DE HAAGSE HOGESCHOOL
Technology, Innovation & Society Delft

VOORBLAD SCHRIFTELIJKE TOETSEN

OPLEIDING	**: ELEKTROTECHNIEK**
TOETSCODE	**: RTSYST-SC1|REAL-TIME SYSTEMEN**
GROEP	**: ECV**
TOETSDATUM	**: 17 JUNI 2013**
TIJD	**: 11.00 – 12.30 uur**
AANTAL PAGINA'S (incl. voorblad)	**: 4**
DEZE TOETS BESTAAT UIT	**: 3** open vragen
GEBRUIK HULPMIDDELEN	**: JA**/~~NEE~~
TOETSOPGAVE INLEVEREN	**:** ~~JA~~/**NEE**
TOEGESTANE HULPMIDDELEN	**: Tijdens dit tentamen mogen alle boeken, dictaten, aantekeningen enz. worden gebruikt.**
OVERIGE OPMERKINGEN	**: Geen**

OPSTELLER VAN DEZE TOETS	**: Harry Broeders**
TWEEDE LEZER VAN DEZE TOETS	**: Jesse op den Brouw**

BELANGRIJKSTE PUNTEN UIT ARTIKEL 12 VAN DE ONDERWIJS- EN EXAMENREGELING:

- je dient je via Osiris ingeschreven te hebben voor deze toets
- schrijf je naam, je studentnummer, de toetscode en de naam van de docent meteen op het tentamenpapier
- leg je identiteitsbewijs op de hoek van de tafel
- zet alle elektronische communicatiemiddelen (mobiele telefoon, PDA, etc.) uit en stop deze in je tas; deze mogen niet als calculator of klok worden gebruikt
- je mag het lokaal het eerste halfuur niet verlaten
- volg de instructies op het toetsvoorblad
- steek je hand op als je een vraag hebt

HHS TIS Delft

DE HAAGSE HOGESCHOOL

KLAS(SEN)	:	ECV		BLAD	: 1 van 3 BLADEN
TOETS	:	real-time systemen (TOETS01)		DOCENT	: Harry Broeders
CODE	:	**RTSYST-co1**		DATUM	: 17 juni 2013
KWARTAAL	:	4	TYPE : tentamen	TIJD	: 11:00 – 12:30

Tijdens dit tentamen mogen **alle** boeken, dictaten, aantekeningen enz. worden gebruikt.

Bij elke opgave staat tussen haakjes het maximale aantal te behalen punten vermeld.
Eindcijfer = (aantal behaalde punten + 10) / 10.

1. In een beeldherkenningsapplicatie die geschreven is in C moeten de beelden van 3 zwart-wit camera's worden gebinariseerd (dit is een bepaalde bewerking die verder niet ter zake doet). Hiertoe is een functie geschreven `binarize` waarvan het prototype hieronder gegeven is. Deze functie heeft 2 parameters: een pointer `beeld` naar een array met `uint8_t` getallen die de grijswaarde van elk pixel in een camerabeeld weergeven en een `size_t` variabele `n` die het aantal pixels in de array weergeeft.

```
void binarize(uint8_t beeld[], size_t n);
```

In de functie `binarizeAll` wordt de functie `binarize` 3x achter elkaar aangeroepen om de beelden van de 3 verschillende camera's te binariseren.

```
void binarizeAll(uint8_t beeld1[], size_t n1, uint8_t beeld2[],
                                 size_t n2, uint8_t beeld3[], size_t n3 ) {
    binarize(beeld1, n1);
    binarize(beeld2, n2);
    binarize(beeld3, n3);
}
```

Als deze C functie op een multi-core machine wordt uitgevoerd wordt slechts 1 core belast.

(25) Laat zien hoe de bovenstaande functie aangepast moet worden om er m.b.v. de pthread library voor te zorgen dat het binariseren van de 3 beelden parallel (op verschillende cores) uitgevoerd kan worden. De functie `binarizeAll` mag pas eindigen als alle beelden gebinariseerd zijn.

2. Hieronder is een C++11 programma gegeven waarbij een concurrent proces wordt gesimuleerd. Er zijn bakkers die broodjes bakken. Elke bakker heeft een eigen oven en levert na een bepaalde tijd een aantal (`n`) broodjes af. Daarnaast zijn er ook meerder inpakkers die de broodjes in zakken van 10 stuks verpakken. De bakkers en de inpakkers werken parallel.

```
#include <thread>
#include <mutex>
#include <condition_variable>
#include <iostream>
using namespace std;

int broodjes = 0;
mutex m;
condition_variable c;
```

Zie volgende blad ⇨

```
void bakker(int n) {
    while (1) {
        this_thread::sleep_for(chrono::milliseconds(n * 100));
        {
            lock_guard<mutex> lock(m);
            broodjes += n;
            cout << "Er zijn " << n << " broodjes gebakken" << endl;
        }
        c.notify_all();
    }
}

void inpakker() {
    while (1) {
        {
            unique_lock<mutex> lock(m);
            while (broodjes < 10)
                c.wait(lock);
            broodjes -= 10;
        }
        this_thread::sleep_for(chrono::milliseconds(500));
        cout << "Zak met 10 broodjes gevuld" << endl;
    }
}

int main() {
    thread b1(&bakker, 100);
    thread b2(&bakker, 90);
    thread i1(&inpakker);
    thread i2(&inpakker);
    while (1);
    return 0;
}
```

A. (10) Geef de **exacte** uitvoer van het bovenstaande programma ervan uitgaande dat de het programma door de gebruiker na exact 10,7 seconden wordt afgebroken.

B. (5) Verandert de uitvoer van het bovenstaande programma als je de regel:
`c.notify_all();`
vervangt door:
`c.notify_one();`
Verklaar je antwoord!

In het bovenstaande programma wordt ervan uitgegaan dat er voldoende ruimte is om alle gebakken broodjes op te slaan totdat ze worden ingepakt. We willen nu een situatie simuleren waarbij er slechts plaats is om maximaal 150 broodjes op te slaan voordat ze worden ingepakt.

C. (20) Breid de bovenstaande code uit om te kunnen simuleren dat er slechts plaats is om maximaal 150 broodjes op te slaan voordat ze worden ingepakt.

Zie volgende blad ⇨

HHS TIS Delft

DE HAAGSE HOGESCHOOL

KLAS(SEN)	:	ECV		BLAD	: 3 van 3 BLADEN
TOETS	:	real-time systemen (TOETS01)		DOCENT	: Harry Broeders
CODE	:	**RTSYST-co1**		DATUM	: 17 juni 2013
KWARTAAL	:	4	TYPE : tentamen	TIJD	: 11:00 – 12:30

Tijdens dit tentamen mogen **alle** boeken, dictaten, aantekeningen enz. worden gebruikt.

3. Een programma bestaat uit 4 taken T1 t/m T4. Deze taken gebruiken geen gedeelde resources. In de onderstaande tabel staat *i* voor het nummer van de taak, *T(i)* voor de periodetijd van taak *i* en *C(i)* voor de maximale executietijd van taak *i*. Gegeven is dat de deadline van elke taak gelijk is aan zijn periodetijd.

i	***T(i)***	***C(i)***
1	100	50
2	280	45
3	200	20
4	300	40

Alle gegeven tijden zijn in ms.

A. (10) Bepaal de schedulability van deze taken met behulp van de "Utilization based schedulability test". Geef de benodigde berekening en trek daaruit je conclusie!

B. (5) Bepaal de prioriteiten *P(i)* van de verschillende taken als gebruik gemaakt wordt van FPS-RMPA (Fixed Priority Scheduling - Rate Monotonic Priority Assignment). Het systeem kent 4 verschillende prioriteiten (1 t/m 4) waarbij 4 de hoogste prioriteit is.

C. (15) Bereken voor taak 2, 3 en 4 of de deadline wordt gehaald en geef, indien de deadline wordt gehaald, de response tijd R_i.

Succes!