

Notes :

- Tous les documents sont interdits
- Les exercices sont indépendants.

Exercice 1 – Le code mystérieux (5 pts)

Indiquez les erreurs dans le code suivant en donnant le numéro de ligne et la correction correspondant à chaque erreur. Le symbole □ pourra être utilisé pour représenter un caractère espace et si des lignes sont manquantes, vous pourrez les ajouter en indiquant après quelle ligne elles doivent être insérées en donnant le numéro de la ligne précédente suivi du caractère i. Par exemple, pour insérer une ligne après la ligne 01, vous écririez :

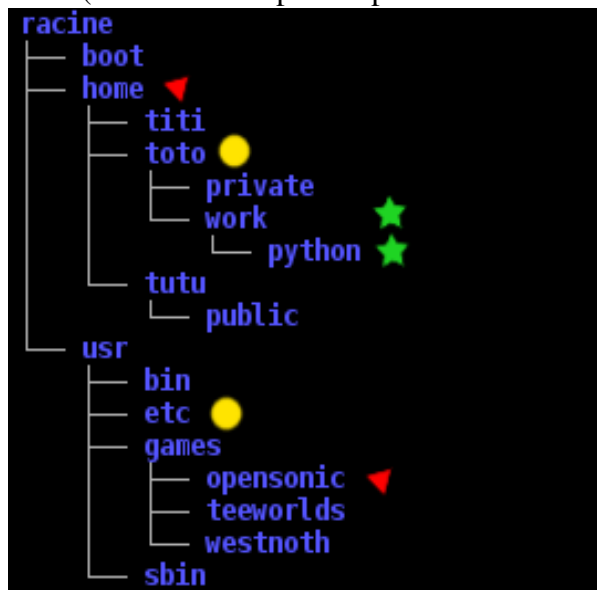
```
01i : □□i = 2
```

Enfin, expliquez ce que ce code est censé faire.

```
01 : def zglub(zouix, pouf)
02 :     j = 1
03 :     if (pouf == "stop")
04 :         exit()
05 :     while (j <= zouix)
06 :         print pouf . "\n"
07 :
08 : def piouc()
09 :     zgra = input_raw("XXXXXXXXXXXXXXXXXXXX : ")
10 :     zglub 10, zgra
11 :     piouc()
12 :
13 : piouc()
```

Exercice 2 – Chemins relatifs et absolus (5 pts)

Le schéma suivant représente l'arborescence d'un système Linux. Pour chaque couple de formes identiques, indiquez la commande shell permettant de se déplacer d'un répertoire à l'autre en utilisant des chemins relatifs et absolus (4 commandes par couple de formes et la racine se note /).





Exercice 3 – Un peu de shell

On souhaite manipuler des dates depuis le shell grâce à la commande `date` dont voici un extrait de la page de manuel :

DATE(1)	User Commands	DATE(1)
NAME	<code>date - print or set the system date and time</code>	
SYNOPSIS	<code>date [OPTION]... [+FORMAT]</code> <code>date [-u --utc --universal] [MMDDhhmm[[CC]YY][.ss]]</code>	
DESCRIPTION	Display the current time in the given FORMAT, or set the system date. <code>-d, --date=STRING</code> display time described by STRING, not `now' <code>-R, --rfc-2822</code> output date and time in RFC 2822 format. Example: Mon, 07 Aug 2006 12:34:56 -0600 <code>-u, --utc, --universal</code> print or set Coordinated Universal Time <code>--help</code> display this help and exit <code>--version</code> output version information and exit FORMAT controls the output. Interpreted sequences are: <code>%%</code> a literal % <code>%a</code> locale's abbreviated weekday name (e.g., Sun) <code>%A</code> locale's full weekday name (e.g., Sunday) <code>%b</code> locale's abbreviated month name (e.g., Jan) <code>%B</code> locale's full month name (e.g., January) <code>%c</code> locale's date and time (e.g., Thu Mar 3 23:05:25 2005) <code>%C</code> century; like %Y, except omit last two digits (e.g., 20) <code>%d</code> day of month (e.g., 01) <code>%D</code> date; same as %m/%d/%y <code>%e</code> day of month, space padded; same as %_d <code>%F</code> full date; same as %Y-%m-%d <code>%g</code> last two digits of year of ISO week number (see %G)	



```
%y    last two digits of year (00..99)
%Y    year
%z    +hhmm numeric timezone (e.g., -0400)
%:z   +hh:mm numeric timezone (e.g., -04:00)
%::z  +hh:mm:ss numeric time zone (e.g., -04:00:00)
%:::z numeric time zone with : to necessary precision (e.g., -04, +05:30)
%Z    alphabetic time zone abbreviation (e.g., EDT)
```

By default, date pads numeric fields with zeroes. The following optional flags may follow `%'` :

```
-      (hyphen) do not pad the field
_      (underscore) pad with spaces
0      (zero) pad with zeros
^      use upper case if possible
#      use opposite case if possible
```

After any flags comes an optional field width, as a decimal number; then an optional modifier, which is either `E` to use the locale's alternate representations if available, or `O` to use the locale's alternate numeric symbols if available.

1. Donnez la commande permettant de connaître la version de la commande
2. Donnez la commande permettant d'afficher la date sous la forme 07/06/2010
3. Donnez la commande permettant d'afficher la date sous la forme 7/6/2010
4. Donnez la commande permettant d'afficher la date sous la forme LUNDI 7 JUIN 2010 (en majuscules !)

Exercice 4 – Palindromes (5 pts)

Un palindrome est une phrase qui peut être lue à l'envers en conservant le même sens (on fait abstraction de la position des espaces). Par exemple, la phrase « esope reste ici et se repose » est un palindrome.

Vous allez écrire un détecteur de palindromes.

1. Écrire une fonction qui étant donnée une phrase passée en paramètre renvoie cette phrase en ayant supprimé tous les espaces.
2. Écrire une fonction qui renvoie l'inverse de la phrase qui lui est passée en paramètre
3. Écrire une fonction *palindrome* qui renvoie la valeur booléenne « vrai » si la phrase passée en paramètre est un palindrome.
4. Donnez le code du programme final demandant à un utilisateur de saisir une phrase et indiquant s'il s'agit ou non d'un palindrome.