range(2, n+1):
6. for s in range(p**2, n+1, p):
7. num = [True] * (n+1)
8. print(p, end=' ')
9. if num[p]:
```

Het dient om een aantal priemgetallen te vinden, en geeft het volgende resultaat:

```
De priemgetallen tot en met 100 zijn:
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97
```

Opdracht: Zet de programmaregels in de juiste volgorde, en geef hierbij ook duidelijk aan hoever steeds ingesprongen moet worden zoals in het voorbeeldantwoord hiernaast is weergegeven. Meerdere volgorden zijn goed; kies er slechts één.

Voorbeeldantwoord:

```
1
    2
    3
        4
            5
    6
        7
    8
        9
```

1.4 (max. 10 punten)

Beschouw onderstaande Python code:

```
def bubblesort(lst):
    for i in range(len(lst)-1, 0, -1):
        for j in range(i):
            if lst[j] < lst[j+1]:
                lst[j] = lst[j+1]
                lst[j+1] = lst[j]

lijst = [6, 4, 11, 12, 17, 7, 1, 14, 10, 16, 18, 19, 9, 2, 3, 0, 8, 5,
15, 13]
bubblesort(lijst)
print(lijst)
```

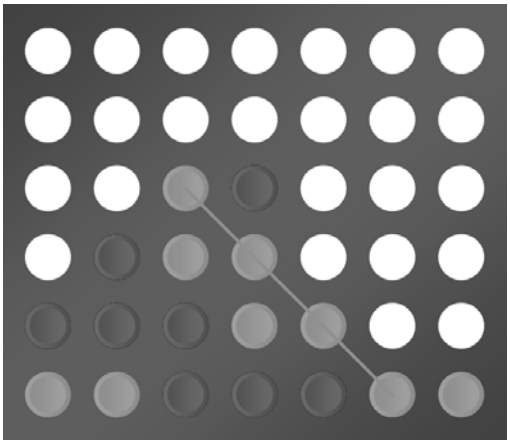
Deze code zou via een zogenaamde *bubble sort* een lijst van laag naar hoog moeten sorteren. Het werkt echter niet correct; er zitten meerdere fouten in.

Opdracht: Identificeer de fouten in dit programma, en geef aan hoe die hersteld kunnen worden.

2. Programmeeropdracht: vier op een rij.

Het spel vier op een rij[1] wordt gespeeld op een verticaal geplaatst bord bestaande uit zeven kolommen en zes rijen. Iedere speler beschikt over 21 schijven met zijn/haar eigen kleur, meestal geel en rood. De spelers laten om de beurt een schijf in één van de nog niet volle kolommen vallen. De schijf bezet altijd het laagst beschikbare vak in een kolom. Een speler wint door met vier van zijn eigen schijven een aaneengesloten rij te vormen. Vaak roept de speler hierbij luid en duidelijk "Vier op een rij!". Een rij kan zowel verticaal, horizontaal als diagonaal worden gevormd en beëindigt het spel. Het spel eindigt in een gelijkspel als geen van de twee spelers erin slaagt een aaneengesloten rij te vormen voordat het bord volledig door schijven is gevuld.

In deze opdracht gebruiken we een 2D lijst om het speelbord te representeren in Python. Gele schijven worden hierin bewaard als een 1 (van speler 1) en rode schijven als een 2. Lege plekken bevatten een 0. We kiezen ervoor om het speelbord te beschouwen als een rij van kolommen. Het hiernaast getoonde spel ziet er dan als volgt uit:



```
bord = [  [1, 2, 0, 0, 0, 0],
          [1, 2, 2, 0, 0, 0],
          [2, 2, 1, 1, 0, 0],
          [2, 1, 1, 2, 0, 0],
          [2, 1, 0, 0, 0, 0],
          [1, 0, 0, 0, 0, 0],
          [1, 0, 0, 0, 0, 0]]
```

De vierde steen van onder in de derde kolom wordt gerepresenteerd door `bord[2][3]`, het vakje rechtsonder is `bord[6][0]`.

2.1 Bord tonen (max. 10 punten)

Schrijf de functie `print_spel(bord)` die het speelbord afdrukt zoals hiernaast weergegeven: Speler 1 met een 'x', speler 2 met een 'o' en een lege positie met '.'. Je mag ervan uitgaan dat er geen ongeldige waarden in de lijst `bord` voorkomen als deze functie wordt aangeroepen, en dat het formaat correct is.

```
>>> print_spel(bord)
.......
.......
..xo...
.oxx...
oooxx..
xxoooxx
```

[1] Bron: wikipedia

2.2 Een zet doen (max. 15 punten)

Schrijf de functie `zet(bord, speler, kolom)` die de lijst `bord` aanpast door een 1 of een 2 (afhankelijk van `speler`) te plaatsen op de plek die bepaald wordt door `kolom`, en die `True` teruggeeft als dat een geldige zet is. De winnende zet in het weergegeven speelbord was dan bijvoorbeeld `zet(bord, 1, 2)`: plaats een gele schijf in de derde kolom (zie ook hiernaast). Je mag ervan uitgaan dat `bord` een geldige lijst is (7 x 6 elementen die alleen uit 0, 1 of 2 kunnen bestaan) en dat `speler` en `kolom` correcte waarden hebben; daar hoeft niet op gecontroleerd te worden. Er moet echter wel getest worden of er in de betreffende kolom nog plaats is – zo niet, dan moet de lijst `bord` onveranderd blijven en geeft de functie `False` terug.

```
>>> print_spel(bord)
.......
.......
...o...
.oxx...
oooxx..
xxoooxx

>>> zet(bord, 1, 2)
True

>>> print_spel(bord)
.......
.......
..xo...
.oxx...
oooxx..
xxoooxx
```

2.3 Winnaar bepalen (max. 15 punten)

Schrijf de functie `winnaar(bord)` die bepaalt of er een winnaar is. Deze functie moet 1 of 2 teruggeven als speler 1 of 2 vier stenen op een rij heeft (horizontaal, verticaal of diagonaal). Is er (nog) geen winnaar dan moet de functie 0 teruggeven. Ga ervan uit dat het een werkelijke spelsituatie is; het kan dus niet voorkomen dat beide spelers vier op een rij hebben. Ook hier hoeft niet gecontroleerd te worden of `bord` een correcte lijst is.

Tip: in Python kun je meerdere elementen tegelijk vergelijken: de operatie `bord[k][r] == bord[k][r+1] == bord[k][r+2] == bord[k][r+3]` levert alleen `True` op als deze vier elementen identiek zijn.

2.4 Compleet spel programmeren (max. 20 punten)

Schrijf een programmaatje, gebruikmakend van de functies `print_spel`, `zet` en `winnaar` hierboven, om vier op een rij te kunnen spelen. Om en om moeten de spelers een kolomnummer intypen. Als een speler een kolom typt die al vol zit of die niet geldig is (kleiner dan 0 of groter dan 6), moet hij/zij alsnog een correcte kolom typen. Na elke zet moet het aangepaste speelbord worden getoond. Het spelletje eindigt als er een winnaar is, of met gelijkspel als alle kolommen vol zijn.

Tip: Een leeg bord kun je in Python compact definiëren met de volgende regel:

```
bord = [[0 for r in range (6)] for k in range (7)]
```

```python
## 1.1
def f1(n):
    while False:
        for i in range(n):
            print (i)
# Orde: O(1) want tijd hangt NIET van n af (vanwege de False)


def f2(n):
    resultaat = 0
    for i in range(n):
        for j in range(i):
            resultaat = resultaat + i**2 - j
    return resultaat
# Orde: O(n^2) want zowel de i als de j loop schalen lineair met n.
# De berekening wordt n*(n-1)/2 keer uitgevoerd
# n=10: 45
# n=100: 4950
# 10 -> 100: 100 x vaker uitgevoerd. n^2







##
def f3(n):
    if n<10:
        print (n)
    else:
        print (n % 10)
        f3(n // 10)

f3(314)
if 314<10:
    print (314)
else:
    print (314 % 10)            #4
    #f3(314 // 10)
    if 31<10:
        print (31)
    else:
        print (31 % 10)         #1
        #f3(31 // 10)
        if 3<10:
            print (3)           #3
        else:
            #f3(3 // 10)
            print (3 % 10)




#4
#1
#3

def f4(n):
    if n<10:
        print (n)
    else:
        f4(n // 10)
        print (n % 10)
#3
#1
#4


f4(314)
if 314<10:
    print (314)
else:
```

```
    #f4(314 // 10)
    if 31<10:
        print (31)
    else:
        #f4(31 // 10)
        if 3<10:
            print (3)           #3
        else:
            #f4(3 // 10)
            print (3 % 10)
        print (31 % 10)         #1
    print (314 % 10)            #4
```

```
## Zeef
def zeef(n):                                                    #2
    print ('De priemgetallen tot en met {0:d} zijn:'.format(n))     #4
    nummers=[True]*(n+1)                                        #7
    for p in range(2,n+1):                                      #5
        if nummers[p]:                                              #9
            print(p, end=' ')                                           #8
            for s in range(p**2,n+1,p):                                 #6
                nummers[s]=False                                            #3
zeef(100)                                                       #1
```

```
##
def zeef(n):
    num=[True]*(n+1)
    print ('De priemgetallen tot en met {0:d} zijn:'.format(n))
    for p in range(2,n+1):
        if num[p]:
            for s in range(p**2,n+1,p):
                num[s]=False
            print(p, end=' ')
zeef(100)
```

```
##
def bubblesort(lst):
    for i in range(len(lst)-1, 0, -1):
        for j in range(i):
            if lst[j] < lst[j+1]:
                lst[j] = lst[j+1]
                lst[j+1] = lst[j]

lijst = [6, 4, 11, 12, 17, 7, 1, 14, 10, 16, 18, 19, 9, 2, 3, 0, 8, 5, 15, 13]
bubblesort(lijst)
print(lijst)
```

```
##
def bubblesort(lst):
    for i in range(len(lst)-1, 0, -1):
        for j in range(i):
            # if lst[j] < lst[j+1]:
            if lst[j] > lst[j+1]:
                # lst[j] = lst[j+1]
                # lst[j+1] = lst[j]
                lst[j], lst[j+1] = lst[j+1], lst[j]

lijst=[6, 4, 11, 12, 17, 7, 1, 14, 10, 16, 18, 19, 9, 2, 3, 0, 8, 5, 15, 13]
bubblesort(lijst)
print(lijst)
```

```
## 4 op een rij
```

```python
#2.1
def print_spel(bord):
    for r in range(5,-1, -1): #hoogste nummers eerst
        for k in range(7):
            if bord[k][r]==1:
                print ('x', end='')
            elif bord[k][r]==2:
                print ('o', end='')
            else:
                print ('.', end='')
        print () #nieuwe regel
    print('0123456') # service

#2.1 compacter
def print_spel(bord):
    tekens=[".", "x", "o"]
    for r in range(5,-1, -1): #hoogste nummers eerst
        for k in range(7):
            print (tekens[bord[k][r]], end='')
        print () #nieuwe regel
    print('0123456') # service

#2.2
def zet(bord, speler, kolom):
    for rij in range(6):
        if bord[kolom][rij] == 0:
            bord[kolom][rij] = speler
            return True
    return False

#2.3
def winnaar(bord):
    # vertikaal
    for r in range(3):
        for k in range(7):
            if bord[k][r]==bord[k][r+1]==bord[k][r+2]==bord[k][r+3]>0:
                return bord[k][r]
    # horizontaal
    for r in range(6):
        for k in range(4):
            if bord[k][r]==bord[k+1][r]==bord[k+2][r]==bord[k+3][r]>0:
                return bord[k][r]
    # diagonalen
    for r in range(3):
        for k in range(4):
            if bord[k][r]==bord[k+1][r+1]==bord[k+2][r+2]==bord[k+3][r+3]>0:
                return bord[k][r]
            if bord[k][r+3]==bord[k+1][r+2]==bord[k+2][r+1]==bord[k+3][r]>0:
                return bord[k][r+3]
    return 0


## 2.4 Spel!

bord=[[0 for r in range (6)] for k in range (7)] # leeg bord
print_spel(bord)
for zetnummer in range(42):
    speler = zetnummer % 2 + 1
    while True: # loop totdat juiste kolom is ingevoerd
        kolom=int(input('Speler {0:d}, uw zet graag:'.format(speler)))
        if kolom<0 or kolom>6:
            print ('Ongeldige kolom. Opnieuw graag')
        elif not zet(bord, speler, kolom):
            print('Kolom', kolom, 'zat al vol. Opnieuw graag')
        else:
            break
    print_spel(bord)
    if winnaar(bord)>0:
        break
if winnaar(bord)>0:
    print ('Gefeliciteerd, speler', winnaar(bord))
```

```
214   else:
215       print ('Gelijkspel!')
216
```