Computergebruik: EXAMEN

1e Bachelor Informatica
prof. dr. Peter Dawyndt
groep 1

donderdag 11-01-2018, 08:30
academiejaar 2017-2018
eerste zittijd

Opgave 1 (winterspelen) **(10 pt)**

Van 7-23 februari 2014 organiseerde het Internationaal Olympisch Commité (IOC) haar Olympische Winterspelen in Sotchi, Rusland.

In totaal werden er tijdens die Spelen 98 wedstrijden georganiseerd in 15 verschillende wintersportdisciplines. De wedstrijden vonden plaats in twee clusters van nieuwe locaties: een Olympisch Park gebouwd in Sotsji's Imeretinsky-vallei aan de kust van de Zwarte Zee, met het Olympisch Stadion Fisht en de indoor-locaties van de Spelen op loopafstand, en sneeuwevenementen in het resort van Krasnaya Polyana.
De Olympische Winterspelen van 2014 waren de duurste Olympische Spelen uit de geschiedenis. Hoewel oorspronkelijk begroot op 12 miljard dollar, zorgden verschillende factoren ervoor dat het budget groeide naar 51 miljard dollar, wat meer dan drie keer de kost is van de laatste Olympische Spelen in Londen en ruim boven de geschatte kost van 44 miljard dollar voor de Olympische Zomerspelen van 2008 in Beijing.
De aanloop naar deze Spelen werd gekenmerkt door een aantal grote controverses, waaronder beschuldigingen dat corruptie onder ambtenaren leidde tot bovengenoemde kostenoverschrijdingen, zorgen over de veiligheid van LGBT-atleten en -toeschouwers als gevolg van recente overheidsacties, protesten van etnische Circassian-activisten over het gebruik van een locatie waarvan ze geloven dat er in de 19e eeuw een volkerenmoord plaatsvond, en bedreigingen door jihadistische groepen verbonden aan de opstand in de Noord-Kaukasus. In 2016 bevestigde een onafhankelijk rapport in opdracht van het *World Anti-Doping Agency* (WADA) beschuldigingen dat van ten minste eind 2011 tot augustus 2015 meer dan duizend Russische atleten in verschillende sporten, waaronder zomer-, winter- en Paralympische sporten, profiteerden van een door de overheid georganiseerde doofpotoperatie. In december 2017 besliste het IOC om het Russisch Olympisch Comité te schorsen, met een optie voor schone atleten om als onafhankelijke deel te nemen aan de Olympische Winterspelen van 2018. Het IOC stelde dat het door de Russische staat gesponsorde dopingprogramma

> *[...] één van de ergste aanslagen was tegen de integriteit en de reputatie van de Olympische Spelen.*

Opgave

Het tekstbestand `winterspelen.txt` bevat informatie over alle medailles die uitgereikt werden tijdens de Olympische Winterspelen van 2014. Elke regel van het bestand bevat informatie over één van de uitgereikte medailles. Deze omschrijving bestaat uit de volgende zes velden die van elkaar worden gescheiden door tabs: *i*) wintersportdiscipline, *ii*) wedstrijd, *iii*) doelgroep, *iv*) naam van winnaar(s), *v*) nationaliteit van de winnaar(s) en *vi*) kleur medaille (`goud`, `zilver` of `brons`). Indien er meerdere

medaillewinnaars zijn van een bepaalde kleur, dan worden hun namen (vierde veld) en nationaliteiten (vijfde veld) van elkaar gescheiden door komma's. Voor en na die komma's komen nooit spaties voor. Je mag ervan uitgaan dat de velden geen tabs of puntkomma's bevatten.

Gevraagd wordt om — gebruik makend van de teksteditors `vi` of `vim` — een reeks commando's op te stellen die achtereenvolgens de volgende opdrachten uitvoeren. Probeer voor elke opdracht zo weinig mogelijk commando's te gebruiken en zorg er voor dat elk van deze commando's bestaat uit zo weinig mogelijk tekens. Alle opdrachten moeten na elkaar uitgevoerd worden. De opdrachten mogen de eerste regel niet wijzigen, tenzij dit expliciet anders vermeld staat. Ter controle kan je gebruik maken van de meegeleverde bestanden `winterspelen.`*`i`*`.txt` (ZIP), die telkens de inhoud van het bestand bevatten nadat de i-de opdracht werd uitgevoerd.

1. Vervang alle scheidingstekens (tabs) door puntkomma's en verwijder alle spaties vooraan en achteraan de informatievelden. Deze wijzigingen moeten ook doorgevoerd worden op de eerste regel. Zo moet

```
 sport    onderdeel      groep     naam          land     medaille
Langlaufen      50km vrije stijl         mannen Maksim Vylegsjanin      RUS     zilver
kunstrijden             paren       Aliona Savchenko,Robin Szolkowy     GER     brons
curling         vrouwen Jennifer Jones,Kaitlyn Lawes,Dawn McEwen,Jill Officer,Kirsten Wall      CAN     goud
    Alpineskiën         Super G mannen  Jan Hudec,Bode Miller   CAN,USA    brons
Langlaufen      Sprint  vrouwen Vesna Fabjan     SLO    brons
 kunstrijden                paren         Tatjana Volosozjar,Maksim Trankov     RUS     goud
...
```

   bijvoorbeeld omgezet worden naar (`winterspelen.1.txt`)

```
sport;onderdeel;groep;naam;land;medaille
Langlaufen;50 km vrije stijl;mannen;Maksim Vylegsjanin;RUS;zilver
kunstrijden;;paren;Aliona Savchenko,Robin Szolkowy;GER;brons
curling;;vrouwen;Jennifer Jones,Kaitlyn Lawes,Dawn McEwen,Jill Officer,Kirsten Wall;CAN;goud
Alpineskiën;Super G;mannen;Jan Hudec,Bode Miller;CAN,USA;brons
Langlaufen;Sprint;vrouwen;Vesna Fabjan;SLO;brons
kunstrijden;;paren;Tatjana Volosozjar,Maksim Trankov;RUS;goud
...
```

2. Schrap alle Russische medaillewinnaars uit het bestand. We krijgen dan (`winterspelen.2.txt`)

```
sport;onderdeel;groep;naam;land;medaille
kunstrijden;;paren;Aliona Savchenko,Robin Szolkowy;GER;brons
curling;;vrouwen;Jennifer Jones,Kaitlyn Lawes,Dawn McEwen,Jill Officer,Kirsten Wall;CAN;goud
Alpineskiën;Super G;mannen;Jan Hudec,Bode Miller;CAN,USA;brons
Langlaufen;Sprint;vrouwen;Vesna Fabjan;SLO;brons
snowboarden;Halfpipe;vrouwen;Torah Bright;AUS;zilver
biatlon;10km sprint;mannen;Dominik Landertinger;AUT;zilver
...
```

3. Zet alle wintersportdisciplines (eerste veld) en wedstrijden (tweede veld) om naar kleine letters. We krijgen dan (`winterspelen.3.txt`)

```
sport;onderdeel;groep;naam;land;medaille
kunstrijden;;paren;Aliona Savchenko,Robin Szolkowy;GER;brons
curling;;vrouwen;Jennifer Jones,Kaitlyn Lawes,Dawn McEwen,Jill Officer,Kirsten Wall;CAN;goud
alpineskiën;super g;mannen;Jan Hudec,Bode Miller;CAN,USA;brons
langlaufen;sprint;vrouwen;Vesna Fabjan;SLO;brons
snowboarden;halfpipe;vrouwen;Torah Bright;AUS;zilver
...
```

4. Gebruik het formaat `<`*`eerste naam`*`>` `en` `<`*`tweede naam`*`>` in het veld met de namen van de atleten bij alle medailles die werden uitgereikt aan twee atleten. Gebruik het formaat `<`*`eerste naam`*`>, ...` in het veld met de namen van de atleten bij alle medailles die werden uitgereikt aan meer dan twee atleten. We krijgen dan (`winterspelen.4.txt`)

```
sport;onderdeel;groep;naam;land;medaille
kunstrijden;;paren;Aliona Savchenko en Robin Szolkowy;GER;brons
curling;;vrouwen;Jennifer Jones, ...;CAN;goud
alpineskiën;super g;mannen;Jan Hudec en Bode Miller;CAN,USA;brons
langlaufen;sprint;vrouwen;Vesna Fabjan;SLO;brons
snowboarden;halfpipe;vrouwen;Torah Bright;AUS;zilver
...
```

5. Sorteer de uitgereikte medailles eerst alfabetisch volgens de eerste drie velden (zonder onderscheid te maken tussen hoofdletters en kleine letters), en tenslotte volgens de kleur van de uitgereikte medaille (zesde veld) in de volgorde goud, zilver, brons. We krijgen dan (`winterspelen.5.txt`)

```
sport;onderdeel;groep;naam;land;medaille
alpineskiën;afdaling;mannen;Matthias Mayer;AUT;goud
alpineskiën;afdaling;mannen;Christof Innerhofer;ITA;zilver
alpineskiën;afdaling;mannen;Kjetil Jansrud;NOR;brons
alpineskiën;afdaling;vrouwen;Dominique Gisin en Tina Maze;SUI,SLO;goud
alpineskiën;afdaling;vrouwen;Lara Gut;SUI;brons
alpineskiën;combinatie;mannen;Sandro Viletta;SUI;goud
...
```

Opgave 2 (base64) **(10 pt)**

Base64 is een manier om binaire bestanden te converteren naar tekst bestaande uit ASCII-karakters. Dit gebeurt door binaire data te beschouwen als cijfers uit het talstelsel met grondtal $2^6 = 64$. Hierbij zijn 6 bits nodig om alle verschillende cijfers uit dit talstelsel te kunnen voorstellen. Drie bytes van 8 bits vormen samen 24 bits, en kunnen dus voorgesteld worden door 4×6 bits in de base64 codering. De 64 verschillende karakters die gebruikt worden om de cijfers uit het 64-tallig stelsel voor te stellen, variëren tussen de verschillende implementaties. De algemene strategie is om 64 karakters te gebruiken die voorkomen in de doorsnede van de meeste karaktercoderingen en die ook afdrukbaar zijn. Hierdoor wordt het weinig waarschijnlijk dat de gegevens zullen gewijzigd worden bij uitwisseling tussen verschillende informatiesystemen (bijvoorbeeld email). Zo gebruikt de base64-implementatie van MIME een reeks karakters die bestaat uit de 26 hoofdletters (`A-Z`), de 26 kleine letters (`a-z`), de 10 cijfers (`0-9`), het plusteken (`+`) en de slash (`/`).

WAARDE	KARAKTER	WAARDE	KARAKTER	WAARDE	KARAKTER	WAARDE	KARAKTER
0	A	16	Q	32	g	48	w
1	B	17	R	33	h	49	x
2	C	18	S	34	i	50	y
3	D	19	T	35	j	51	z
4	E	20	U	36	k	52	0
5	F	21	V	37	l	53	1
6	G	22	W	38	m	54	2
7	H	23	X	39	n	55	3
8	I	24	Y	40	o	56	4
9	J	25	Z	41	p	57	5
10	K	26	a	42	q	58	6
11	L	27	b	43	r	59	7
12	M	28	c	44	s	60	8
13	N	29	d	45	t	61	9
14	O	30	e	46	u	62	+
15	P	31	f	47	v	63	/

Opgave

De verzameling $\mathcal{P}$ stelt alle afdrukbare karakters voor: hoofdletters, kleine letters, cijfers en leestekens. Deze verzameling bevat dus geen witruimtekarakters (spatie, tab, newline, ...). Elke regel van het tekstbestand `base64.txt` bestaat uit een reeks karakters uit de verzameling $\mathcal{P}$, gevolgd door een spatie en een woord dat enkel bestaat uit kleine letters. De reeks karakters vóór de spatie noemen we de **codering**. Gevraagd wordt:

1. Bepaal reguliere expressies voor elk van onderstaande verzamelingen. Daarbij staat $\mathcal{C}$ voor de verzameling van alle mogelijke coderingen. Dit zijn met andere woorden alle mogelijke reeksen karakters waarvan de karakters behoren tot de verzameling $\mathcal{P}$.

   (a) $\alpha = \{c \in \mathcal{C} \mid c \text{ bevat karakters die niet voorkomen in de base64-implementatie van MIME}\}$

   voorbeeld: `FR+w0|^{}Z4<bq_,WVT{%xfVxdB<>0JY phis` $\in \alpha$,

(b) $\beta = \{c \in \mathcal{C} \mid c$ bevat vier of meer opeenvolgende cijfers, ingesloten tussen slashes $\}$

voorbeeld: `Q4HXD/7100525/+wgzC5mV2IimhFXh falcate` $\in \beta$,

(c) $\gamma = \{c \in \mathcal{C} \mid$ er is een cijfer dat minstens vier keer voorkomt in $c\}$

voorbeeld: `ZU4mg4I4BAVUzm5lfAbm4r9reE9zyk tarweeds` $\in \gamma$,

(d) de verzameling δ bestaat uit de regels die geselecteerd worden door de volgende twee commando's; beide commando's selecteren exact dezelfde regels

```
# eerste alternatief
egrep -i '(.+).* \1$' base64.txt

# tweede alternatief
rev base64.txt | egrep -i '^(.*) .*\1' | rev
```

i. omschrijf in woorden welke regels door deze commando's geselecteerd worden

ii. leg uit waarom de ene oplossing veel sneller werkt dan de andere

Gebruik een commando uit de `grep` familie om enkel die regels van het bestand `base64.txt` waarvan de codering behoort tot de opgegeven verzameling uit te schrijven naar standaard uitvoer.

Opmerking: Gebruik voor deze opgave een recente versie van GNU `grep`. Op `helios` is een recente versie van GNU `grep` geïnstalleerd, maar Mac OS X gebruikt standaard typisch een oudere versie van GNU `grep`. Mac gebruikers kunnen voor de zekerheid dus best hun `grep` versie updaten naar de meest recente versie.

2. Beschouw de verzamelingen α, β, γ en δ zoals hierboven gedefinieerd. Gebruik nu deze verzamelingen om op de volgende manier een boodschap bestaande uit vier woorden te achterhalen:

(a) het eerste woord staat op de unieke regel met codering uit de verzameling $\alpha \cap \beta$

(b) het tweede woord staat op de unieke regel met codering uit de verzameling $\beta \cap \gamma$

(c) het derde woord staat op de unieke regel met codering uit de verzameling $\gamma \cap \delta$

(d) het vierde woord staat op de unieke regel met codering uit de verzameling $\delta \cap \alpha$

Geef telkens een Unix commando dat elk van deze woorden opzoekt in het bestand en uitschrijft naar standaard uitvoer, zonder evenwel het woord letterlijk uit te schrijven (bv. `echo xxx`).

Opgave 3 (wandelende vingers) **(5 pt)**

Met software zoals Swype of SwiftKey kunnen smartphonegebruikers tekst invoeren door hun vinger over het schermtoetsenbord te slepen in plaats van afzonderlijk op elke letter te drukken.

Opgave

Schrijf een `bash` shell script waaraan als eerste argument een reeks van twee of meer letters moet doorgegeven worden. Aan het shell script moet ook nog een woordenboek doorgegeven worden, dat bestaat uit een lijst van woorden (enkel letters) die allemaal op een afzonderlijke regel staan. De locatie van het woordenboek (een tekstbestand) kan als tweede argument aan het shell script doorgegeven worden. Indien geen tweede argument aan het shell script wordt doorgegeven, dan moet het woordenboek uitgelezen worden uit `stdin`. Het shell script mag ervan uitgaan dat de argumenten die eraan doorgegeven worden geldig zijn, zonder dat dit expliciet moet gecontroleerd worden.
Het shell script moet naar `stdout` een lijst uitschrijven van alle woorden uit het woordenboek die uit minstens vijf letters bestaan en die met de gegeven reeks letters kunnen gevormd worden. Een woord kan gevormd worden uit een reeks letters als de volgende voorwaarden voldaan zijn:

1. het woord begint met de eerste letter uit de reeks
2. het woord eindigt met de laatste letter uit de reeks
3. de letters van het woord vormen een deelreeks van de reeks letters; opeenvolgende gelijke letters in het woord kunnen wel op dezelfde letter uit de reeks afgebeeld worden (het woord `queen` kan bijvoorbeeld gevormd worden met de reeks letters `qxuyezn`, ondanks het feit dat de letter `e` maar één keer voorkomt in de reeks letters)

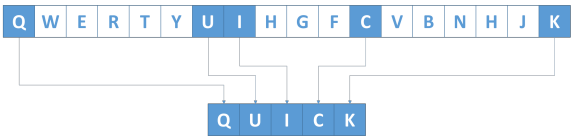
Hierbij wordt geen onderscheid gemaakt tussen hoofdletters en kleine letters. Bij het oplijsten van de woorden moet hun oorspronkelijke volgorde uit het woordenboek aangehouden worden.

Voorbeeld

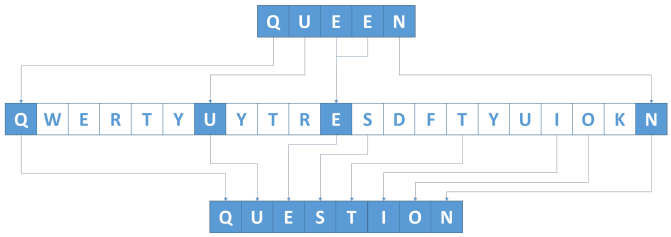
Onderstaande voorbeeldsessie geeft aan hoe het shell script (dat we `wandelingen` genoemd hebben) moet kunnen gebruikt worden. Hierbij gaan we ervan uit dat het tekstbestand `woordenboek.txt` zich in de huidige directory bevindt.

```
$ wandelingen "qwertyuihgfcvbnhjk" < woordenboek.txt
quick
$ wandelingen "qwertyuytresdftyuiokn" < woordenboek.txt
queen
question
$ wandelingen "ghijakjthoijerjidsdfnokg" woordenboek.txt
gaeing
garring
gathering
gating
geeing
gieing
going
goring
```

Voor het eerste voorbeeld uit bovenstaande sessie tonen we hieronder hoe het woord `quick` uit de gegeven reeks letters (bovenaan de figuur) kan gevormd worden.



Voor het tweede voorbeeld uit bovenstaande sessie tonen we hieronder hoe de woorden `queen` en `question` kunnen gevormd worden uit de gegeven reeks letters (in het midden van de figuur). Let hierbij op het feit dat de twee opeenvolgende letters `e` in het woord `queen` beide gebruik maken van dezelfde letter `e` in de reeks letters.



Opgave 4 (overzicht git-status) **(5 pt)**

We gaan ervan uit dat een **git repository** een directory is die kan herkend worden aan de aanwezigheid van een `.git` subdirectory. Geef een Unix commando dat naar `stdout` een overzicht uitschrijft van de status van alle git repositories in de huidige directory en alle onderliggende directories. Voor elke git repository moet het overzicht bestaan uit

- een regel met drie groter dan tekens (`>>>`), een spatie, de absolute padnaam van de git repository, nog een spatie en nog drie kleiner dan tekens (`<<<`)
- de uitvoer naar `stdout` van het commando `git status` voor die git repository

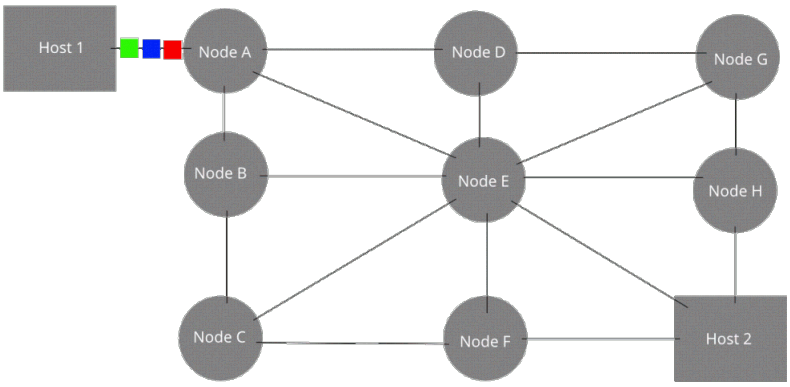
De volgorde waarin de git repositories moeten opgelijst worden, is de volgorde waarin het commando `sort` de absolute padnamen van de git repositories rangschikt. Het resultaat ziet er bijvoorbeeld als volgt uit:

```
>>> /home/runner/workdir/demo/git1 <<<
On branch master
nothing to commit, working directory clean
>>> /home/runner/workdir/demo/git2 <<<
On branch master

Initial commit

nothing to commit (create/copy files and use "git add" to track)
>>> /home/runner/workdir/demo/temp/git3 <<<
On branch master
Untracked files:
(use "git add ..." to include in what will be committed)

run.sh

nothing added to commit but untracked files present (use "git add" to track)
```

Opgave 5 (packet assembler) **(10 pt)**

Wanneer een bestand over het Internet verstuurd wordt, dan wordt het aan de kant van de verzender eerst door een ***splitter*** opgedeeld in kleinere segmenten. Die segmenten worden **pakketten** genoemd. Alle pakketten worden daarna individueel over het Internet verstuurd naar dezelfde ontvanger. Als de ontvanger alle pakketten van het bestand heeft ontvangen, dan gebruikt hij een ***assembler*** om het originele bestand te reconstrueren.

Het Internet gebruikt **packet switching** om de route van de pakketten doorheen het netwerk te bepalen. Daarbij is het vaak wenselijk dat de pakketten van hetzelfde bestand verschillende routes volgen om opstopping te voorkomen. Als gevolg daarvan bereiken die pakketten de ontvanger niet noodzakelijk in hun originele volgorde.



Bovendien is het Internet niet feilloos en gebeurt het wel eens dat pakketten hun bestemming nooit of pas zeer laat bereiken.

Opgave

Schrijf een `bash` shell script `assembler` dat kan gebruikt worden om boodschappen te reconstrueren op basis van binnengekomen pakketten. Aan het shell script moet als argument de locatie van een tekstbestand doorgegeven worden dat alle binnengekomen pakketten bevat. Zo een tekstbestand ziet er bijvoorbeeld als volgt uit:

```
6220    1   10  Because he's the hero Gotham deserves,
6220    9   10
5181    5   7   in time, like tears in rain. Time to die.
6220    3   10  So we'll hunt him.
6220    5   10  Because he's not a hero.
5181    6   7
5181    2   7   shoulder of Orion. I watched C-beams
5181    4   7   Gate. All those moments will be lost
6220    6   10  He's a silent guardian.
5181    3   7   glitter in the dark near the Tannhäuser
6220    7   10  A watchful protector.
5181    1   7   believe. Attack ships on fire off the
6220    0   10  We have to chase him.
5181    0   7   I've seen things you people wouldn't
6220    4   10  Because he can take it.
6220    2   10  but not the one it needs right now.
6220    8   10  A Dark Knight.
```

Elke regel van het tekstbestand stelt één enkel pakket voor en bestaat uit vier informatievelden die van elkaar gescheiden worden door tabs:

1. unieke code $m \in \mathbb{N}$ van de boodschap waaruit het pakket komt
2. volgnummer $i \in \mathbb{N}$ van het pakket; bij het opdelen van een boodschap worden pakketten opeenvolgend genummerd: het eerste pakket heeft volgnummer 0 en het laatste pakket heeft volgnummer $p - 1$
3. aantal pakketten $p \in \mathbb{N}_0$ waarin de boodschap werd opgedeeld
4. tekstfragment (string) dat zelf geen tabs bevat

De pakketten van eenzelfde boodschap komen in willekeurige volgorde voor en het tekstbestand moet niet noodzakelijk alle pakketten van die boodschap bevatten. Hetzelfde pakket komt echter nooit meerdere keren voor.
Daarnaast moet het shell script ook ook de volgende opties ondersteunen:

- optie `-m <int>`: gebruik deze optie om de reconstructie van een binnengekomen boodschap weer te geven; de unieke code van die boodschap moet als verplicht argument aan de optie `-m` doorgegeven worden
- optie `-p`: gebruik deze optie om in het statusoverzicht ook een percentage weer te geven of bij de reconstructie van een boodschap ook de volgnummers van de pakketten weer te geven (zie verder)

Het shell script moet voor de verwerking van de opties de flexibiliteit aan de dag leggen die gebruikelijk is bij Unix commando's: volgorde van opties speelt geen rol, opties kunnen eventueel samengenomen worden, ...
Als de optie `-m` **niet** gebruikt wordt, dan moet het shell script een **statusoverzicht** uitschrijven naar `stdout` dat aangeeft hoe volledig de boodschappen reeds zijn binnengekomen. Voor elke binnengekomen boodschap moet het overzicht een regel bevatten met de unieke code van de boodschap, gevolgd door een dubbelpunt (`:`), een spatie, het aantal pakketten van de boodschap die reeds zijn binnengekomen, een slash (`/`) en het aantal pakketten waarin de boodschap werd opgedeeld. Indien de optie `-p` gebruikt wordt, dan moet op het einde van elke regel ook nog een extra spatie staan, gevolgd door het percentage van het aantal binnengekomen berichten. Het percentage moet tussen ronde haakjes staan en moet afgerond tot op twee cijfers na de komma weergegeven worden. De boodschappen moeten in het overzicht weergegeven worden volgens oplopende unieke code.
Als aan de optie `-m` een argument m wordt doorgegeven, dan moet het shell script voor elk binnengekomen pakket van de boodschap met unieke code m een regel met het tekstfragment van het pakket uitschrijven naar `stdout`. Hierbij moet de originele volgorde van de tekstfragmenten gereconstrueerd worden. Indien de optie `-p` gebruikt wordt, dan moet elk tekstfragment voorafgegaan worden door het volgnummer van het pakket, een punt (`.`) en een spatie.
Het shell script moet de volgende foutafhandeling voorzien:

- indien het shell script niet de gepaste opties meekrijgt (enkel ondersteuning voor de opties `-m` en `-p`), het verplicht argument bij de optie `-m` wordt niet meegegeven, of er wordt niet één enkel argument doorgegeven aan het shell script, dan moet de gepaste boodschap (zie onderstaande voorbeeldsessie) uitgeschreven worden naar `stderr` en moet het shell script eindigen met *exit status* 1
- indien de bestandslocatie die aan het shell script wordt doorgegeven niet verwijst naar een gewoon leesbaar bestand, dan moet de gepaste boodschap (zie onderstaande voorbeeldsessie) uitgeschreven worden naar `stderr` en moet het shell script eindigen met *exit status* 2
- indien het argument dat aan de optie `-m` wordt doorgegeven geen unieke code is van een boodschap waarover het shell script pakketten heeft ontvangen, dan moet de gepaste boodschap (zie onderstaande voorbeeldsessie) uitgeschreven worden naar `stderr` en moet het shell script eindigen met *exit status* 3
- indien het argument dat aan de optie `-m` wordt doorgegeven de unieke code is van een boodschap waarvan het shell script nog niet alle pakketten heeft ontvangen, dan moet de gepaste boodschap (zie onderstaande voorbeeldsessie) uitgeschreven worden naar `stderr` en moet het shell script eindigen met *exit status* 3

Indien er zich geen fouten voordoen, dan moet het shell script eindigen met *exit status* 0.

Voorbeeld

Onderstaande voorbeeldsessie geeft aan hoe het shell script `assembler` moet kunnen gebruikt worden. Hierbij gaan we ervan uit dat de tekstbestanden `packets.01.txt` en `packets.02.txt` zich in de huidige directory bevinden.

```
$ assembler
Syntaxis: assembler [-p] [-m ID] FILE
$ echo $?
1
$ assembler -abc packets.01.txt
Syntaxis: assembler [-p] [-m ID] FILE
$ echo $?
1
$ assembler xxx
assembler: onbestaand of onleesbaar bestand "xxx"
$ echo $?
2
$ assembler packets.01.txt
5181: 7/7
6220: 10/10
$ assembler -p packets.02.txt
1938: 13/17 (76.47%)
2997: 19/19 (100.00%)
6450: 11/11 (100.00%)
7469: 7/7 (100.00%)
9949: 10/10 (100.00%)
$ echo $?
0
$ assembler -m 666 packets.02.txt
assembler: boodschap 666 is onbekend
$ echo $?
3
$ assembler -m 1938 packets.02.txt
assembler: boodschap 1938 is onvolledig
$ echo $?
3
$ assembler -m 7469 packets.02.txt
I've seen things you people wouldn't
believe. Attack ships on fire off the
shoulder of Orion. I watched C-beams
glitter in the dark near the Tannhäuser
Gate. All those moments will be lost
in time, like tears in rain. Time to die.

$ echo $?
0
$ assembler -pm 7469 packets.02.txt
0. I've seen things you people wouldn't
1. believe. Attack ships on fire off the
2. shoulder of Orion. I watched C-beams
3. glitter in the dark near the Tannhäuser
4. Gate. All those moments will be lost
5. in time, like tears in rain. Time to die.
6.
```