TP1 PROLOG (2h)

Pour définir une base de connaissances, ouvrez une fenêtre éditeur de texte dans laquelle vous écrirez votre code. Enregistrez votre fichier source avec l'extension .pl (ex : *graphe.pl*).

Pour compiler un fichier source en bytecode et le charger dans la base de connaissances de l'interpréteur, tapez (sous l'interpréteur) :

```
consult('graphe.pl').
```

Pour lister toute la base de connaissances :

```
listing.
```

Pour lister un paquet de clauses :

```
listing(arc).
```

Pour interroger la base de connaissances :

```
arc(X,Y).  /* une variable commence par une majuscule */
```

Gprolog affiche la première solution. Tapez *;* pour obtenir la solution suivante, *a* pour toutes les solutions, *rc* pour arrêter.

Lorsqu'on charge la base de connaissances décrite dans un fichier .pl, celle-ci s'ajoute à la base de connaissances de l'interpréteur. Cependant, les prédicats définis dans le fichier chargé écrasent les prédicats de même nom dans la base de connaissances.

Pour sortir de l'interpréteur :

```
halt.  /* ou ctrl d*/
```

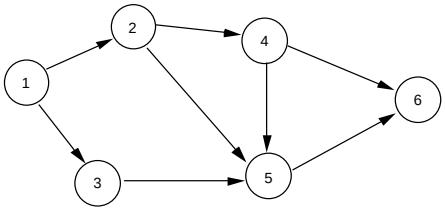
Pour arrêter un programme qui boucle :

ctrl c

Problème n° 1 : Parcours de graphe

Un graphe sera représenté par une base de faits contenant l'ensemble des transitions entre les sommets du graphe. Le prédicat `arc(X, Y)` est vrai s'il y a un arc du sommet `X` vers le sommet `Y`.

1) Modéliser le graphe suivant :



2) Écrire le prédicat `chemin(X, Y)` qui est vrai s'il existe un chemin menant du sommet `X` au sommet `Y` (on suppose que le graphe ne contient pas de cycle).

3) Tester votre programme en posant les questions : `chemin(1, 5)`, `chemin(1, Y)`, `chemin(X, 6)`, `chemin(X, Y)`.

Problème n° 2 : **Manipulation d'ensembles**

Dans cet exercice, on travaille sur des ensembles représentés par des listes. Une liste ne pourra dons pas contenir deux fois le même élément.

Écrire les prédicats suivants :

- `membre(X, Y)` qui est vrai si l'élément `X` appartient à l'ensemble `Y`.
- `inclus(X, Y)` qui est vrai si l'ensemble `X` est inclus dans l'ensemble `Y`.
- `enlever(X, Y, Z)` où `Z` est l'ensemble `Y` privé de l'élément `X`.
- *`egal_ens(X, Y)`* qui est vrai si `X` et `Y` sont des ensembles égaux.
- `intersection(X, Y, Z)` où `Z` est l'intersection de `X` et de `Y`.
- `disjoint(X, Y)` qui est vrai si `X` et `Y` sont des ensembles disjoints.

Le prédicat prédéfini `non(P)`, qui est vrai si `P` n'est pas démontrable peut être utilisé. Ce prédicat est défini dans le fichier « *familleKennedy3.pl* » fourni à la section 2.1.