Tentamen Programmeermethoden NA
Vrijdag 8 december 2017, 14.00 - 17.00 uur
Universiteit Leiden — Informatica

Het tentamen bestaat uit 4 opgaven verdeeld over 3 pagina's. De duur van het tentamen is 3 uur. De te behalen punten (totaal 100) staan tussen haakjes bij de opgaven.

Bij alle functies moeten de variabelen (globale constanten eventueel uitgezonderd) als parameter of lokale variabele voorkomen. Gebruik `____` om één niveau van inspringen aan te duiden. Twee keer dit karakter betekent dus twee keer inspringen. Denk aan de dubbele punten!

Veel succes!

---

1. (35 punten) We hebben een lijst `P` met `len(P)` (gegarandeerd > 0) elementen. Gegeven zijn functies `isgetal(element)` en `iskar(element)` die `True` opleveren wanneer `element` een getal respectievelijk een karakter is, anders `False`. Een voorbeeld van een dergelijke lijst is `P = [14, 55, 'a', 'q', 39, 'f', 'e', 23, 78, 'x']`.

a. (5) Schrijf een Python-functie `telparen(P)` die het aantal mogelijk te maken paren van een getal en karakter in `P` oplevert door elk element ten hoogste éénmaal te gebruiken. Voor het voorbeeld is het resultaat 5. Voor een lijst `[14, 55, 'a']` is het resultaat 1 (er blijft een getal over waarmee geen paar kan worden gemaakt).

b. (5) Schrijf een Python-functie `zoekelement(P, getalkar)` die de lijst-index teruggeeft van het laatste element in de lijst dat een getal (indien `getalkar==True`) of een karakter is (indien `getalkar==False`). Er moet dus vanaf het einde van de lijst worden gezocht. Indien zo'n element niet voorkomt, retourneer `0`.

c. (8) Neem aan dat `P` evenveel getallen als karakters bevat. Schrijf een Python-functie `maakparen(P)` die de elementen in `P` zodanig verwisselt dat er in `P` achtereenvolgens paren van een getal gevolgd door een karakter komen te staan. Gebruik de functie van **b**. Voor het voorbeeld heeft `P` na het uitvoeren van deze functie de volgende inhoud: `[14, 'x', 55, 'q', 39, 'f', 78, 'a', 23, 'e']`.

d. (3) Geef een Python-functie `wisselparen(P, i, j)` die de paren op posities `i` en `j` omwisselt. `i` en `j` wijzen naar de eerste index van een paar (dus de index waar het getal van een paar staat).

e. (6) Schrijf een Python-functie `sorteerparen(P)` die de paren getal, karakter in `P` oplopend sorteert op basis van de getalwaarde. Wanneer toegepast op het resultaat van **c** heeft `P` na het sorteren de volgende inhoud: `[14, 'x', 23, 'e', 39, 'f', 55, 'q', 78, 'a']`.

f. (8) Gegeven een lijst `L` met `len(L)` > 0 bestaande uit gehele getallen. Schrijf een Python-functie `deelrij(L)` die de lengte bepaalt van een langste deelrij in `L` bestaande uit alleen positieve getallen (≥ 0) en tevens het kleinste getal in die deelrij bepaalt. De functie wordt aangeroepen als volgt: `lengte, kleinste = deelrij(L)` en geeft dus als returnwaarden de lengte van de gevonden deelrij en het kleinste getal in die deelrij. Wanneer er geen deelrij wordt gevonden, geef dan `0,0` als returnwaarden. Zijn er meerdere deelrijen met dezelfde lengte, geef dan de kleinste van de eerste deelrij. Voor de lijst `8 -1 -2 3 -6 7 3 1 8 -4` zouden de returnwaarden `4,1` zijn, namelijk de lengte van de deelrij `7 3 1 8` en kleinste `1`.

2. (20 punten)

a. (4) Bij een functie kun je te maken hebben met *lokale* en *globale* variabelen. Verder onderscheiden we *formele* en *actuele* parameters. Leg deze vier begrippen duidelijk uit.

b. (6) Gegeven een Python-programma met daarin de volgende twee functies:

```
def peppa(a, b):
    q, som, t, a = a, 0, 1, 1
    while a <= b:
        som += t
        t = 2*a + 1
        a += 1
        print "Q", a, t, som
    return q + som
```

```
def george(a, b):
    x = a - 1
    a -= 1
    b += 1
    t = peppa(b, peppa(x, a + 2))
    return t * (b - b + 1)
```

Verder zijn de globale variabelen `x`, `y` en `som` gegeven (allen van type `int`). Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit):

```
x, y, som = 1, 4, 0
som = george(x, y)
print x, y, som
```

c. (5) Geef een functie `konijn(a,b)` die dezelfde returnwaarde oplevert als `george(a, b)` voor alle mogelijke integer-parameters `a`, `b` ≥ 0, maar die uit slechts één `return`-statement bestaat, en waarin `peppa` niet meer wordt aangeroepen.

d. (5) Stel we plaatsen bovenaan `peppa` het statement `global som` en we voeren het stukje programma onder **b** opnieuw uit. Is de uitvoer nu anders? En hoe zit dat wanneer we na het initialiseren van de globale variabelen de regels `som = george(x, y); print x, y, som` meerdere keren uitvoeren?

3. (10 punten) Gegeven een NumPy array `M` zoals hiernaast weergegeven. Schrijf voor de hieronder gegeven slices van `M` de juiste expressie die resulteert in die slice.

```
array([[92, 66, 89, 42, 25, 45],
       [62, 83, 71, 47, 90, 30],
       [54, 84, 16, 10, 62,  8],
       [17, 28,  8, 31, 50, 34],
       [95, 41, 95, 44, 78, 17],
       [86, 88, 21, 54, 39, 24]])
```

a. (2) `array([95, 41, 95, 44, 78, 17])`

b. (2)
```
array([[92, 66, 89, 42, 25, 45],
       [54, 84, 16, 10, 62,  8],
       [95, 41, 95, 44, 78, 17]])
```

c. (2)
```
array([[42, 25],
       [47, 90],
       [10, 62],
       [31, 50],
       [44, 78],
       [54, 39]])
```

d. (2)
```
array([[92, 45],
       [86, 24]])
```

e. (2) Wanneer we met `np.mean` het gemiddelde willen bepalen van elke **rij**, langs welke as (*axis*) moet de operatie dan worden uitgevoerd? De rij-as (`0`) of de kolom-as (`1`)?

4. (35 punten) We hebben een `n` bij `n` (> 0) NumPy array `F` met gehele getallen in het interval $[1, \mathtt{k}]$. Dit stelt een speelveld voor waarbij het de bedoeling is dat in ieder vakje hetzelfde getal komt te staan. Vanuit de linkerbovenhoek (element $0, 0$) wordt een aaneengesloten veld gevormd bestaande uit vakjes met hetzelfde getal. Vakjes kunnen alleen in de horizontale en verticale richting worden verbonden (dus *niet* diagonaal). Wanneer het getal in de linkerbovenhoek wordt veranderd, veranderen alle aaneengesloten vakjes mee. Door achtereenvolgens andere getallen te kiezen wordt het aaneengesloten veld vergroot totdat het even groot is als het gehele speelveld.

Hiernaast staat een voorbeeld met $\mathtt{n} = 6$ en $\mathtt{k} = 5$. De vakjes die behoren tot het huidige aaneengesloten veld zijn gemarkeerd. De variabelen `n` en `k` zijn als globale "constanten" gedefinieerd en hoeven niet doorgegeven te worden als parameters.

4*	4*	4*	2	3	1
4*	2	4*	3	4	1
4*	5	2	4	3	4
1	4	1	4	3	1
4	2	1	3	3	2
4	2	3	1	1	2

a. (5) Schrijf een Python-functie `opgelost(F)` die `True` oplevert wanneer het spel is opgelost, dus wanneer in alle vakjes hetzelfde getal staat.

Om te bepalen welke vakjes behoren tot het aaneengesloten veld maken we gebruik van een array `mask` met dezelfde dimensies als `F` waarbij elk element een Boolean-waarde is. Een element in `mask` is `True` wanneer het corresponderende element in `F` behoort tot het aaneengesloten veld. `F[0,0]` hoort altijd tot het aaneengesloten veld en `mask[0,0]` heeft daarom altijd de waarde `True`.

b. (4) Schrijf een Python-functie `buurinmask(mask,i,j)` die `True` teruggeeft wanneer een gegeven element `i,j` een horizontale of verticale buur heeft waarvoor de mask-waarde `True` is. Geef anders de returnwaarde `False`. Bijvoorbeeld voor een `mask` waarbij alleen `mask[0,0]` de waarde `True` heeft: `i=1,j=0` geeft `True` en `i=2,j=0` geeft `False`.

c. (10) Schrijf een Python-functie `gebied(F)` die het huidige aaneengesloten veld in `F` bepaalt door een `mask` array op te bouwen en deze array als returnwaarde geeft. Initialiseer een nieuw array ter grootte van `F` op `False` en zet `mask[0,0]` op `True`. Bezoek alle vakjes en ga voor een vakje na of het een buur heeft waarvoor de mask-waarde `True` is (gebruik de functie van **b**) én het vakje hetzelfde getal bevat als de linkerbovenhoek. Herhaal het bezoeken van de vakjes totdat er geen veranderingen meer worden gemaakt. Voor het voorbeeld moeten de volgende elementen in `mask` op `True` komen te staan: `mask[0,0]`, `mask[0,1]`, `mask[0,2]`, `mask[1,0]`, `mask[1,2]` en `mask[2,0]`.

d. (10) We gaan nu proberen om gegeven `F` en `mask` een goed getal te kiezen dat we kunnen invullen in de linkerbovenhoek. We bepalen het getal waardoor in de nieuwe situatie het grootst mogelijke aaneengesloten veld zal ontstaan. Schrijf hiertoe een functie `kiesgetal(F,mask)` die als volgt te werk gaat: bezoek alle vakjes die niet tot het huidige aaneengesloten veld behoren, maar één van hun buren wel (gebruik weer **b**). Houd van de vakjes waarvoor dit geldt bij hoe vaak elk getal voorkomt: gebruik een hulp-array. Bepaal nadat alle vakjes zijn bezocht het kleinste getal dat het vaakst voorkomt en retourneer deze waarde. Voor het voorbeeld met correct bepaalde mask: 2 wordt gekozen als beste getal, het werd 3 keer geteld.

e. (6) Tenslotte schrijven we een Python-functie `oplossen(F)` welke het spel oplost door gebruik te maken van de hierboven geschreven functies. Bepaal telkens het getal om in te vullen en geef alle elementen van het aaneengesloten veld de waarde van dit getal, totdat het spel is opgelost. Retourneer het aantal stappen dat nodig was om het spel op te lossen.