Tentamen Programmeermethoden
Dinsdag 5 januari 2016, 14:00–17:00 uur
Universiteit Leiden — Informatica

Bij alle functies moeten de variabelen (constanten eventueel uitgezonderd) in de heading of locaal voorkomen; vul zelf headings goed in. De te behalen punten (totaal 100) staan tussen haakjes bij de opgaven. Succes! Cijfers: `www.liacs.leidenuniv.nl/~kosterswa/pm/cijf/res.html`.

1. (25) In een array `int A[n]` staan `n` (een `const` ≥ 1) gehele getallen > 0.
a. (5) Schrijf een Booleaanse C++-functie `hoe (A,X,n)` die bepaalt of er een getal dat *hooguit* 1 van het gehele getal `X` verschilt in het array `A` voorkomt. Gebruik geen `(f)abs`.
b. (8) Schrijf een C++-functie `int langste (A,n,gem)` die de lengte uitrekent van een langste stijgende (of beter: niet-dalende) aaneengesloten deelrij in het array `A`. Hierbij moet `gem` het op de gebruikelijke manier naar een geheel getal afgeronde gemiddelde zijn van de getallen in de betreffende deelrij. Voor het array `1 7 1 1 2 7 6 3` zou het antwoord 4 zijn, namelijk de lengte van de deelrij `1 1 2 7`. En `gem` moet 3 worden (via 11/4). Als er meer deelrijen dezelfde maximale lengte realiseren: het gemiddelde van de eerste.
c. (8) Schrijf een C++-functie `busort (A,n)` die het array `A` met de volgende variant van *bubblesort* oplopend sorteert. In de eerste ronde wordt het gehele array van links naar rechts doorlopen, in de tweede ronde van rechts naar links, in de derde van links naar rechts, etcetera. Stop als er tijdens een ronde geen verwisselingen waren.
d. (4) Hoeveel vergelijkingen tussen array-elementen worden minimaal/maximaal uitgevoerd in de functie van **c**? Geef voor beide situaties een voorbeeld.

2. (25) **a.** (6) 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. (6) Gegeven een C++-programma met daarin de volgende twee functies:

```
int hillary (int n, int m) {
  n--; return n+m-1; }//hillary
int donald (int n, int m) {
  int a = 6; m--; n += 2; a++;
  m = n + hillary (n,m) + hillary (m,n) + a;
  cout << a << "," << m << "," << n << endl; return a+n-m; }//donald
```

Verder zijn de globale variabelen `a` en `m` gegeven, beide van type `int`. Wat is dan de uitvoer van het volgende stukje programma (leg je antwoord duidelijk uit):

```
a = 1; m = 4; cout << donald (m,a) << endl;
cout << a << "," << m << endl;
```

c. (5) We voegen nu vier maal een `&` toe bij de parameters in de heading van `hillary` en `donald`. Beantwoord opnieuw vraag **b**; leg uit waarom verschillende uitkomsten mogelijk zijn, en geef deze.
d. (4) Als in de functie `hillary` ergens `a = donald (42,a-a);` staat, compileert het programma dan nog? Onderscheid gevallen met en zonder `&`.
e. (4) Geef een eenvoudige uitdrukking voor de functiewaarde van `donald (r,s)`, uitgedrukt in `r` en `s`, voor de situatie zonder `&`'s (net zoals bij **b**).

3. (25) Gegeven is een `m` bij `n` (beide `const` > 0) array `M` met gehele getallen ≥ 0. Hierbij geeft `M[i][j]` ≥ 0 het aantal mensen ter plaatse $(\mathtt{i}, \mathtt{j})$ aan (met $0 \leq \mathtt{i} < \mathtt{m}$ en $0 \leq \mathtt{j} < \mathtt{n}$). Zie hiernaast voor een voorbeeld met $\mathtt{m} = 4$ en $\mathtt{n} = 5$. De constantes `m` en `n` hoeven bij deze opgave niet doorgegeven te worden als parameter.

3	0	4	2	0
7	2	4	4	2
1	2	0	4	1
6	9	3	1	0

a. (7) Schrijf een C++-functie `int druk (M,som)` die de index van de drukste kolom, dat wil zeggen de kolom met de grootste som, geeft. In `som` moet de desbetreffende som komen. In het voorbeeld kolom 0, met `som` gelijk aan 17. Als er meerdere kolommen dit maximum realiseren, geef degene met de hoogste kolomindex.

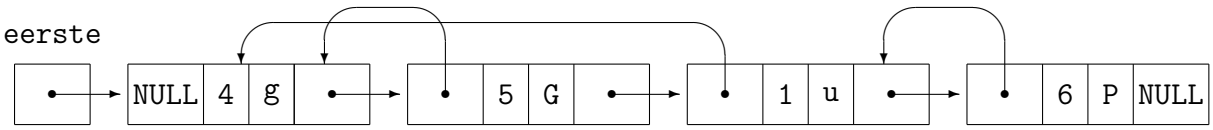
b. (8) Schrijf een C++-functie `int duos (M)` die het aantal horizontaal of verticaal direct aan elkaar grenzende (eventueel overlappende) paren geeft, waarbij de een precies het dubbele van de ander is. In het voorbeeld: 4–2, 2–4, 4–2, 1–2, 2–4, 2–1: 6 stuks.

c. (10) We lopen als volgt door het array `M`. Start bij $(\mathtt{i}, \mathtt{j})$. De waarde van dit array-element geeft aan hoeveel plaatsen naar rechts je gaat; de waarde van de nieuwe plek bepaalt hoeveel plaatsen je omlaag gaat, enzovoorts. Als je rechts het array uitloopt kom je in dezelfde rij links weer binnen, en analoog voor de verticale richting. De wandeling stopt als je ergens komt waar je al eerder bent geweest (in het bijzonder bij een verplaatsing van 0). Schrijf een C++-functie `int aantal (M,i,j)` die het aantal stappen uitrekent; in `i` en `j` moeten de coördinaten van het laatst bezochte punt komen. In het voorbeeld, beginnend bij $(0, 0)$ eerst 3 naar rechts, dan 2 omlaag, dan 4 naar rechts, dan 0 omlaag en klaar: antwoord 4, en $(\mathtt{i}, \mathtt{j}) = (2, 2)$. Hint: gebruik een Booleaans hulparray.

4. (25) Gegeven is het volgende type:

```
class kamer { public: kamer* vorig; int nr; char naam; kamer* volg; };
```

Hiermee wordt een lijst van kamers gemaakt (`naam` is een kleine letter of hoofdletter). Het veld `volg` bevat een pointer naar het volgende `kamer`-object, en `vorig` naar het vorige *of* voor-vorige, *of* `NULL` als die er beide niet zijn. Een voorbeeld (`eerste` van type `kamer*`):



a. (5) Schrijf een C++-functie `voegtoe (eerste,kamernr,kamernm)` die een nieuw `kamer`-object met `kamernr` en `kamernm` erin vooraan in de lijst met ingang `eerste` toevoegt. Zet de `vorig`-pointer van het oude voorste object (als dat bestond) ook goed.

b. (5) Schrijf een C++-functie `verwijder (eerste)` die het voorste `kamer`-object uit de structuur die door `eerste` wordt aangewezen, netjes verwijdert — mits het bestaat. Zet eventuele `vorig`-pointers die er naar wijzen goed.

c. (5) Schrijf een C++-functie `wissel (eerste)` die de kamer-nummers van de twee voorste kamers omwisselt, mits de ene letter de hoofdletter van de andere is (zoals in het voorbeeld; 4 en 5 worden verwisseld). Controleer of de lijst minstens twee objecten heeft.

d. (4) 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. (6) Schrijf een C++-functie `herstel (eerste)` die alle `vorig`-pointers, behalve die van de voorste twee objecten (zo die al bestaan), naar het voor-vorige object laat wijzen. In het voorbeeld zou de pointer bij het object met `P` erin naar het object met `G` erin moeten gaan wijzen, verder verandert er niets.