TU Delft Faculteit EWI

Tentamen Programmeren in C (EE1400)

5 april 2012, 9.00 – 12.00

- Zet op elk antwoordblad je naam en studienummer.
- Beantwoord alle vragen zo nauwkeurig mogelijk.
- Wanneer C code gevraagd wordt:
  - Denk aan de declaraties en initialisaties.
  - Gebruik een overzichtelijke indentering.
  - Functies uit de standard C library mogen gebruikt worden, tenzij anders vermeld.
  - Tenzij er expliciet om gevraagd wordt, hoeven er geen header files van de standard C library (bijv. stdio.h) te worden opgenomen.
  - Wellicht is het een goed idee om de code eerst uit te werken in klad en daarna het eindresultaat op het antwoordblad te zetten.
- Bij elke vraag staat aangegeven hoeveel punten maximaal met die vraag zijn te behalen. In totaal zijn maximaal 90 punten te behalen.
- Tijdens het tentamen mag je het boek "A Book on C" (of een ander boek over C), de collegeslides en de practicumhandleiding erbij houden. NIET toegestaan zijn oude toetsen en tentamens, en practicumuitwerkingen. Een eenvoudige, niet programmeerbare, rekenmachine is toegestaan: TI-30 en Casio FX-82, maar geen andere elektronische apparatuur.

Opgave 1 (24 punten)

Wat zal afgedrukt worden bij het uitvoeren van de volgende C code (let goed op de format specifiers):

a.
```
int a, b;
a = 10;
b = a++ + 2;
printf ("%d %d\n", a, b);
```

b.
```
int a, b = 2, c = 5;
a = --c;
b *= --c;
printf ("%d %d\n", a, b);
```

c.
```
short a, b;
a = 13 % 5;
b = 12 / 2.5;
printf ("%d %d\n", a, b);
```

d.
```
int a, b;
a = 0222;
b = 010 + a;
printf ("%d %x\n", a, b);
```

e.
```
unsigned a = 66, b = 14;
a = a | b;
b = (a & b) >> 1;
printf ("%d %d\n", a, b);
```

f.
```
char a;
int b = 4, c = 10;
a = 'a' + 'Z' - 'z';
b = c = 8 + (c == 10);
printf ("%c %d\n", a, b);
```

Opgave 2 (8 punten)

Wat zal afgedrukt worden bij het uitvoeren van de volgende C code:

a.
```
int i, z;
z = 0;
for (i = 0; i <= 6; i++) {
     z = z + 2;
}
printf ("%d\n", z);
```

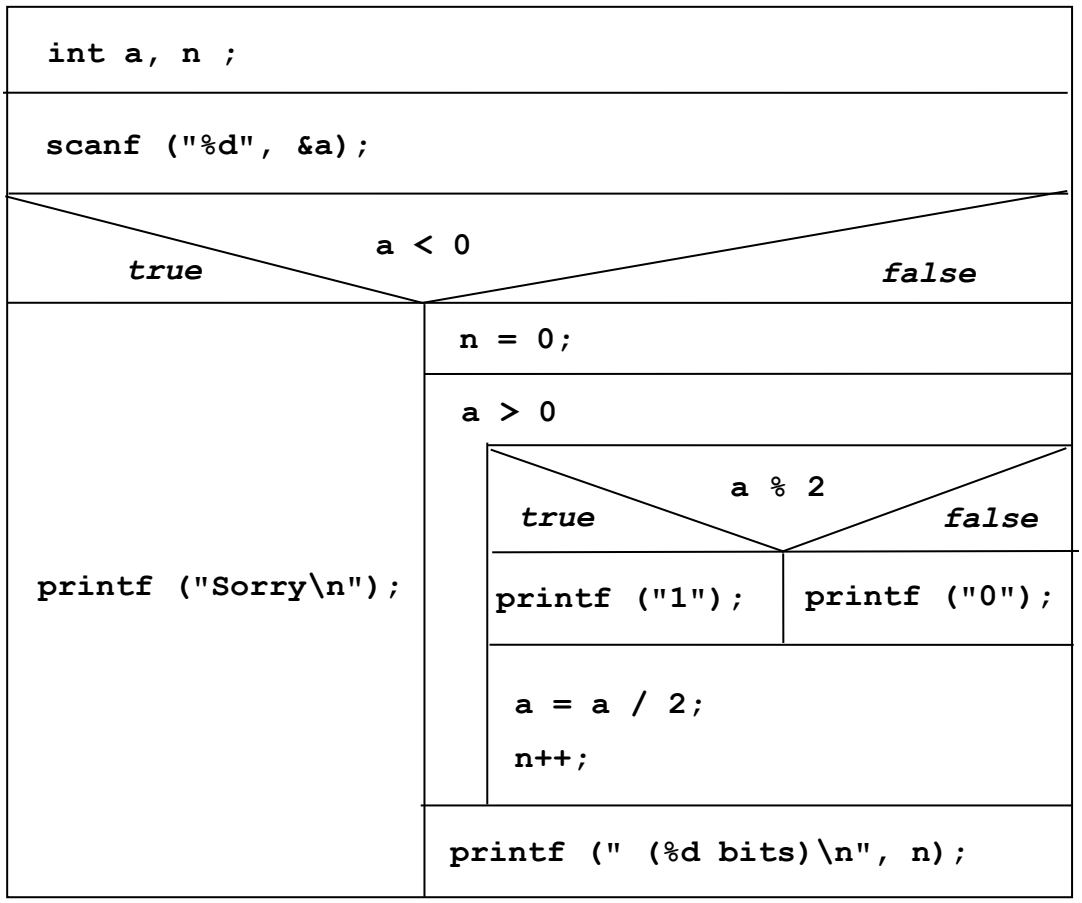
b.
```
int i, z;
z = 0;
for (i = 0; i <= 6; i++) {
     if (i > 1 && i < 4)
          continue;
     else
          z = z + 1;
     z = z + 10;
}
printf ("%d\n", z);
```

c.
```
int i, z;
z = 10;
for (i = 0; i <= 6; i++) {
     z--;
     if (i > z) break;
}
printf ("%d\n", z);
```

d.
```
int i, z;
i = 5;
z = 5;
while (i-- && z < 50) {
     z += 5;
}
printf ("%d\n", z);
```

Opgave 3 (8 punten)

Gegeven het volgende Nassi-Schneiderman diagram. Geef de bijbehorende C code.



Opgave 4 (8 punten)

Gegeven de volgende recursieve functie. Er geldt dat n >= 0.

```
int fi (int n)
{
     if (n <= 1)
          return n;
     else
          return fi (n-1) + fi (n-2);
}
```

Schrijf een iteratieve versie van deze functie. Hint: analyseer het gedrag en schrijf hiervoor C code waarin gebruik wordt gemaakt van een lus.

Opgave 5 (8 punten)

De lengte $|v|$ van een n dimensionele vector $v = (x_0, x_1, x_2, \ldots x_{N-1})$ wordt gegeven door

$$|v| = \sqrt{\sum_{i=0}^{n-1} x_i{}^2} = \sqrt{x_0^2 + x_1^2 + x_2^2 + \ldots + x_{n-1}^2}$$

Een genormaliseerde versie v' van de vector v kan worden verkregen door elk element x_i' van v' te bepalen aan de hand van:

$$x_i' = \frac{x_i}{|v|}$$

Schrijf nu een functie

```
void normalize (double x[], int n)
```

om de vector v gegeven door de array **`x[0], x[1], x[2], ... x[n-1]`** te normaliseren. De oorspronkelijke array dient daarbij te worden overschreven met de genormaliseerde vector.

Opgave 6 (8 punten)

Schrijf een functie om 2 strings samen te voegen tot een nieuwe string. Je mag GEEN gebruik maken van andere functies (dus ook niet van functies uit de standard C library zoals strcat() en sprintf()). De functie header dient er als volgt uit te zien:

```
void stradd (char *s, char *a, char *b)
```

waarbij na een aanroep van de functie de string s de samenvoeging dient te bevatten van string a en string b (b achter a). Neem aan dat s verwijst naar een char array welke groot genoeg is om het resultaat te bevatten.

Opgave 7 (8 punten)

Diergaarde Blijdorp wil wel eens weten welke verschillende soorten dieren ze hebben in de dierentuin en al deze soorten in alfabetische volgorde afdrukken. Om dat uit te zoeken schrijven ze een C programma. Het programma maakt gebruik van een linear linked list waarin elke diersoort wordt gerepresenteerd met een structure.

```
struct diersoort {
     char *naam;
     struct diersoort *volgend;
};
```

De lijst is alfabetisch gesorteerd op naam. Voor het eerste element uit de lijst geldt dat dit een 'dummy' element is met de naam **"a"** is. Voor het laatste element uit de lijst geldt dat de pointer 'volgend' gelijk is aan NULL.
Onderdeel van het programma is een functie waarmee een nieuwe diersoort aan de lijst kan worden toegevoegd. Aan u de taak om deze functie te schrijven. De functie krijgt als argumenten: (1) een pointer naar het begin van de lijst en (2) een pointer naar een structure met de nieuw toe te voegen diersoort, waarbij 'naam' de naam van de nieuwe soort bevat en 'volgend' NULL is. Gebruik voor de te schrijven functie, de volgende functie header:

```
void nieuw_soort (struct diersoort *lijst,
                  struct diersoort *ds)
```

Na het toevoegen van de nieuwe soort moet de lijst uiteraard nog steeds alfabetisch gesorteerd zijn op naam.

Opgave 8 (8 punten)

Voor een voetbalcompetitie worden in een matrix (2 dimensionaal array) a[N][N], de doelsaldo's bijgehouden: Element a[i][j] (i = 0 … N-1, j = 0 … N-1, i $\neq$ j) geeft aan hoeveel doelpunten team i tegen team j heeft gescoord. Verder geldt a[i][i] = 0 (i = 0 … N-1). Gevraagd wordt om een functie te schrijven welk voor elk team, op een aparte regel, naar stdout print: (1) het teamnummer, (2) het aantal doelpunten voor en (3) het aantal doelpunten tegen. Bijvoorbeeld:

```
0 112 40
1 70 96
2 90 60
etc.
```

Gebruik de volgende functie header:

```
void print_doelsaldos(int a[N][N])
```

waarbij N als constante is gedefinieerd met een #define.

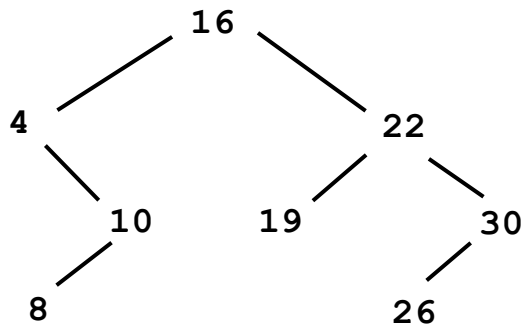
Opgave 9 (10 punten)

Gegeven een gesorteerde binary tree datastructuur waarin de nodes van de tree worden gerepresenteerd met de volgende structure:

```
struct node {
   int data;
   struct node *left;
   struct node *right;
};
```

Member left verwijst naar de top van de subtree waarin alle nodes zitten met een kleinere waarde dan de waarde gegeven door 'data', en member right verwijst naar de top van de subtree waarin alle nodes zitten met een grotere waarde. Wanneer er geen subtree is links of rechts, dan heeft het desbetreffende member de waarde NULL.

a. Gegeven de nevenstaande gesorteerde binary tree. Neem de tree over op je antwoordblad en geef duidelijk aan (gaat de verbindende lijn schuin naar links of schuin naar rechts ?) waar nieuwe nodes met de getallen 9 en 24 geplaatst moeten worden



b. Schrijf een functie waarmee de waarden in omgekeerde volgorde (d.w.z. van groot naar klein) worden afgedrukt (geprint naar stdout). Gebruik als functieheader:

```
void print_reverse (struct node * n)
```

waarbij n de root van de tree is (hint: gebruik recursie).

***** EINDE *****