Hertentamen Programmeermethoden NA
Vrijdag 27 januari 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 (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 `B` met `len(B)` (gegarandeerd > 0) gehele getallen. De lijst stelt een binair getal voor wanneer alle elementen van de lijst of 1 of 0 zijn.

a. (4) Schrijf een Python-functie `is_binair(B)` die precies `True` teruggeeft als alle elementen van de lijst `B` of 1 of 0 zijn. Stop zodra het antwoord bekend is.

b. (6) Schrijf een Python-functie `decimaal(B)` die het decimale getal teruggeeft dat door de lijst `B` wordt gerepresenteerd. Bijvoorbeeld voor de lijst `[0, 1, 0, 1, 1, 0]` is de returnwaarde `22` via $0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 0 \cdot 2^0$.

c. (8) Schrijf een Python-functie `optellen(B, C)` die twee soortgelijke lijsten `B` en `C` (allebei "binair", met dezelfde lengte en het eerste element is gegarandeerd 0) optelt en het resultaat opslaat in een nieuwe lijst. Deze nieuwe lijst is de returnwaarde van de functie. Pak het optellen als volgt aan: begin bij het meest rechtse (en dus laatste) element en werk naar voren. $1 + 1 = 10$ in binair, dus een 0 opschrijven en 1 onthouden, $1 + 1 + 1 = 11$.

d. (4) Schrijf een Python-functie `controle(B, C)` waarmee we onze functie `optellen` kunnen testen. Zet `B` en `C` om naar een decimaal getal en tel deze bij elkaar op. Vergelijk dit met het resultaat van de aanroep `optellen(B, C)` omgezet naar decimaal. Als de resultaten overeenkomen, return `True`, anders `False`. Gebruik de functies geschreven bij **b** en **c**.

e. (5) Gegeven een lijst `G` bestaande uit gehele getallen. Geef een eenvoudige Python-functie `sorteren(G)` die `G` *aflopend* sorteert.

f. (8) Schrijf een Python-functie `veelvouden(A)` die de lengte uitrekent van een langste aaneengesloten deelrij van veelvouden van 3 in de lijst `A`. De functie geeft als returnwaarde zowel de lengte van de gevonden deelrij als het grootste veelvoud van 3 dat in de gehele lijst voorkomt. De functie wordt aangeroepen als volgt: `lengte, grootste = veelvouden(A)` en voor de lijst `1 1 3 9 27 6 2 9 33 7` zou de returnwaarde `(4, 33)` zijn, namelijk de lengte van de deelrij `3 9 27 6` en het grootste veelvoud `33`. (Tip: denk aan de modulo operator!)

2. (20 punten)

a. (4) Bij een functie kun je te maken hebben met *locale* 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 woezel(p, q):
    q -= 3
    bla = p
    tel = q + 1
    while tel >= 0:
        q += (q + 1)
        bla *= p
        print "W", tel, q, bla
        tel -= 1
    return bla
```

```
def pip(a, b):
    p = 2 * a
    a = b + b
    c = b - 2
    res = woezel(p/2, a-5) + woezel(c+2, 3)
    return res * (0*b+1)
```

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

```
p, c = 3, 4
print pip(p, c + 1), p, "woef", c
```

c. (5) Geef een functie `molletje(a,b)` die dezelfde returnwaarde oplevert als `pip` voor alle mogelijke integer-parameters `a`, `b` met `b` ≥ 3, maar die uit slechts één `return`-statement bestaat, en waarin `woezel` niet meer wordt aangeroepen.

d. (5) Stel nu we plaatsen direct onder de regel `def pip(a, b):` steeds één van de volgende regels:

1. `global a, b`
2. `global p, b`
3. `global p, c`
4. `global l`

Geef voor elke regel aan of het programma dan wel of niet valide is (en dus door de interpreter wordt geaccepteerd) en waarom wel of niet.

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

```
array([[10, 11, 12, 12, 11, 10],
       [13, 14, 15, 15, 14, 13],
       [16, 17, 18, 18, 17, 16]])
```

a. (2)
```
array([[10, 11, 12, 12, 11, 10],
       [13, 14, 15, 15, 14, 13],
       [16, 17, 18, 18, 17, 16]])
```

b. (2)
```
array([13, 15])
```

c. (2)
```
array([[10, 12],
       [13, 15],
       [16, 18]])
```

d. (2)
```
array([[11, 11],
       [17, 17]])
```

e. (2) Geef een expressie die de som bepaalt van de *eerste* 3 kolommen van `X`. Resultaat: `array([39, 42, 45])`.

4. (35 punten) Het spel *Mijnenveger* wordt gespeeld op een rechthoekig speelveld opgedeeld in vakjes. Onder elk vakje is wel of niet een mijn geplaatst. De inhoud van de vakjes is niet zichtbaar voor de speler. De speler heeft telkens de keuze om een vakje te openen zodat de inhoud zichtbaar wordt (de speler verwacht geen mijn) of om een vakje van een vlag te voorzien (de speler verwacht *wel* een mijn). Het doel van het spel is om alle vakjes met mijnen van een vlag te voorzien en *niet* te openen.
Voor het spel gebruiken we twee `m` bij `n` (beide $>$ `0`) NumPy arrays: `M` en `T`. In het array `M` is het mijnenveld opgeslagen met gehele getallen ≥ -1. Het getal -1 duidt een mijn aan, de getallen $\geq$ `0` geven aan onder hoeveel buren van dat vakje (horizontaal, verticaal en diagonaal, dus maximaal 8 in totaal) een mijn is verstopt ("hints"). Het array `T` slaat de "toestand" van elk vakje op met de gehele getallen `0`, `1` en `2`: `0` betekent dat het vakje nog niet is "geopend", `1` is een "geopend" vakje en `2` betekent dat het vakje is voorzien van een vlag. (De speler kan alleen maar de waarde van `M` (een mijn of hint) zien van *geopende* vakjes.) Hieronder staan voorbeelden met `m` $= 5$ en `n` $= 8$, waarbij de hints in `M` nog niet zijn berekend:

```
      0  0  0  0  0  0  0  0          0  0  0  0  0  0  0  0
      0 -1  0  0  0 -1  0  0          0  0  0  0  0  0  0  0
M:    0  0 -1  0  0 -1  0  0    T:    0  0  0  1  1  0  1  1
      0  0  0  0  0  0  0 -1          0  0  1  1  1  0  1  2
     -1  0  0  0  0  0  0  0          0  1  1  1  1  0  1  1
```

De variabelen `m` en `n` zijn als globale "constanten" gedefinieerd en hoeven bij deze opgave niet doorgegeven te worden als parameter.

a. (5) Schrijf een Python-functie `mijnen(M)` die het aantal mijnen in de array `M` teruggeeft.

b. (10) Schrijf een Python-functie `buren(A, i, j, waarde)` die binnen array `A` voor vakje $(\texttt{i}, \texttt{j})$ telt hoeveel buren een waarde hebben die gelijk is aan de parameter `waarde`. Als returnwaarde moet worden gegeven het aantal getelde buren en de rij en kolom van een buur (welke maakt niet uit) die die waarde heeft. Als er geen buur wordt gevonden, geef dan rij `-1` en kolom `-1`. Voorbeeld: voor de aanroep `hoeveel, k, l = buren(M, 2, 6, -1)` is een correct resultaat `(3, 3, 7)`.

c. (7) Voor alle vakjes van `M` met de waarde `0` gaan we nu de hints berekenen. Dit wordt gedaan door voor elk vakje het aantal buren te bepalen dat een mijn bevat. De berekende getallen worden weer opgeslagen in het array `M`, het resultaat is hier rechts te zien. Schrijf een Python-functie `hints(M)` die dit doet. Gebruik de functie geschreven onder **b**.

```
 1  1  1  0  0  1  1  0
 1 -1  2  1  2 -1  2  0
 1  2 -1  1  2 -1  3  1
 1  2  1  1  1  1  2 -1
-1  1  0  0  0  0  1  1
```

d. (5) Schrijf een Python-functie `opgelost(M, T)` die `True` teruggeeft wanneer het spel is opgelost en in alle andere gevallen `False`. Het spel is opgelost wanneer alle mijnen een vlag hebben en er geen enkele vlag op een niet-mijn is geplaatst.

e. (8) We gaan nu proberen om met een eenvoudige methode mijnen te vinden. Schrijf hiertoe een functie `vlagbaar(M, T)` die op zoek gaat naar een al *geopend* vakje waarvan het aantal buren dat nog niet is geopend gelijk is aan het aantal te verwachten mijnen minus het aantal buren dat al is voorzien van een vlag. Retourneer de rij en kolom van een vakje (dus één van de buren) dat kan worden voorzien van een vlag (indien niet gevonden: -1 en -1). Gebruik wederom de functie geschreven onder **b**. In het voorbeeld kan vakje $(2, 2)$ volgens deze regels van een vlag worden voorzien.