Voorbeeldtentamen Voortgezette informatica (TVIT)

Ongeroosterd

Instructie

Dit voorbeeldtentamen is bedoeld om je een indruk te geven van het soort, en de hoeveelheid, vragen die je kunt verwachten op het tentamen. Je kunt deze maken tijdens de voorbereiding op de toets om te controleren of je de stof beheerst.

Er kunnen aan dit voorbeeldtentamen geen rechten worden ontleend. Er kunnen uiteraard ook vragen gesteld worden die onderdeel zijn de besproken stof, maar niet in dit voorbeeld aan bod komen.

Je hebt 3 uur de tijd voor het maken van de opgaven.

Succes!

Vraag 1: Eindige automaten (10)

(a) Wat is het nadeel van een non-deterministische eindige automaat tov een deterministische eindige automaat? (1)

(b) Hoe zou je kort het belangrijkste verschil tussen een reguliere expressie en een grammatica kunnen omschrijven? (1)

(c) Beschouw de taal van willekeurige strings bestaande uit de symbolen `a` en `b` waarin iedere `a` vooraf wordt gegaan en gevolgd wordt door minimaal 1 `b`. Bijvoorbeeld: `babab` of `bbabbabbb`. (2)

Maak voor deze taal een eindige automaat.

(d) Beschouw de taal van willekeurige strings bestaande uit de symbolen `a` en `b` waarin de substring `aba` of `aab` minimaal 1 keer voorkomt, bijvoorbeeld: `bbbaba`, `bbbaabbba`. (2)

Geef een reguliere expressie die deze taal beschrijft.

(e) Maak een eindige automaat die de taal uit de voorgaande vraag accepteert. Doe dit door eerst een niet deterministische versie te maken aan de hand van de reguliere expressie en deze daarna deterministisch te maken. Laat beide versies zien. (4)

Vraag 2: Grammatica's (8)

(a) Maak een grammatica voor de volgende taal bestaande uit de symbolen 0,1: (2)
Alle strings waarbij een string bestaande uit een aantal 0-en (1 of meer) gevolgd wordt door een oneven aantal 1-en en daarna net zoveel nullen als in het begin (bv 0011100 of 010)

(b) Maak een grammatica voor de volgende taal bestaande uit de symbolen 0,1: (2)
Alle strings die omgekeerd hetzelfde zijn (bv 00, 010, 1001). Ze mogen dus even of oneven lengte hebben.

(c) Geef de grammatica voor rekenkundige infix expressies met getallen, haakjes en de operatoren +,−,∗,/ met hun gebruikelijk prioriteiten. (2)

(d) Geef met behulp van de grammatica uit de vorige vraag de ontleedboom voor: `3 - 4 * 3 - 5 * ( 4 - 1)` (2)

Vraag 3: Turingmachines (10)

In de deelvragen mag je het volgende aannemen: de invoer bestaat uit 0, 1 en _ eventueel met hulpsymbolen. Je mag steeds zelf kiezen welke symbolen je schrijft. Rechts van de invoer is de tape gevuld met (oneindig veel) _.

(a) Een informaticus beweert een programmeertaal te hebben ontwikkeld die krachtiger is dan een Turing machine, dwz er kunnen berekeningen op worden gemaakt die niet met een Turing machine kunnen worden gedaan. Is dit mogelijk? (1)

(b) Geef een voorbeeld van een niet berekenbaar probleem. (1)

(c) De invoer bestaat uit een 0-1 string afgesloten met een #. Geef een Turing machine die de gehele string naar de tape achter # kopieert. Zorg er (zo nodig) ook voor dat de originele invoer weer wordt teruggezet. (4)

(d) De invoer bestaat uit enen gevolgd door nullen. Maak een Turing-machine die de invoer accepteert als het aantal enen groter is dan het aantal nullen. (4)

Vraag 4: IJMVM (14)

(a) Geef voor de volgende rekenkundige expressies IJVM code die de expressie uitrekent: (3)

- `4 * 7 -3 * 2 - 5`
- `3 * 8 - 5 + 3`

(b) Beschouw de volgende Python functie en aanroep: (4)

```
def g(n,k):
    a = 0
    while a <= n:
        a = a + k
    return n + k - a

print(g(8,3))
```

Geef de IJVM code van de functie g inclusief de aanroep en `print`.

(c) Geef aan hoe de stack er uitziet op het moment dat je de eerste keer de while loop uit de vorige vraag ingaat. (2)

(d) Beschouw de volgende IJVM code: (5)

```
   bipush 9
   call 1 1 11
   print
   stop
11 bipush 1
   istore 1
33 iload 0
   ifeq 22
   iload 0
   bipush 1
   isub
   istore 0
   bipush 1
   iload 1
   isub
   istore 1
   goto 33
22 iload 1
   ireturn
```

Geef hiervoor equivalente Python code.

Vraag 5: Recursie (8)

(a) Beschouw (nogmaals) de volgende python functie: (4)

```
def g(n,k):
    a = 0
    while a <= n:
        a = a + k
    return n + k - a
```

Geef een recursieve versie van de functie g.

(b) Beschouw de volgende recursieve python functie: (4)

```
def r(s):
    if len(s) <= 1:
        return s
    else:
        return r(s[1:]) + s[0]
```

Geef een iteratieve versie van de functie r.

EINDE TOETS