TENTAMEN ORIENTATIE INFORMATICA

4 februari 2008, 14:00-17:00 uur

Dit tentamen is GEEN open boek tentamen.

Voorzie de in te leveren bladen van je naam, en nummer ze. Schrijf op het eerste blad het aantal ingeleverde bladen. Met elk van de 11 vragen is 3 punten te verdienen. ***Je hoeft dus maar drie opgaven te maken****. Je krijgt 1 punt gratis. Succes!*

Opgave 1. NOAG-ict
Kies één van de NOAG-ict thema's, en schrijf een essay van 350-400 woorden, waarin je in je eigen woorden uitlegt wat het thema omvat, wat de belangrijkste uitdagingen zijn, en wat de belangrijktste maatschappelijke belangen zijn. Noem ook een aantal vakgebieden binnen de informatica die een rol spelen bij het thema.

Opgave 2. Geschiedenis
Alan Turing bedacht in de jaren 30 van de vorige eeuw het concept van de Turingmachine.

a. Geef een omschrijving van dit concept, i.h.b. de structuur en werking van de machine.

b. Geef aan waarin dit concept afwijkt van de mechanische rekenmachines die in die tijd al bestonden.

c. Geef de relatie aan tussen de Turingmachine en de huidige computer. Ga hierbij in in op aspecten van de huidige computer waarvan de Turingmachine abstraheert.

Opgave 3. Software Engineering
Schrijf een kort opstel (350-400 woorden), waarin je aangeeft hoe een bedrijf efficient goede software kan ontwikkelen. Leg ook uit waarom je oplossen werken! Behandel ten minste drie van de volgende vijf aspecten: Analysis, Design, Verification and Validation, Processes, Professional Practice.

Opgave 4. Gedistribueerde systemen

1) Beschouw de definitie van een gedistribueerd systeem gegeven tijdens het college. Volgt uit deze definitie dta het Web een gedistribueerd is? Beargumenteer je stellingname met voor- of tegenvoorbeelden.

2) Wat is het model van de groei van het Web?

3) Wordt de grootte van het Web gemeten of geschat? Leg uit hoe.

Opgave 5. Parallel rekenen
Speed-up is een belangrijke parameter die de performance van een parallel systeem of algorithme beschrijft als functie van het aantalprocessoren p dat gebruikt wordt.

a) Geef de definitie van speed-up, en leg uit wat het uitdrukt.

b) Wat is de ideale speed-up en waarom?

c) Wat is de slechtste speed-up die je tegen zal komen, en wat is er dan aan de hand.

Bijna ieder programmeerprobleem bevat een deel dat goed te parallelliseren is, en een stuk (zoals het lezen van data van schijf, dat alleen sequentiëel werkt).

d) Wat is de invloed hiervan op de speed-up, en schets in een grafiek hoe het gedrag van speed-up is als functie van p.

Opgave 6. Non-photorealistic rendering
In computer graphics was in het algemeen het doel om beelden te maken die zo goed mogelijk op echte foto's lijken. Pas recent zijn onderzoekers technieken gaan ontwikkelen die zich van dit ene doel "losweken" en hebben het vakgebied "non-photorealistic rendering" ontwikkeld.

Bespreek in 350-400 woorden wat deze nieuwe technieken te bieden hebben, wat toepassingsgebieden zijn, en welke uitdagingen er zijn die niet in de traditionele computer graphics te vinden zijn. Bespreek ook hoe interfaces een rol spelen binnen de computer graphics en in het creëeren van non-fotorealistische beelden.

Opgave 7. Softwarevisualisatie
Softwarevisualisatie onderzoekt methodes en technieken die inzicht proberen te bieden in de structuur, gedrag, evolutie en onderhoudbaarheid van grote en complexe softwaresystemen, door middel van het visualiseren van verschillende aspecten en attributen ervan. Een van de grootste uitdagingen van het begrijpen van een modern softwaresysteem is de groote, of schaal, van zo'n systeem. Op een scherm is maar een beperkte hoeveelheid informatie die vertoond kan worden, afhankelijk van de resolutie van dat scherm. Het is duidelijk dat er, ongeacht de groote van het scherm, altijd softwaresystemen zullen zijn die te groot en/of te complex zijn om vertoond te worden op een enkel scherm.

Moderne softwaresystemen hebben miljoenen regels code; de executie ervan levert tientallen megabytes gegevens op; en zulke systemen hebben honderden versies, geschreven door tientallen programmeurs.

Op basis van het collegemateriaal, beschrijf een aantal mogelijkheden waarmee softwarevisualisatie het schaalprobleem kan aanpakken, gegeven de beperkte resolutie van een computerscherm. In het bijzonder, beantwoord de volgende sub-vragen:

a) welke zijn efficiente en effectieve manieren om meer regels code te vertonen dan mogelijk met een klassieke broncode-editor? Welke mogelijkheid van degene die zijn gepresenteerd heeft de meeste schaalbaarheid? Waarom?

b) welke zijn de schaalbare manieren om softwarestructuur te visualiseren? Denk aan structuur zoals UML-softwarearchitectuur diagrammen. Is de UML-diagram visualisatie de meest schaalbare manier? Zo niet, waarom, en welke andere structuurvisualisatie is schaalbarder?

c) neem de situatie waarin men veel numerieke softwareattributen, zoals codekwaliteitsmetrieken of runtime-snelheid tegelijkertijd wil visualiseren. Hoe kunnen verschillende attributen tegelijkertijd worden getekend? Denk aan combinaties van visuele attributen zoals kleur, vorm, textuur, belichting, of animatie. Geef een voorbeeld van een applicatie van jouw keuze waarin tenminste drie zulke attributen worden gevisualiseerd voor elk software-element, zoals functie, klasse, of executiethread, en beschrijf door middel van welke visuele stimuli worden deze attributen gevisualiseerd.

Opgave 8. Computer Vision
Beeldanalyse systemen voor een specifieke toepassing kennen een reeks verschillende stappen die ze moeten nemen, in een min-of-meer sequentieële aanpak, beginnend bij acquisitie, en eindigend met herkenning en interpretatie. Neem het ADIAC project (diatomee herkenning) als voorbeeld. Benoem deze stappen in het ADIAC project, omschrijf wat de doelstelling van ieder stap is, en wat de belangrijkste obstakels zijn, in van 350-400 woorden.

Opgave 9. Visualisatie
De mate waarin visualisatietechnieken door gebruikers worden geaccepteerd verschilt enorm. Zo wordt de methode van Chernoff faces voor visualisatie van multivariate data nauwelijks gebruikt, terwijl de SequoiaView applicatie voor het visualiseren van de inhoud van de harde schijf van een PC, of de surface rendering methode voor de visualisatie van medische volume-data enorm succesvol zijn gebleken.

Geef een korte beschouwing (350-400 woorden) waarin je een verklaring geeft voor bovengenoemd verschijnsel. Betrek de volgende elementen in je beschouwing:

- het aantal gebruikers van de applicatie
- de frequentie waarmee de applicatie gebruikt wordt door de gebruiker
- hoe gemakkelijk nieuwe kennis kan worden verkregen door de gebruiker

- hoe waardevol de verkregen kennis is
- de ontwikkelkosten van de applicatie
- de steilheid van de leercurve van de applicatie

Opgave 10. Rekenen met Kekulécellen
Een koolwaterstofmolecuul wordt gerepresenteerd door een graaf met de koolstofatomen als knopen, en met de bindingen tussen de koolstofatomen als ribben. Een knoop met slechts één buurknoop heet een *poort*. Alle andere knopen zijn *intern*. Een *kanaal* is een tweetal verschillende poorten.

Een *Kekulétoestand* W van de graaf is een verzameling ribben zo, dat elke interne knoop op precies één van de ribben van W zit. De ribben in W heten *dubbele* bindingen, de ribben buiten W *enkele*. Een kanaal $\{p, q\}$ heet *open* in toestand W als er een alternerend pad tussen p en q is, dwz. een pad met om en om enkele en dubbele bindingen (het maakt niet uit hoe het pad begint en eindigt).

a) Teken een graaf met 3 poorten en 3 interne knopen, waarin de interne knopen een driehoek vormen en elke interne knoop met één poort verbonden is. Laat zien dat deze graaf vier en niet meer dan vier Kekulétoestanden heeft.

b) Teken een graaf met 4 poorten, 4 interne knopen, en 8 ribben, waarin de interne knopen een vierkant vormen en elke interne knoop met één poort verbonden is. Bepaal alle Kekulétoestanden. Hoeveel zijn het er?

c) Noem de poorten van de graaf van onderdeel b) met de klok mee achtereenvolgens a, b, c en d.
c1) Bepaal in welke Kekulétoestanden van deze graaf het kanaal (a, b) gesloten (dwz. niet open) is.
c2) Bepaal in ieder van de Kekulétoestanden W waarin het kanaal (a, b) open is, welke Kekulétoestanden verkregen kunnen worden door de enkele en dubbele bindingen langs een alternerend pad tussen a en b te verwisselen.

d) Niet iedere graaf heeft Kekulétoestanden. Geef een samenhangende graaf met twee poorten die geen Kekulétoestanden heeft. Motiveer je antwoord. Houd het bij voorkeur eenvoudig.

e) Geef een samenhangende graaf met twee poorten die precies één Kekulétoestand heeft. Motiveer je antwoord. Houd het bij voorkeur eenvoudig.

Opgave 11. Wederzijdse uisluiting
Er zijn een aantal processen die tegelijk een niet-eindigende lus uitvoeren:

```
while (true) {
     NCS // andere activiteit
     intro
     CS // critische sectie
     exit
 }
```

Het probleem van wederzijdse uitsluiting is de commando's intro en exit zo te programmeren dat er nooit meer dan één proces tegelijk in CS is. We mogen hiertoe variabelen gebruiken die door alle processen gelezen en geschreven worden.

(a) Beschrijf één of enkele alinea's het belang van wederzijdse uitsluiting voor computerprogramma's en computersystemen.

(b) Over implementaties.

Bij het toilet gebruikt men gewoonlijk een deur die van binnen op slot gedaan kan worden. We kunnen dit modelleren met een gedeelde variabele

```
boolean open = true ; // initieel

while (true) {
     NCS // andere activiteit
     while (! open) skip ;      // wacht tot de deur open is
     open = false ;             // doe de deur achter je op slot
     CS // critische sectie
```

```
        open = true ;                // doe de deur weer open
    }
```

Dit werkt helaas niet voor processen, want twee of meer processen zouden net na elkaar `open == true` kunnen lezen en dan allebei na elkaar `open = false` zetten en CS in kunnen gaan. Wel correct is:

```
    while (true) {
        NCS // onbelangrijk
wacht:  < if (! open) goto wacht ; else open = false ; >
        CS // critische sectie
        open = true ;
    }
```

De hoekjes `<S>` om een commando `S` drukken uit dat het commando `S` atomair uitgevoerd moet worden, dus zonder interferentie van andere processen. De opdracht "goto wacht" is een sprongopdracht, zoals we die in ons programmeeronderwijs jullie nooit leren (wacht is het label waarnaartoe gesprongen wordt). Het bezwaar van deze oplossing is de oneerlijkheid. Het zou kunnen zijn, dat een proces bij wacht staat te wachten en open telkens false vindt wanneer hij het test, omdat er steeds een ander proces voordringt.

Om dit te voorkomen voert men (bv) bij de apotheek een tellertje in. Dit is een gedeelde variabele

```
int teller = 0 ; // initieel
```

die in één atomaire opdracht gelezen en opgehoogd kan worden volgens

```
< eigenTeller = teller ; teller = teller + 1 ; >
```

b1) Programmeer nu eerlijke wederzijdse uitsluiting met deze teller. Zorg dat elk proces keurig op zijn beurt wacht. Argumenteer dat aan wederzijdse uitsluiting voldaan wordt en dat elk wachtend proces ooit aan de beurt komt. Je mag zelf desgewenst andere gedeelde variabelen en privévariabelen invoeren (maar houd het eenvoudig). Zorg dat alle variabelen goed geïnitialiseerd zijn.

b2) Neem nu aan dat de teller modulo 100 rekent, zodat de atomaire opdracht als volgt is

```
< eigenTeller = teller ; teller = (teller + 1) % 100 ; >
```

Onder welke omstandigheden blijft het algoritme dan correct? Onder welke omstandigheden gaat het fout, en wat gaat er dan fout?