Tentamen Programmeermethoden
Donderdag 27 maart 2025
13:00–16:00 uur Informatica

Universiteit Leiden
The Netherlands

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: te zijner tijd via Brightspace/uSis.

1. (25 punten) In het array `int A[n]` staan `n` (een `const int` ≥ 3) verschillende gehele getallen.
a. (4) Schrijf een Booleaanse C++-functie `zoek (A,X,n)` die bepaalt of het gehele getal `X` in `A` voorkomt. Stop zodra het antwoord bekend is.
b. (4) Schrijf een C++-functie `int invers (A,n)` die bepaalt hoeveel elementen in `A` groter zijn dan hun directe rechter buur.
c. (5) Geef een C++-functie `drie (A,i,n)` die de drie elementen `A[i]`, `A[i+1]` en `A[i+2]` oplopend sorteert door *precies* drie vergelijkingen te doen. Neem aan dat $0 \leq$ `i` $<$ `n` $- 2$.
d. (7) Schrijf een C++-functie `int sort (A,n)` die vergelijkbaar met *bubblesort* `A` oplopend sorteert. Roep systematisch de functie van **c** aan. Maak handig gebruik van de functie van **b** (stoppen als gesorteerd). De functie `sort` geeft het aantal gedane bubblesort-rondes als return-waarde terug.
e. (5) Hoeveel vergelijkingen tussen array-elementen doet de functie van **d** precies (via de aanroepen van `invers` en `drie`) in het slechtste geval en wanneer is dat?

2. (25 punten) **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. (7) Gegeven een C++-programma met daarin de volgende twee functies:

```
int clark (int r, int s) {
   int x = z, i; z++;
   for ( i = s; i >= 0; i-- ) { x = x / 2; }//for
   cout << z << "," << r << "," << s << "," << i << " en " << x << endl;
   s--; r = 0; return x; r = 4242; }//clark
int diana (int p, int q) {
   int y = clark (p,q); q++; int x = clark (p,q);
   return 4 * clark (x,y); }//diana
```

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

```
   x = 8; y = 4; z = 64; x = diana (x,y) + 10;  //(*)
   cout << x << "," << y << " en " << z << endl;
```

c. (7) Als **b**, maar nu met een `&` bij de vier parameters van de twee functies, en bij `(*)`: `diana (y,y)` in plaats van `diana (x,y)` .
d. (5) Mag de implementatie van functie `diana` vervangen worden door:

```
   return clark (clark (q,p),clark (p,q));
```

Onderscheid de gevallen met en zonder `&`.

3. (25 punten) Gegeven is een `m` bij `n` (beide `const int` > 0; ze hoeven bij deze opgave niet te worden doorgegeven als parameter) array `Zee` met integers. Hierbij stelt iedere horizontale of verticale aaneengesloten reeks getallen > 0 een schip voor, waarbij de waarde gelijk is aan de lengte van het schip. Een voorbeeld met `m` $= 4$ en `n` $= 5$ staat hierboven. Hier heeft element `Zee[0][2]` waarde `1` (een schip van lengte 1).

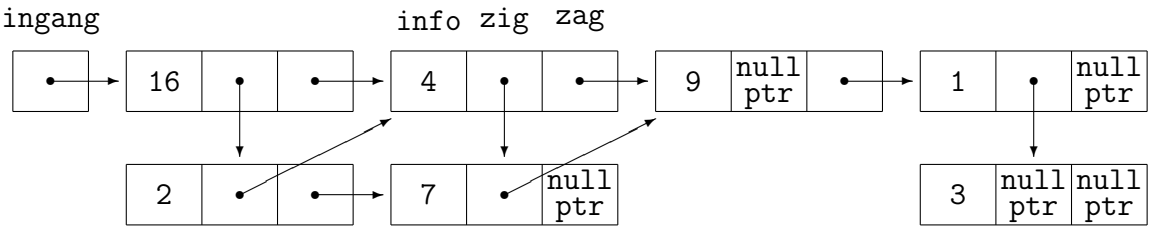
```
0 0 1 0 0
0 3 0 2 2
0 3 0 0 0
0 3 0 2 2
```

a. (7) Schrijf een C++-functie `int kruis (Zee,i,j)` die telt hoeveel van de (maximaal) vijf getallen in `Zee` op positie (`i`, `j`) en diens (maximaal) vier directe orthogonale (= horizontaal/verticaal) buren niet 0 zijn. Neem aan dat $0 \leq$ `i` $<$ `m` en $0 \leq$ `j` $<$ `n`.
b. (8) Schrijf een C++-functie `int meeste (Zee,i)` die de rij-index `i` oplevert van de rij met de meeste schepen erin, en dat aantal schepen retourneert. Als er meerdere rijen hetzelfde maximum hebben, geef dan die met de kleinste index. In het voorbeeld heeft rij `1` het maximum aantal van 2 schepen.
c. (10) Schrijf een Booleaanse C++-functie `plaats (Zee,i,j,k)` die bepaalt of en waar een *verticaal* schip van lengte `k` in `Zee` kan worden geplaatst zodat de schepen niet overlappen en ook niet direct aan elkaar grenzen in orthogonale richting. Gebruik de functie van **a**. Indien het schip kan worden geplaatst, moet een geldige positie (bovenste coördinaten) in de parameters `i` en `j` worden teruggegeven (anders beide -1).

4. (25 punten) Gegeven is het volgende type:

```
class object { public: int info; object* zig; object* zag; };
```

Hiermee worden twee lijsten met objecten (voorzien van een `info`) opgebouwd. De `zag`-pointer wijst naar het volgende element in de lijst (als dat niet bestaat: `nullptr`). De `zig`-pointer wijst naar een element in de andere lijst, voor de bovenste elementen naar het element er direct onder in de onderste lijst (als dat er is); voor de onderste elementen naar het element rechtsboven. Een voorbeeld (`ingang` van type `object*`):



a. (5) Schrijf een C++-functie `voegtoe (ingang,info1,info2)` die een nieuw `object` met `info1` vooraan de lijst toevoegt, met een *nieuw* object "als `zig`", met `info2`.
b. (5) Schrijf een C++-functie `verwijder (ingang)` die het eerste `object` uit de lijst (met `ingang` van type `object*` als ingang) verwijdert, mits dat er is. Denk dus aan de lege lijst. Gooi ook het eventuele "`zig`-object" weg.
c. (5) Schrijf een C++-functie `ruil (ingang)` die de door `ingang` en `ingang->zig` aangewezen *objecten* verwisselt indien zij bestaan en de som van hun `info`-velden deelbaar is door 3.
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. (7) Schrijf een C++-functie `void repareer (ingang,w)` die alle eventuele gaten (zoals onder 9 in het voorbeeld) vult met een object met waarde `w`.