NAAM (in drukletters)	VOORNAAM (in drukletters)	Examennummer
Handtekening	**Nummer Studentenkaart**	**Richting** (omcirkel)
		1 BA Informatica 1 BA Wiskunde 1 BA Fysica Andere richting

Examen Beginselen van Programmeren: Deel 2
KU Leuven
26 januari 2015

Richtlijnen

- Schrijf je voornaam, naam, richting, examennummer op **elk** blad dat je afgeeft (ook de opgave). Nummer je bladen.
- Schrijf enkel op **1 zijde** van een blad !
- Ook je **kladbladeren afgeven** ! Vermeld duidelijk dat het kladbladeren zijn !
- Gebruik enkel **papier** dat je van de surveillant krijgt. Geen eigen papier dus !
- Schrijf duidelijk **leesbaar** !
- Zorg voor een duidelijk, **gestructureerd** antwoord !

SPECIFIEK VOOR DEEL 2:

- Dit deel van het examen is "**open boek**".
- Tijdsduur: je krijgt **4 uur** voor het volledige examen. De verwachte tijdsbesteding voor Deel 2 is 3 uur.
- De antwoorden op de vragen van Deel 2 schrijf je op afzonderlijke bladen.

Veel succes !

Marie-Francine Moens
Tom Holvoet

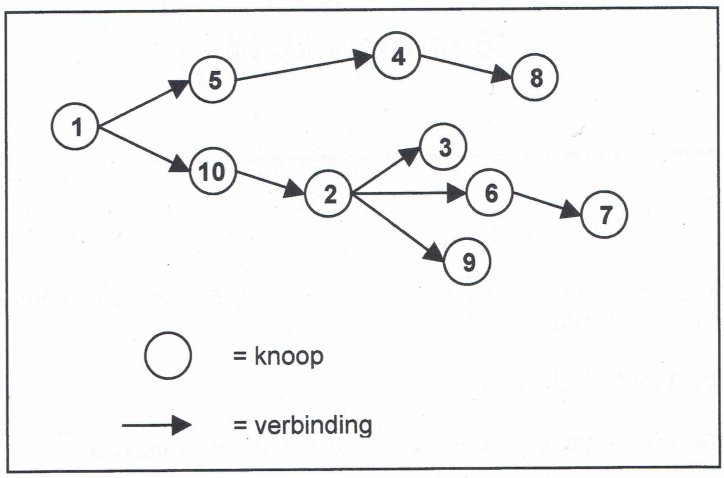
Grafen

Grafen zijn een wiskundig instrument dat vaak aangewend wordt om informatica-problemen te omschrijven en te bestuderen. Voor deze opgave moet een programma worden geschreven dat toelaat om eenvoudige studies uit te voeren op een bepaald soort grafe, nl. "unidirectionele grafen met enkelvoudige voorganger".

Opgave

Grafen

Grafen bestaan uit **knopen** en **verbindingen tussen knopen**. Knopen worden genummerd m.b.v. een geheel getal (dit kan dus ook negatief zijn). Een grafisch voorbeeld maakt de uitleg eenvoudiger.



Dit is een voorbeeld van een grafe met 10 knopen, hier genummerd van 1 t.e.m. 10, en 9 verbindingen, van knoop 1 naar knoop 5, van 1 naar 10, van 5 naar 4, enz.
We noemen een knoop *x* de **voorganger** van knoop *y* indien er een pijl vertrekt in knoop *x* die toekomt in knoop *y*. In dit geval noemen we knoop *y* een **opvolger** van knoop *x*.

Een **pad** tussen twee knopen wordt gedefinieerd als een opeenvolging van knopen waarbij er voor elke 2 opeenvolgende knopen een verbinding bestaat. Voorbeelden zijn:

- een pad tussen knopen 5 en 8, nl. het pad 5 – 4 – 8
- een pad tussen knopen 1 en 6, nl. het pad 1 – 10 – 2 – 6
- een pad tussen knopen 5 en 4, nl. het pad 5 – 4
- enz.

We beschouwen dus enkel grafen die **unidirectioneel** zijn, d.w.z. dat, als er een verbinding is tussen knopen x en y, er *geen* verbinding bestaat tussen *y* en *x*.

Bovendien mag je ervan uitgaan dat, in de grafische voorstelling, in iedere knoop maximaal 1 pijl mag toekomen. Dit garandeert dat er **geen overlap** kan bestaan tussen paden.
Zo zal er in bovenstaande grafe geen verbinding toegevoegd worden die bvb. de knopen 10 en 4 met elkaar verbindt, want dan zouden er 2 paden bestaan, nl. 5 – 4 – 8 en 10 – 4 – 8, die elkaar gedeeltelijk overlappen, nl. de verbinding tussen 4 en 8.

Samengevat kunnen we stellen dat onze grafe bestaat uit knopen en verbindingen; vanuit elke knoop kunnen meerdere pijlen vertrekken, maar in 1 knoop kan slechts 1 pijl toekomen. Of anders gezegd, elke knoop heeft maximaal 1 voorganger maar kan meerdere opvolgers hebben.

Het gewicht van een pad

Met elke *knoop* wordt een **gewicht** geassocieerd (een geheel getal).
Bovendien definiëren we het **gewicht van een *pad*** als de som van de gewichten van de knopen die deel zijn van dit pad.
Onderstel bvb. dat in de grafe in het voorbeeld het gewicht van een knoop overeenkomt met zijn nummer (knoop 1 heeft gewicht 1, knoop 4 heeft gewicht 4, enz.). Dan is bvb. het gewicht van het pad 1 – 10 – 2 – 6 gelijk aan 19.

Verschillende soorten knopen

Typisch (voor deze opgave althans) wordt het gewicht van een knoop echter niet gedefinieerd als het nummer van de knoop, maar steeds als de som van 2 berekeningen:

- een "**voorgangerberekening**", dit is een berekening o.a. op basis van informatie van de voorganger van de knoop;
- een "**opvolgerberekening**", een berekening o.a. op basis van informatie van de opvolgers van de knoop.

We onderscheiden 2 soorten knopen, "**eenvoudige knopen**" en "**complexe knopen**". Deze soorten verschillen enkel in hoe ze deze berekeningen uitvoeren:

- voor een "eenvoudige knoop" geldt het volgende:
  - de *voorgangerberekening* is gelijk aan het product van het nummer van de knoop en het nummer van de voorganger van de knoop (indien die bestaat!);

    $$\text{nummer\_knoop} * \text{nummer\_voorganger}$$

  - de *opvolgerberekening* is gelijk aan de som van het nummer van de knoop en het aantal opvolgers van de knoop

    $$\text{nummer\_knoop} + \text{aantal\_opvolgers}$$

- voor een "complexe knoop" geldt het volgende:
  - de *voorgangerberekening* is gelijk aan de som van het kwadraat van het nummer van de knoop, en het kwadraat van het nummer van de voorganger (indien die bestaat!);

    $$\text{nummer\_knoop}^2 + \text{nummer\_voorganger}^2$$

  - de *opvolgerberekening* is gelijk aan de som van het nummer van de knoop en een som op basis van de nummers van de opvolgers

    $$\text{nummer\_knoop} + \sum_i (\text{nummer\_opvolger}_i - \text{nummer\_knoop})$$

Concreet ...

Schrijf een programma dat van een grafe het pad met het grootste gewicht kan berekenen. Bemerk dat een volledig programma dat met grafen kan werken aan de gebruiker zou vragen om een grafe in te geven, om daarna het pad met het grootste gewicht te laten berekenen. Voor deze opgave moet je de interactie met de gebruiker niet zelf implementeren. Zorg er wel voor dat je functies implementeert die onderstaand hoofdprogramma correct kunnen laten uitvoeren. Voeg ev. aan dit hoordprogramma code toe en/of pas parameters van de functies aan waar nodig.

Definieer daarvoor dus verschillende functies, zoals. een functie om een knoop toe te voegen, een functie om een verbinding toe te voegen, een functie om het gewicht van een pad te berekenen, en een functie om het pad met het grootste gewicht terug te geven. Ook andere functies (e.g. voor voorganger- en opvolger-berekening) kunnen nuttig zijn.

Hint: er zijn 2 manieren om na te gaan of er een pad is tussen knopen *x* en *y*:

- vertrekken vanuit *x* en proberen een pad te vinden via alle mogelijke opvolgers van de knoop tot aan knoop *y*;
- of vertrekken vanuit knoop *y* en nagaan of je tot bij *x* geraakt door telkens de voorganger te volgen.

Hint: implementeer een functie die het volgende doet:

"zoek het pad met grootste gewicht dat aankomt bij een bepaalde knoop, nl. knoop (parameter) y"

en gebruik deze functie voor het implementeren van het zoeken naar het grootste pad in de volledige grafe.

```
def main ()
    voegKnoopToe(1, "eenvoudige")
    voegKnoopToe(2, "complexe")
    voegKnoopToe(3, "eenvoudige")
    voegKnoopToe(4, "eenvoudige")
    voegKnoopToe(5, "complexe")
    voegKnoopToe(6, "complexe")
    voegKnoopToe(7, "eenvoudige")
    voegKnoopToe(8, "complexe")
    voegKnoopToe(9, "complexe")
    voegKnoopToe(10, "eenvoudige")

    voegVerbindingToe(1,5)
    voegVerbindingToe(5,4)
    voegVerbindingToe(4,8)

    voegVerbindingToe(1,10)
    voegVerbindingToe(10,2)
    voegVerbindingToe(2,3)
    voegVerbindingToe(2,6)
    voegVerbindingToe(2,9)
    voegVerbindingToe(6,7)

    pad = geefPadMetGrootsteGewicht()
    schrijfPadUit(pad)

main()
```