Tentamen Programmeermethoden
woensdag 24 februari 2010, 14.00–17.00 uur
Universiteit Leiden — Informatica

Bij alle functies moeten de variabelen (constanten eventueel uitgezonderd) in de heading of als locale variabele voorkomen; vul zelf headings goed in. De opgaven tellen alle vier even zwaar mee. Succes! Cijfers: `http://www.liacs.nl/home/kosters/pm/cijf/res.html`.

1. We hebben twee arrays `A` en `B` met `n` (een `const` > 0) gehele getallen.
a. Schrijf een C++-functie `int hoevaak (X,A,n)` die bepaalt hoe vaak de integer `X` in `A` voorkomt.
b. Schrijf een C++-functie `voegtoe (A,i)` die array-element `A[i]` gesorteerd opbergt tussen de reeds oplopend gesorteerde array-elementen `A[0]`, `A[1]`, ..., `A[i-1]`. Doe dit door `A[i]` zolang als nodig met zijn voorganger te verwisselen. Neem aan dat $0 \leq$ `i` $<$ `n`.
c. Schrijf een C++-functie `uniek (A,B,n)` die alle getallen die precies één keer in `A` voorkomen oplopend gesorteerd vooraan in `B` zet. De eventueel resterende elementen van `B` moeten gelijk worden gemaakt aan 0. Gebruik **a** en **b**.
d. Hoeveel verwisselingen van array-elementen doet het algoritme van **c** (in aanroepen van `voegtoe`) *in het beste geval*, en wanneer is dat?
e. Hoeveel verwisselingen van array-elementen doet het algoritme van **c** (in aanroepen van `voegtoe`) *in het slechtste geval*, en wanneer is dat?

2.a. Bij een functie kun je te maken hebben met *call by value* en *call by reference*, en ook met *locale* en *globale* variabelen. Verder onderscheiden we ook nog *formele* en *actuele* parameters. Leg deze zes begrippen duidelijk uit.
b. Gegeven een C++-programma met daarin de volgende twee functies:

```
int sven (int a, int b) {
  int i, som = 0; z++;
  for ( i = a; i <= b; i++ ) som += i;
  cout << z << ": " << a << ", " << b << ", " << i << " en " << som << endl;
  a = -1; return som; a = 0;
}//sven

int bob (int p, int q) {
  return sven (sven (p,q),sven (p,q));  // (*)
}//bob
```

Verder zijn de globale variabelen `x`, `y` en `z` gegeven (alle van type `int`). Voordat de functie `bob` wordt aangeroepen hebben zij de waarde 1, 3 en 0, respectievelijk. Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit):

```
x = bob (x,y); cout << x << ", " << y << " en " << z << endl;
```

c. Als **b**, maar nu met een `&` ("ampersand") bij de vier parameters van de functies.
d. Vervang de regel bij `(*)` door:

```
return ( sven (p,q) + sven (q,p) );  // (*)
```

Voeg verder weer een `&` toe bij de vier parameters van de functies. Maak nu opnieuw **b**. Er zijn verschillende antwoorden mogelijk; geef deze, en leg uit.
e. De functie `sven` mag de functie `bob` nu niet aanroepen. Waarom niet, en wat moet worden toegevoegd om dit wel mogelijk te maken?

3. Gegeven is een `m` bij `n` (beide `const` > 0) array `temper`; `temper[i][j]` stelt de temperatuur (tussen -50 en $+50$) op punt $(\texttt{i}, \texttt{j})$ voor, waarbij 99 staat voor "onbekend".

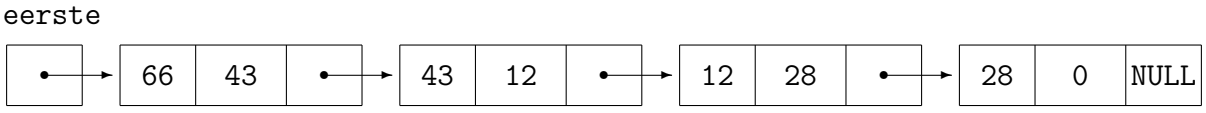
```
12 -1 -3 -3 -6  9
99 -1  2  2  2  7
99 -1 -1 99 99 99
```

a. Schrijf een C++-functie `int tel (temper)` die telt hoeveel rijen van de matrix zowel een temperatuur > 0 als een temperatuur < 0 bevatten. De waarde 99 telt niet mee. In het voorbeeld: 2.
b. Schrijf een C++-functie `bool schaatsbaan (temper,i,j,p,q)` die kijkt of je vanuit punt $(\texttt{i}, \texttt{j})$ in punt $(\texttt{p}, \texttt{q})$ kunt komen, waarbij je herhaald naar een verticaal aangrenzend punt mag gaan, of herhaald naar een horizontaal aangrenzend punt (maar niet gemengd). Alle gebruikte punten moeten een temperatuur < 0 hebben. Als $(\texttt{i}, \texttt{j}) = (\texttt{p}, \texttt{q})$ is het geen schaatsbaan. In het voorbeeld is er een schaatsbaan van $(0, 2)$ naar $(0, 4)$, maar niet van $(0, 4)$ naar $(1, 1)$, en ook niet van $(0, 2)$ naar $(2, 2)$. Neem aan dat $0 \leq \texttt{i}, \texttt{p} < \texttt{m}$ en $0 \leq \texttt{j}, \texttt{q} < \texttt{n}$.
c. Schrijf een C++-functie `bool kruis (temper)` die bepaalt of er een horizontale en een verticale schaatsbaan zijn die elkaar *kruisen*, dat wil zeggen precies één punt gemeenschappelijk hebben. Gebruik **b**. In het voorbeeld is het resultaat `true`.

4. Gegeven is het volgende type:

```
class hetgetal { public: int info; int point; hetgetal* volg; };
```

Met behulp hiervan worden lijstjes met getallen opgebouwd. Het veld `volg` bevat een pointer naar het volgende `hetgetal`-object; en het `point`-veld bevat hetzelfde getal als het `info`-veld van het door deze pointer aangewezen `hetgetal`-object (`0` als dat `NULL` is). Een voorbeeld (`eerste` van type `hetgetal*`):



a. Schrijf een C++-functie `verwissel (eerste)` die de twee eerste *objecten* — indien aanwezig — van de lijst (met `eerste` van type `hetgetal*` als ingang) verwisselt. De `point`-velden moeten zonodig aangepast worden.
b. Schrijf een C++-functie `voegtoe (eerste,get)` die een nieuw `hetgetal`-object met `get` in het `info`-veld vooraan de lijst (met `eerste` van type `hetgetal*` als ingang) toevoegt. Het `point`-veld moet op de juiste manier gevuld worden.
c. Schrijf een C++-functie `verwijder (eerste)` die het *tweede* `hetgetal`-object uit de lijst (met `eerste` van type `hetgetal*` als ingang) verwijdert, mits dat er is. Denk dus aan de lege lijst en aan een lijst met één element. Let ook weer op de `point`-velden.
d. In de functies bij **a**, **b** en **c** staat in de heading een pointer. Deze heb je call by value of call by reference doorgegeven (met een `&`). Maakt het voor de werking van deze functies verschil uit of die `&` erbij staat? Leg duidelijk uit.
e. Een boef heeft ergens aan de lijst (met `eerste` van type `hetgetal*` als ingang) stiekem een `hetgetal`-object toegevoegd, met twee niet in de lijst voorkomende getallen erin. Schrijf een C++-functie `corrigeer (eerste)` die dit ene object weer verwijdert uit de lijst, en verder niets verandert — en daarmee de velden weer aan de eisen laat voldoen. Neem aan dat de oorspronkelijke lijst minstens twee elementen heeft.