KU LEUVEN

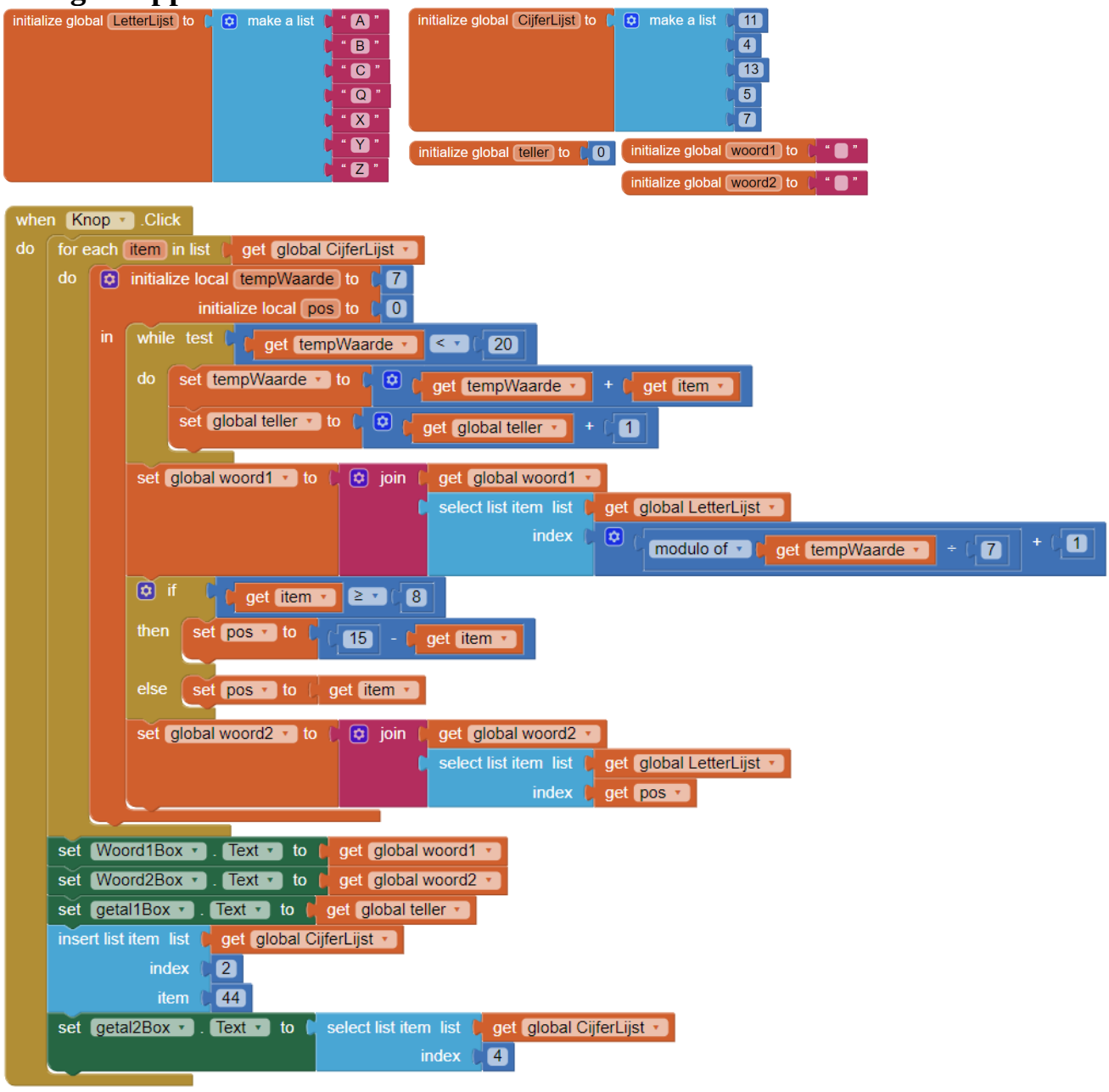
Examen Computationeel Denken
Vragenbladen

Richtlijnen:

1. Deze vragenbundel mag niet uit elkaar gehaald worden.
2. Je voorlopige antwoorden mag je invullen in de vakjes op deze vragenbladen.
3. Leg de antwoorden niet zichtbaar voor medestudenten.
4. Dien op het einde van het examen zowel het vragenblad als het antwoordenblad terug in.

DEEL A: GESLOTEN VRAGEN – antwoordenblad 1

Vraag 1: AppInventor



1.1 Wat wordt er in Woord1Box geplaatst?
1.2 Wat wordt er in Woord2Box geplaatst?
1.3 Wat wordt er in getal1Box geplaatst?
1.4 Wat wordt er in getal2Box geplaatst?

1.1	
1.2	
1.3	
1.4	

Vraag 2: Programmeren in Python A

```
1  lijstA = [1,3,5,6]
2  lijstB = [True,False,True,False]
3  count = 0
4  for i in range(len(lijstA)):
5      c = lijstA[i]
6      for j in range(c):
7          p = j%len(lijstB)
8          lijstB[p] = not lijstB[p]
9          print(lijstB)
10         count = count + 1
11 print(count)
12 print(lijstB)
```

2.1 Wat is de waarde die geprint wordt op regel 11?
2.2 Welke waarden worden er geprint op regel 12?

2.1	
2.2	

Vraag 3: Programmeren in Python B

```
1  lijstX = [3.5, 7.8, 6.9, 2.1, 0.7, 9.9]
2  result =[0,0,0]
3  for e in lijstX:
4      e = int(e * 10)
5      if e<10:
6          result[0]+=e
7      elif e<50:
8          result[1]-=1
9      else:
10         result[2] = e/2
11 t = result[1]
12 result[1] = result[0]
13 result[0] = t
14 print(result[0])
15 print(result[1])
16 print(result[2])
```

3.1 Wat is de waarde die wordt geprint op regel 14?
3.2 Wat is de waarde die wordt geprint op regel 15?
3.3 Wat is de waarde die wordt geprint op regel 16?

3.1	
3.2	
3.3	

DEEL B: VARIANT BEELDVERWERKING – antwoordenblad 2

Implementeer een algoritme dat op basis van een oorspronkelijke figuur en een patroon bepaalt of het patroon voorkomt in de figuur. Het algoritme dat wordt uitgevoerd heeft dus een waarde *True* of *False* terug.

Gegeven bovenstaande figuur. Het linkse patroon komt niet voor in bovenstaande figuur. Het programma zal dus *False* teruggeven. Het rechtse patroon komt wel voor en zal dus *True* teruggeven.

Om na te gaan of een patroon in een figuur voorkomt wordt de kleur van elk pixel (i.e. elke positie) in de oorspronkelijke figuur vergeleken met de kleurwaarde van de pixel in de linkerbovenhoek van het patroon. Indien deze waarden overeenkomen, dan wordt drie keer een hulpmethode `matchingPatternTest` opgeroepen. Telkens wordt hierbij voor een bepaalde kleurwaarde (`rood`, `groen` en `blauw`) nagegaan of de waarden van alle pixels in de oorspronkelijke figuur vanaf die bepaalde positie overeenkomen met het patroon. Indien dat zo is, dan geeft de methode `True` terug, anders `False`. Indien de drie testen slagen, dan komt het patroon immers voor in de oorspronkelijke figuur.

Vul de aan te vullen code in, en geef dit weer op deel B van het antwoordenblad.

```
# importeer libraries
from PIL import Image
import math
import numpy as np

def matchingPatternTest(waardenFiguur, waardenPatroon, i, j):

    #CODE - DEEL 2 - MATCHINGPATTERNTEST AANVULLEN
    #vul hier de code aan om te testen of een bepaalde kleurwaarde in de
    #oorspronkelijke figuur vanaf rij i en kolom j overeenkomt met de
    #kleurwaarde in het patroon.  waardenFiguur is de volledige matrix van een
    #bepaalde kleurwaarde in de oorspronkelijke figuur.  waardenPatroon is de
    #volledige matrix van een bepaalde kleurwaarde van een patroon.  Je mag
    #ervan uitgaan dat de dimensies van een patroon kleiner zijn dan die van de
    #oorspronkelijke figuur.


# converteer een ingelezen afbeelding naar drie matrices
input_image = Image.open("./beeld/landschap.jpg")
r_image_in, g_image_in, b_image_in = input_image.split()
r_in = np.uint32(np.array(r_image_in))
g_in = np.uint32(np.array(g_image_in))
b_in = np.uint32(np.array(b_image_in))

# converteer een ingelezen afbeelding naar drie matrices
pattern_image = Image.open("./beeld/patroon.jpg")
r_image_pat, g_image_pat, b_image_pat = input_image.split()
r_pat = np.uint32(np.array(r_image_pat))
g_pat = np.uint32(np.array(g_image_pat))
b_pat = np.uint32(np.array(b_image_pat))

#CODE - DEEL 1 - MAIN AANVULLEN
#schrijf hier de code om te testen of het opgegeven patroon voorkomt in de figuur.
#Maak hierbij herhaaldelijk gebruik van de methode matchingPatternTest

```

DEEL C: Algoritme – antwoordenblad 3

Tribonacci: Schrijf een algoritme om het n-de getal in de rij van Tribonacci te vinden. Dit is een variant op de rij van Fibonacci waar de eerste drie getallen 0, 1, 1 zijn, en alle volgende getallen de som van de vorige drie.
Het begin van de rij ziet er uit als volgt:

0,1,1,2,4,7,13,24,44,81,149, …

Schrijf het volledige Python algoritme om het n-de getal in deze rij te vinden. Dit algoritme moet werken voor ieder natuurlijk getal n>3.