Tentamen Programmeermethoden
Donderdag 15 maart 2018
14:00–17:00 uur Informatica

Universiteit Leiden
The Netherlands

Bij alle functies moeten de variabelen (constanten 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 het array `int A[n]` staan `n` (een `const` ≥ 5) gehele getallen.
a. (5) Schrijf een C++-functie `int groot (A,n)` die de grootste waarde uit `A` teruggeeft.
b. (8) Schrijf een C++-functie `int langst (A,start,n)` die de lengte teruggeeft van een langste serie gelijke waardes die direct naast elkaar staan: een *plateau*. In `start` moet de array-index van het eerste element van dit plateau worden opgeleverd (de kleinste indien er meer langste plateaus zijn).
c. (4) Schrijf een C++-functie `busorteer (A,n)` die het array `A` met behulp van de meest eenvoudige versie van *bubblesort* oplopend sorteert.
d. (8) Schrijf een C++-functie `speciaal (A,n)` die het array `A` als volgt oplopend sorteert. Kijk eerst of er een langste plateau is van lengte `n` $-$ `2` (gebruik **b**) dat op positie `1` begint, en bestaat uit de grootste waarde van `A` (gebruik **a**). Als dat niet zo is, gebruik de functie van **c** om te sorteren, en anders: wissel zonodig nog enkele array-elementen.

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 piet (int x) {
  x--; cout << x << endl; return ( 2 * ( x + 1 ) ); }//piet
int theo (int u, int v) {
  u = u + v; v = u - v; u = u - v; x = u;
  u = piet (piet (piet (v))); v = piet (piet (piet (x)));
  return u + v + 2;
}//theo
```

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

```
x = 2; y = 3; cout << theo (x,y) << ","; cout << x << "," << y << endl;
```

c. (5) Wat levert op:

```
x = 2; cout << theo (x,x) << ","; cout << x << endl;
```

d. (4) Als **c**, maar nu met twee `&`'s in de heading van `theo`.
e. (2) Is hier ergens sprake van recursie? Leg uit.
f. (2) Mag in de heading van `piet` een `&` voor `x` staan, en waarom wel/niet?

3. (25) Gegeven bij deze opgave is het 2-dimensionale array `int doelpunten[n][n];`, met zekere `const int n` ≥ 2. Er geldt dat `doelpunten[i][j]` ≥ 0 het aantal goals van club `i` bij de wedstrijd van club `i` tegen club `j` is. Een voorbeeld met `n = 5` staat hiernaast. Zo won club `0` met 5–1 van club `1`.

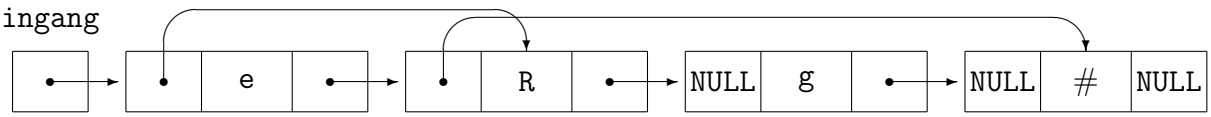
0	5	0	3	2
1	0	1	4	1
2	1	0	3	0
3	1	0	0	3
0	6	3	0	0

a. (8) Schrijf een C++-functie `int score (doelpunten,i,totaal)` die de totaalscore van club `i`, met `0` $\leq$ `i` $<$ `n`, bepaalt. Bij winst krijgt een club 3 punten, bij gelijk spel 1, en bij verlies 0. In het voorbeeld voor club `0`: `3 + 0 + 1 + 3 = 7`. In `totaal` wordt het aantal door club `i` gemaakte doelpunten opgeleverd, in het voorbeeld `10`.
b. (7) Schrijf een C++-functie `int winnaar (doelpunten)` die de winnaar van deze competitie bepaalt: degene met de hoogste totaalscore. Als er meer clubs de hoogste score realiseren, wint degene daarvan met de meeste gemaakte doelpunten. Als dat nog geen uitsluitsel biedt, wint de club met het laagste nummer. In het voorbeeld wint club `0`: de hoogste score, en meer doelpunten dan club `2` (die ook `7` punten heeft).
c. (10) Begin bij een zekere club `i1`, met `0` $\leq$ `i1` $<$ `n`. Zoek de club `i2` met de kleinste index, waarvan `i1` gewonnen heeft. Ga op dezelfde manier verder. Dit stopt als we bij een club komen die nooit gewonnen heeft, of bij een club die we al eerder hebben gehad. Schrijf een C++-functie `int keten (doelpunten,i1)` die dit doet, en teruggeeft hoeveel verschillende clubs je dan gezien hebt, inclusief `i1`. Tip: gebruik een 1-dimensionaal Booleaans hulparray. In het voorbeeld, met `i1` $=$ `0`, krijgen we `0` $\to$ `1` $\to$ `3` $\to$ `4` $\to$ `1`, dus `4`.

4. (25) Gegeven is het volgende type:

```
class object { public: object* tja; char info; object* volgende; };
```

Met behulp hiervan kan een lijst van objecten worden opgebouwd, bestaande uit vakjes met een karakter, en twee pointers. De `volgende`-pointer wijst zoals gewoonlijk naar het volgende object, de `tja`-pointer is óf `NULL`, óf wijst naar het volgende, óf naar het daaropvolgende object. Een voorbeeld, met `ingang` van type `object*`:



a. (6) Schrijf een C++-functie `voegtoe (ingang,letter)` die een nieuw object met `letter` erin netjes vooraan de lijst met ingang `ingang` toevoegt. Als het een kleine letter is moet de `tja`-pointer naar het volgende object gaan wijzen (`NULL` als dat er niet is), als het een hoofdletter is naar het daaropvolgende (idem), en anders `NULL` worden.
b. (5) Schrijf een C++-functie `verwijder (ingang)` die het eerste object uit de lijst met ingang `ingang` netjes verwijdert, als dat er is en indien het karakter in dat eerste object een klinker (kleine letter `a`/`e`/`i`/`o`/`u`) is — zoals in het voorbeeld.
c. (5) Schrijf een C++-functie `wissel (ingang)` die de `info`-velden (*niet* de pointers) van de twee voorste objecten omwisselt, indien er minstens drie objecten zijn.
d. (3) 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 `object* eennalaatste (ingang)` die een pointer naar het op een na laatste object retourneert; neem aan dat de lijst minstens drie elementen heeft. Hierbij moeten zoveel mogelijk de `tja`-pointers afgelopen worden als deze niet `NULL` zijn, en anders `volgende`-pointers. (De `volgende`-pointers mogen wel bekeken worden.)