Tentamen Programmeermethoden
Vrijdag 4 januari 2013, 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. In een array `int A[n]` staan `n` (een `const` > 0) verschillende gehele getallen.
a. Schrijf een C++-functie `klein (A,i,kl,n)` die de *array-index* van het kleinste getal uit `A[i]`, `A[i+1]`, ..., `A[n-1]` in de parameter `kl` oplevert. Neem aan dat $0 \leq$ `i` $<$ `n`.
NB Geef hier (en ook in de andere opgaven) de compleet ingevulde heading van de functie!
b. Schrijf een Booleaanse C++-functie `gesorteerd (A,i,n)` die precies dan `true` teruggeeft als `A[i]` $<$ `A[i+1]` $< \ldots <$ `A[n-1]`, en anders `false`. Neem weer aan dat $0 \leq$ `i` $<$ `n`.
c. Schrijf een C++-functie `sorteer (A,n)` die het array `A` oplopend sorteert door herhaald te kijken of het steeds kleiner wordende staartstuk al gesorteerd is, zo ja te stoppen, en zo nee het kleinste getal van het nog resterende gedeelte met het voorste te wisselen. Gebruik de functies van **a** en **b**.
d. Hoeveel vergelijkingen tussen array-elementen doet `sorteer` minimaal? Tel hierbij zowel de vergelijkingen die in de aanroepen van `klein` als in die van `gesorteerd` worden gedaan. In welk(e) geval(len) gebeurt dit?
e. En hoeveel vergelijkingen tussen array-elementen doet `sorteer` maximaal?

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:

```
bool mark (int a, int b) {
  int z;
  a = a + b; b = a - b; a = a - b; cout << "M" << a << "," << b << endl;
  z += 10; return ( a < b );
}//mark
int diederik (int b, int a) {
  bool temp; if ( mark (b,a) ) a += 2;
  while ( a > 0 ) { temp = mark (a,b); a--; cout << a << "," << b << endl; }
  z += 10; return ( a + b + 2 );
}//diederik
```

Verder zijn de globale variabelen `x`, `y` en `z` gegeven (van type `int`). Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit; tip: 9 komma's):

```
x = 1; y = 3; z = 1; x = diederik (y,z); cout << x << "," << y << "," << z;
```

c. Als **b**, maar nu met een `&` ("ampersand") bij de vier parameters van de functies.
d. Als **c**, dus met vier `&`'s erbij, voor:

```
x = 1; y = 3; z = 1; y = diederik (y,y); cout << x << "," << y << "," << z;
```

e. Als in de functie `mark` ergens `a = diederik (static_cast<int>(mark (a,b)),y);` staat, compileert het programma dan nog? Onderscheid gevallen met en zonder `&`.

3. Gegeven is een `m` bij `m` (een `const` > 0) Booleaans array `T`. Hierbij geeft `T[i][j]` aan of er een rechtstreekse trein van station `i` naar station `j` rijdt (`true`) of niet (`false`). Er gaat nooit een directe trein van een station naar zichzelf.

```
false  false  true
true   false  true
false  true   false
```

a. Schrijf een C++-functie `int duos (T)` die berekent hoeveel duos $(\mathtt{i}, \mathtt{j})$ met $0 \leq \mathtt{i} < \mathtt{j} < \mathtt{m}$ er in `T` zijn, waarvoor geldt dat er zowel een rechtstreekse trein van `i` naar `j` is als van `j` naar `i`. In het voorbeeld: 1, namelijk $(1, 2)$.

b. Schrijf een C++-functie `int druk (T)` die het station met de meeste in- en uitgaande directe verbindingen geeft. Als er meer stations met deze eigenschap zijn, geef dan dat met het hoogste nummer. In het voorbeeld: station 2 (3 directe verbindingen, net als 1).

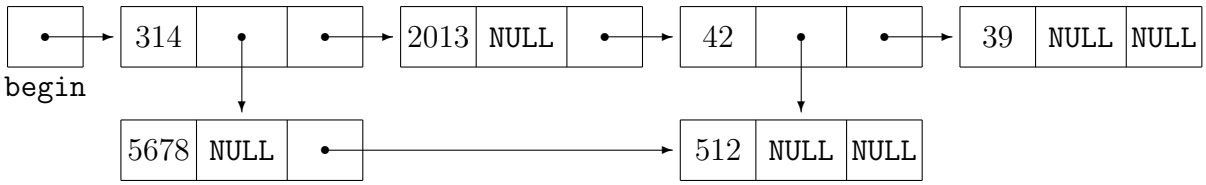
c. Schrijf een Booleaanse C++-functie `bereik (T,i,j)` die precies dan `true` teruggeeft als je `j` vanuit `i` kunt bereiken met één overstap. Of er ook nog een rechtstreekse verbinding is, doet er niet toe. Neem aan dat $0 \leq \mathtt{i}, \mathtt{j} < \mathtt{m}$ en $\mathtt{i} \neq \mathtt{j}$.

d. Schrijf een C++-functie `int aantal (T,i)` die bepaalt hoeveel stations je, beginnende in station `i`, als volgt bezoekt, totdat je niet meer verder kunt. Je gaat steeds, met precies één overstap (gebruik de functie van **c**) naar het eerstvolgende station met een hoger nummer. In het voorbeeld: vanuit $\mathtt{i} = 1$ reis je (met overstap in 0) naar 2, en klaar, met 2 bezochte stations. De stations waar je overstapt worden niet meegeteld.

4. Gegeven is het volgende type:

```
class element { public: int info; element* rechts; element* onder; };
```

Hiermee wordt een graaf-achtige structuur gemaakt. Het veld `rechts` bevat een pointer naar het eerstvolgende rechts ernaast gelegen `element`-object. De `onder`-pointer (steeds het middelste vakje in het voorbeeld) wijst naar het er direct onder gelegen `element`-object. Neem aan dat er maximaal twee rijen zijn, en dat boven elk object uit de tweede rij altijd precies één object uit de eerste zit. Een voorbeeld (`begin` van type `element*`):



a. Schrijf een C++-functie `verwijder (begin)` die het eerste `element`-object uit de structuur (met `begin` van type `element*` als ingang) netjes verwijdert, mits dat object bestaat. Als er een `element`-object onder is, moet dit eerst worden verwijderd. In het voorbeeld moeten de objecten met 5678 en 314 erin verwijderd worden.

b. Schrijf een C++-functie `voegtoe (begin,getal)` die een nieuw `element`-object met `getal` erin vooraan de structuur met ingang `begin` toevoegt — mits de structuur leeg is, of het voorste getal uit de structuur even is (zoals in het voorbeeld).

c. Schrijf een C++-functie `voegtoe2 (begin,getal)` die een nieuw `element`-object met `getal` erin onder het eerste object van de structuur met ingang `begin` toevoegt — mits dat eerste object bestaat, en daar nog geen object onder zit, en anders niets doet. Zet de `rechts`-pointer voorlopig op `NULL`.

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? Mag het, moet het? Leg duidelijk uit.

e. Schrijf een C++-functie `zetrechts (begin)` die alle `rechts`-pointers in de tweede rij goed zet, ongeacht wat hun waardes waren. Denk aan de laatste!