Tentamen Programmeermethoden
Vrijdag 23 januari 2026
13:00–16:00 uur Informatica

Universiteit Leiden
The Netherlands

Bij alle functies moeten de variabelen (constanten eventueel uitgezonderd) in de heading of lokaal 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` ≥ 1) gehele getallen.
a. (4) Schrijf een C++-functie `int aantal (A,n,L,H)` die bepaalt hoe veel getallen zich in het bereik van `L` tot en met `H` bevinden (grenzen inbegrepen).
b. (5) Schrijf een C++-functie `niet_uniek (A,n)` die alle getallen die *niet* meerdere keren voorkomen vervangt door 0. Gebruik de functie van **a**. Zo moet `2 5 3 2 2 4 5 6` worden: `2 5 0 2 2 0 5 0`.
c. (3) Hoeveel vergelijkingen met elementen in de array doet de functie van **b** precies in het slechtste geval uitgedrukt in `n`, en wanneer is dit?
d. (5) Stel dat het array `A` de tientallige representatie voorstelt van een niet-negatief geheel getal. Een voorbeeld: het getal 5273 wordt gerepresenteerd als `5 2 7 3 -1`. Schrijf een C++-functie `int getal (A,n)` die het getal oplevert dat door `A` wordt voorgesteld. Stop zodra je een negatief array-element tegenkomt. Neem aan dat het getal $\leq$ `INT_MAX` is.
e. (8) Schrijf een C++-functie `int langste (A,n,s)` die de lengte uitrekent van de langste dalende aaneengesloten deelrij in het array `A`. Hierbij moet `s` de som van de betreffende deelrij worden. Voor het array `1 7 1 2 3 7 6 3` heeft deelrij `7 6 3` lengte 3 met som 16. Indien er meerdere deelrijen dezelfde maximale lengte realiseren, geef de laatste deelrij.
2. (25 punten) **a**. (6) Bij een functie kun je te maken hebben met *call by value* en *call by reference*, en ook met *lokale* en *globale* variabelen. Verder onderscheiden we ook nog *formele* en *actuele* parameters. Leg deze zes begrippen duidelijk uit.
b. (8) Gegeven een C++-programma met daarin de volgende twee functies:

```
int lisa (bool ok, int x);  // prototype
int bart (bool ok, int x) {
  if ( ! ok ) x = lisa (ok,x); else {
    for ( y = 1; y < x; y++ ) { x = x - y; z += 1000; } }
  cout << x << ", " << y << " en " << z << endl; return x; x= 42;
}//bart
int lisa (bool ok, int x) {
  int y = x + 1; if ( ok ) x = bart (ok,x); else x = x + y;
  cout << x << ", " << y << " en " << z << endl; return x;
}//lisa
```

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

```
x = 1; y = 5; z = 26;
z = lisa (true,y); cout << x << ", " << y << " en " << z << endl;
```

c. (7) Als **b**, maar nu met een `&` ("ampersand") bij de drie voorkomens van parameter `x`.
d. (4) Mag een statement als `x = bart (lisa (x-y,y-z),lisa (z-x,x));` ergens in het `main` programma staan? Onderscheid gevallen met en zonder `&`. Is er sprake van recursie?

3. (25 punten) Gegeven is een `m` bij `n` (beide `const int` > 0; ze hoeven bij deze opgave niet te worden doorgegeven als parameter) Booleaans array `Z` met "schepen" op zee. Hierbij staat `T` voor `true` (een schip op die plek) en `F` voor `false`. Een voorbeeld met `m` $= 4$ en `n` $= 5$ staat hiernaast. Er ligt bijvoorbeeld een schip op $(3, 2)$.

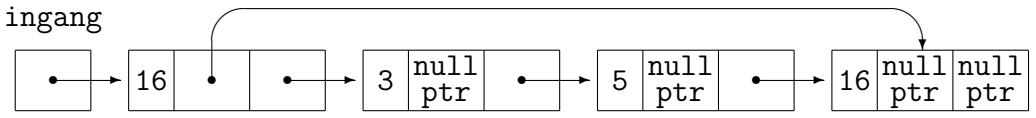
```
T T F T F
T F T T F
T T F F F
F T T T F
```

a. (7) Schrijf een C++-functie `tel (Z,i)` die telt hoe vaak een `F` direct tussen twee `T`'s staat in rij `i`, met $0 \leq$ `i` $<$ `m`. Voor `i` $= 1$ is dit 1.
b. (8) Schrijf een C++-functie `leeg (Z)` die bepaalt welke kolom in array `Z` leeg is, dat wil zeggen geheel uit `F`'s bestaat. Stop zodra je een lege kolom hebt gevonden. Of -1 indien er geen lege kolom is. In het voorbeeld: kolom 4.
c. (10) Schrijf een recursieve C++-functie `raak (Z,i,j)` die berekent hoeveel schepen er liggen in het orthogonaal verbonden gebied van `T`'s dat wordt gegeven door startplek (`i`, `j`) (waar zelf ook een schip moet liggen). Zet alle elementen in dat gebied op `F`. Op $(0, 0)$ worden 8 schepen geraakt: $(0, 0), (0, 1), (1, 0), (2, 0), (2, 1), (3, 1), (3, 2)$ en $(3, 3)$. Op $(1, 1)$ worden 0 schepen geraakt.

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

```
class vakje { public: int getal; vakje* verder; vakje* volg; };
```

Hiermee wordt een pointerlijst met vakjes met daarin een `getal` opgebouwd. Het veld `verder` bevat een pointer naar het eerst volgende vakje met hetzelfde getal (`nullptr` als dat er niet is), en `volg` wijst naar het volgende vakje. Een voorbeeld (`ingang` van type `vakje*`):



a. (4) Schrijf een C++-functie `verwijder (ingang)` die het eerste vakje uit de lijst (met `ingang` van type `vakje*` als ingang) verwijdert, indien dat bestaat, en dat diens `getal` niet deelbaar is door 3. Denk ook aan de lege lijst.
b. (5) Schrijf een C++-functie `wissel (ingang)` die de getallen van de eerste twee vakjes (mits deze bestaan) verwisselt. Denk aan het goed zetten van de twee `verder`-pointers. Zorg dat het ook klopt als de getallen gelijk zijn.
c. (6) Schrijf een C++-functie `voegtoe (ingang,gt)` die een nieuw vakje (met waarde `gt`) vooraan de lijst toevoegt. Hierbij moet ook de `verder`-pointer wijzen naar het eerst volgende vakje met hetzelfde getal.
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 `hoeveel (ingang,gt)` die het aantal voorkomens van `gt` in de lijst bepaalt. Maak op een efficiënte manier gebruik van de `verder`-pointers. In het voorbeeld, met `gt` $= 16$, is het antwoord 2.