Tentamen Programmeermethoden
Woensdag 4 januari 2023
13:15–16:15 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` ≥ 1) gehele getallen.
a. (4) Schrijf een efficiënte Booleaanse C++-functie `pos (A,n)` die `true` geeft als alle elementen in array `A` niet negatief zijn (dus ≥ 0), en anders `false`.
b. (3) Schrijf een C++-functie `int abso (x)` die de absolute waarde van `x` uitrekent.
c. (6) Schrijf een C++-functie `int on (A,p,n)` die de *absolute onbalans* in array `A` berekent ten opzichte van index `p` ($0 \leq$ `p` $<$ `n`). De bijdrage van ieder element wordt bepaald door het product van de waarde van dat element en de afstand tot `p`. Bijvoorbeeld in `A` $= 2, 2, 1, 2, 1, 10, 2$ met `p` $= 3$, telt 2 op index 0 voor $(3 - 0) \times 2 = 6$, en 10 op index 5 voor $(3 - 5) \times 10 = -20$. Dan geeft de functie de absolute waarde van -16 terug, dus 16.
d. (7) Schrijf een C++-functie `int best (A,n)` die een `p` berekent met de kleinste absolute onbalans `on (A,p,n)`. Als er keuze is: de kleinste `p`.
e. (5) Geef een C++-functie `sort (A,n)` die met behulp van *bubblesort* het array `A` van groot naar klein sorteert. En hoeveel vergelijkingen tussen array-elementen doet `sort`?

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

```
int lionel (int s, int t) {
  s--; s--; t++; return s + t; return 2023;
}//lionel
int kylian (int v, int w) {
  int j = 8;
  for ( j = 1; j <= w - 4; j++ ) {
    v = v + lionel (w,j); cout << v << "," << w << endl; }//for
  cout << j << endl; y++; return v + w;
}//kylian
```

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 = 4; y = 7; cout << kylian (x,y) << endl; cout << x << "," << y << endl;
```

c. (6) Als **b**, maar nu met een `&` (“ampersand”) bij de vier parameters van de functies.
d. (4) Net als bij **c** voegen we vier `&`’s toe. We vervangen `v = v + lionel (w,j);` door `v = v - lionel (w,j);` in `kylian`. Wat gebeurt er bij

```
x = 5; y = 41; cout << kylian (x,x) << endl; cout << x << "," << y << endl;
```

Zijn er verschillende antwoorden mogelijk, en waarom? Zo ja, geef deze.
e. (3) Mag ergens in de functie `lionel` staan `s = kylian (3 * s,s)`? Onderscheid 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 `M` met "pijlen" (karakters). De pijlen geven aan in welke richting je kunt bewegen in `M`. Een voorbeeld met $\texttt{m} = 4$ en $\texttt{n} = 5$ staat hiernaast. Hier kan je in `M[2][3]` alleen omhoog, met `'^'`. En `'<'`, `'>'` en `'v'` betekenen naar links, naar rechts en omlaag.

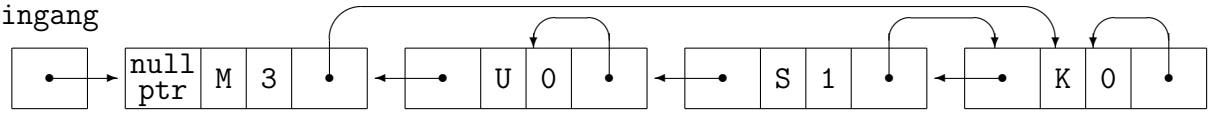
```
< ^ v v v
v v > > v
v ^ > ^ >
> > v < <
```

a. (5) Schrijf een C++-functie `int eind (M)` die berekent op hoeveel plaatsen je `M` kunt verlaten. Bijvoorbeeld op `M[3][2]` verlaat je `M` door naar beneden te gaan. In totaal: 4.
b. (5) We willen soms niet dat twee elementen in `M` direct naar elkaar wijzen: `><` of $\overset{\texttt{v}}{\texttt{\^{}}}$. Schrijf een Booleaanse C++-functie `verboden (M,p,q)` die precies dan `true` teruggeeft als iets dergelijks voorkomt, en anders `false`. Geef in het `true`-geval de coördinaten van één van de betreffende elementen terug via `p` en `q`; in het `false`-geval moeten beide -1 worden. In het voorbeeld `true`, met: 1 en 1 òf 2 en 1.
c. (10) Schrijf een C++-functie `int volg (M,i,j)` die berekent in hoeveel stappen je uit `M` loopt als je op positie $(\texttt{i}, \texttt{j})$ begint. Neem aan dat $0 \leq \texttt{i} < \texttt{m}$ en $0 \leq \texttt{j} < \texttt{n}$. In het voorbeeld voor startpunt $(2, 2)$ is dat 5, via `>`, `^`, `>`, `v` en `>`. Als je er na $\texttt{m} \times \texttt{n}$ stappen niet uit bent, retourneer dan -1.
d. (5) Schrijf een Booleaanse C++-functie `uit (M)` die bepaalt of er een route met minstens 2 stappen is die `M` verlaat. Gebruik de functie van **c**.

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

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

Hiermee wordt een lijst met personen opgebouwd (geheten `naam`). Het veld `vorig` bevat een pointer naar het vorige `persoon`-object, en `volg` wijst naar het $\texttt{nr} \geq 0$ verderop gelegen `persoon`-object. Een voorbeeld (`ingang` van type `persoon*`):



a. (5) Schrijf een C++-functie `verwijder (ingang)` die het eerste `persoon`-object uit de lijst (met `ingang` van type `persoon*` als ingang) verwijdert, mits dat er is en dat diens `nr` 1 is. Denk dus aan de lege lijst. Zet een eventuele `vorig`-pointer ook goed.
b. (5) Schrijf een C++-functie `voegtoe (ingang,name,verder)` die een nieuw `persoon`-object (geheten `name`) vooraan de niet-lege lijst toevoegt. Hierbij moet `nr` gelijk worden aan `verder`, en moet de `volg`-pointer ook goed worden gezet. Neem aan dat $0 \leq \texttt{verder} \leq 1$.
c. (5) Schrijf een C++-functie `wisselen (ingang)` die de namen van de eerste en door diens `volg`-pointer aangewezen `persoon`-objecten verwisselt — indien deze bestaan. In het voorbeeld: `M` $\leftrightarrow$ `K`. (Als de 3 een 0 was: `M` $\leftrightarrow$ `M`.)
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 `repareer (ingang)` die alle `volg`-pointers naar het direct volgende `persoon`-object laat wijzen (`nullptr` bij het laatste object). Neem bij dit onderdeel aan dat alle `nr`-velden > 0 zijn (behalve het laatste, dat 0 is); ze moeten allemaal 1 worden.