NAAM: RICHTING:

Enkele praktische afspraken

- Je krijgt twee uur om deze opdracht **individueel** op te lossen.
- Je raadpleegt **enkel afgedrukte kopies** van de slides (eventueel met handgeschreven nota's) en de ingebouwde manual van SWI-Prolog (gebruik bv `?- help(write).` of `?- apropos(select).`)
- In de map **1617_Gequoteerde/Prolog_Donderdag** op Toledo vind je het bestand `run.pl`. Ook de indienmodule staat daar.
  - Het bestand `run.pl` kan worden gebruikt m.b.v. `swipl -f run.pl` om de voorbeeldqueries uit te voeren. Dit bestand bevat ook de verwachte uitvoer voor deze queries.
  - Als de opdracht expliciet de naam (en ariteit) van een predicaat vermeldt, ben je verplicht om dezelfde naam (en ariteit) te gebruiken in je oplossing.
  - Je oplossing zet je in een bestand `prolog.pl` en de eerste lijnen van dit bestand moeten je naam, studentennummer en richting bevatten.

    ```
    % Jan Jansen
    % r0123456
    % master cw
    ```

  - Na twee uur, of wanneer je klaar ben, dien je het `prolog.pl` bestand in via Toledo.

Gequoteerde zitting Prolog: Ontcijferen

Er bestaan verschillende compressietechnieken om data op een compacte manier voor te stellen. Stel dat je antwoord op een examenvraag inderdaad gecomprimeerd werd en dat enkel de gecomprimeerde versie ervan bij mij geraakt. Dit zou een kleine ramp zijn: hoe recontrueer ik de oorspronkelijke tekst van je antwoord?

Gelukkig weet ik het volgende. Met elk karakter in de oorspronkelijke tekst komt een code overeen die in de gecomprimeerde versie gebruikt wordt. Een code is een sequentie van `0`-en en `1`-en.

Bovendien heeft de gebruikte compressietechniek de eigenschap dat geen enkele code de prefix is van een andere code. Stel dat voor het karakter k de code `[0]` gebruikt wordt, dan zullen de codes voor alle andere karakters in je tekst met een `1` beginnen (`[1,...]`) om verwarring met de code van k te voorkomen.

De frequentie van de individuele karakters in de te comprimeren data bepaalt de toekenning van de codes: karakters met een hoge frequentie komen vaker voor en krijgen liefst de kortste codes.

Om me te helpen bij het ontcijferen, wordt jullie het volgende gevraagd.

Opdracht 1: Schrijf een predicaat `decode(N, Codes, Res)`. Het argument `Codes` is een gegeven lijst en bestaat uit de codes voor de karakters *0*, *1*, .., $N-1$ (in die volgorde). Dit wil zeggen dat de code voor *0* eerst staat, gevolgd door de code van *1* enz. In de lijst `Res` bereken je welke codes er voor alle karakters gebruikt zijn. Hou hierbij rekening met de restrictie dat geen enkele code een prefix mag zijn van een andere code. Hieronder enkele voorbeelden ter verduidelijking.

```
?- decode(3,[1,1,0,1,0], Res).
Res = [0-[1, 1], 1-[0], 2-[1, 0]] ;
No

?- decode(4, [0,0,0,1,1,1,0,1,0], Res).
Res = [0-[0, 0], 1-[0, 1], 2-[1, 1, 0], 3-[1, 0]] ;
Res = [0-[0, 0], 1-[0, 1, 1], 2-[1], 3-[0, 1, 0]] ;
No

?- decode(7,[1,2,0,2,0,1,1,0,0,0,1,0,0,1,2], Res).
Res = [0-[1], 1-[2], 2-[0, 2], 3-[0, 1, 1], 4-[0, 0], 5-[0, 1, 0], 6-[0, 1, 2]];
Res = [0-[1, 2], 1-[0, 2, 0], 2-[1, 1], 3-[0, 0, 0], 4-[1, 0], 5-[0, 1], 6-[2]];
Res = [0-[1, 2], 1-[0, 2, 0], 2-[1, 1, 0], 3-[0, 0], 4-[1, 0], 5-[0, 1], 6-[2]];
No
```

Stel dat je in het eerste voorbeeld voor karakter *0* de code `[1]` gebruikt, dan kan je geen geldige code voor het karakter *1* vinden, want die beginnen allemaal met een `1` (`[1,0]`, of `[1,0,1]`, of ...). Bij de keuze van `[1,1]` voor *0* heb je dat probleem niet. Het kan ook zijn dat de decompressie niet uniek is (zie voorbeelden 2 en 3).

Als er verschillende oplossingen zijn, kan ik gebruik maken van een frequentie-tabel om de beste codering te vinden. De frequentie-tabel geeft de verwachte hoeveelheid voorkomens van de te encoderen informatie. Ik wil voor de gegeven frequenties de korste compressie krijgen. Dit wil zeggen, de geëncodeerde lijst moet zo kort mogelijk zijn voor de gegeven frequenties.

Een frequentie-tabel wordt voorgesteld door een lijst van `Character-Frequence` tuples. Een voorbeeld: `Freqs=[0-3, 1-7, 2-2, 3-1]` specifiëert dat het karakter *0* drie keer zal voorkomen, het karakter *1* zeven keer enz.

Opdracht 2: Schrijf een predicaat `best(N,Codes,Freqs,Res)` waarbij `N` en `Codes` opnieuw gegeven zijn zoals bij het predicaat `decode` en daarnaast bevat de gegeven lijst `Freqs` voor elk getal `i` tussen `0` en `N-1` een `i-f` element waarbij `f` de frequentie is van `i`. Dit predicaat moet voor de lijst van codes `Res` produceren die zorgt voor de kortste compressie. Enkele voorbeelden:

```
?- Freqs=[ 0-3, 1-7, 2-2, 3-1], best(4, [0,0,0,1,1,1,0,1,0],Freqs, Res).
Freqs = [0-3, 1-7, 2-2, 3-1]
Res   = [0-[0, 0], 1-[0, 1], 2-[1, 1, 0], 3-[1, 0]] ;
No


?- Freqs=[ 0-3, 1-7, 2-8, 3-1], best(4, [0,0,0,1,1,1,0,1,0],Freqs, Res).
Freqs = [0-3, 1-7, 2-8, 3-1]
Res   = [0-[0, 0], 1-[0, 1, 1], 2-[1], 3-[0, 1, 0]] ;
No


?- Freqs=[ 0-1, 1-3, 2-3, 3-3], best(4, [0,0,0,1,1,1,0,1,0],Freqs, Res).
Freqs = [0-1, 1-3, 2-3, 3-3],
Res   = [0-[0, 0], 1-[0, 1], 2-[1, 1, 0], 3-[1, 0]] ;

Freqs = [0-1, 1-3, 2-3, 3-3],
Res   = [0-[0, 0], 1-[0, 1, 1], 2-[1], 3-[0, 1, 0]] ;

No
```

Er zijn voor bepaalde instanties meerdere kortste encodings mogelijk (zie voorbeeld 3).