NAAM (in drukletters)	VOORNAAM (in drukletters)	Examennummer

Examenvoorbeeld Beginselen van Programmeren
Deel 1
KU Leuven

Richtlijnen

- Schrijf je voornaam, naam, richting, examennummer op **elk** blad dat je afgeeft (ook de opgave). Nummer je bladen.
- Schrijf enkel op **1 zijde** van een blad !
- Ook je **kladbladeren afgeven** ! Vermeld duidelijk dat het kladbladeren zijn !
- Gebruik enkel **papier** dat je van de surveillant krijgt. Geen eigen papier dus !
- Schrijf duidelijk **leesbaar** !
- Zorg voor een duidelijk, **gestructureerd** antwoord !

SPECIFIEK VOOR DEEL 1:

- Dit deel van het examen is "**gesloten boek**".
- Tijdsduur: je krijgt **4 uur** voor het volledige examen. Voor Deel 1 krijg je **maximum 1 uur**.
- De antwoorden op de vragen van Deel 1 schrijf je op de opgave-bladen zelf.

Veel succes !

Marie-Francine Moens
Tom Holvoet

Vraag 1: Pascal-driehoek

De Pascal-driehoek vormt een toren van getallen waarvan het bovenste gedeelte hieronder wordt getoond. De driehoek kan tot in het oneindige verder uitgebreid worden.

```
          1
        1   1
      1   2   1
    1   3   3   1
  1   4   6   4   1
1   5  10  10   5   1
...
```

De rijen van de Pascal-driehoek zijn genummerd 0, 1, 2, 3, ...

1a) Schrijf een recursieve functie pascalRecur(n) die de nde rij van de Pascal-driehoek teruggeeft.

1b) Bereken de tijdscomplexiteit van de functie pascalRecur(n) in termen van het aantal bezoeken aan een lijst.

1a)

```
def pascalRecur(n):
    if n == 0 :
        return [1]
    else :
        line = [1]
        previous_line = pascalRecur(n - 1)
        for k in range(len(previous_line) - 1) :
            line.append(previous_line[k] + previous_line[k + 1])
        line = line + [1]
    return line
```

1b)

A-pascalRecur(n) = 2 + A-pascalRecur(n - 1) + 2(n - 1)
A-pascalRecur(n) = 2 + 2(n - 1) + [2+ 2(n - 2) + A-pascalRecur(n - 2)]
= 2 +2(n - 1) + 2 + 2(n - 2) + [2 + 2(n - 3) + A-pascalRecur(n - 3)]
= ...

$$= \sum_{i=1}^{n} [2 + 2(n - i)] + 1$$

A-pascalRecur(0) = 1

=> A-pascalRecur(n): $O(n^2)$

Vraag 2: Het herschikken van waarden in een lijst

Veronderstel dat x een lijst van integers is met lengte n, en dat a = x[0] het eerste element van x is.

2a) Schrijf een functie die de waarden in x herschikt zodat in de herschikte array:
- a op positie k staat;
- de getallen in de cellen x[0] tot x[k-1] groter dan of gelijk aan a zijn;
- de getallen in de cellen x[k+1] tot x[n-1] kleiner dan a zijn;
- de indexpositie van k teruggeeft.

De functie moet deze taak realiseren zonder een bijkomende lijst te definiëren.

2b) Bewijs de correctheid van de belangrijkste lus in de functie.

2a)

```
def partition (x) :


    a = x[0]

    k = 0
    n = len(x)
    l = n - 1

    while  k < l   :

        while  k < l and x[l] < a :
                l = l - 1
        if k < l :
                x[k] = x[l]
        while k < l and x[k] >= a :
                k = k + 1
        if k < l    :
                x[l] = x[k]


    x[k] = a
    return k
```

2b)
PRE: x is een lijst met elementen die kunnen worden vergeleken met relationele operatoren AND (len(x) > 0)

Lusinvariant: k <= l AND (x[0 .. k - 1] >= a) AND (x[l +1 .. n - 1] < a)

De lusinvariant geldt voor de uitvoering van de belangrijkste (buitenste) while-lus (initialisatie), na het uitvoeren van elke iteratie van deze lus (instandhouding) en na het beëindigen van deze lus (beëindiging).

Vooruitgang: in elke iteratie verhoogt k met minimum 1, l verlaagt met minimum 1
Eindigheid: op een bepaald moment is k == l: lus stopt.

Na het beëindigen van de functie geldt:
POST: (x[0 .. k - 1] >= a) AND (x[k] == a) AND (x[k+1 .. n - 1] < a)

Vraag 3: Referentiesemantiek

Geef de uitvoer van volgend programmaatje:

```
gevonden = True
voorbij = False
gevonden = voorbij
voorbij = True
lijst = [0.3, 0.5, 1.3, 7.8, 9]
lijst2 = [9, 8.7, 3.34, 3.2]
lijst = lijst2
lijst2[1] = 1.56
print ("%.2f %.2f" % (lijst[1],lijst2[0]))
print(gevonden, voorbij)
```

Uitvoer:
1.56 9.00
(False, True)