Examen Shell année 2009-2010

Les exercices sont indépendants. La correction tiendra compte de la qualité de la rédaction et de la présentation. Le barême est donné à titre indicatif.
Tout document interdit.

⋆ Exercice 1. Questions diverses. (4.5 pts)

▷ **Question 1.** Quelle est la différence entre les symboles `|` et `>` ? Donnez un exemple d'utilisation de chacun de ces symboles.

▷ **Question 2.** Quelle est la différence entre les quotes et les backquotes en shell ? Illustrez votre réponse par un exemple.

▷ **Question 3.** Ecrire une commande équivalente à `cp fich1.c fich2.c`.

▷ **Question 4.** On considère que dans le répertoire courant, un fichier nommé `groupes.txt` est un fichier texte contenant le mot `projet`. Parmi les commandes suivantes, lesquelles sont correctes ? Quel est le résultat affiché par celles qui sont correctes ? **Justifiez vos réponses.**

- `cat groupes.txt | echo`
- `cat groupes.txt > echo`
- `echo projet | groupes.txt`
- `echo projet > groupes.txt`

▷ **Question 5.** Quel est l'effet des commandes suivantes :

```
1. find . -print
2. find . -name '*.c' -print
```

▷ **Question 6.** Quelle est la différence entre les deux commandes suivantes :

```
1. ls -l *.tex
2. find . -name '*.tex' -print
```

▷ **Question 7.** Ecrire une commande qui trouve tous les fichiers `a.out` ou se terminant par `.o` dans l'arborescence et qui les supprime après confirmation.

▷ **Question 8.** On suppose que la commande **date** affiche les informations suivantes :

```
lundi 3 mai 2010, 09:14:30 (UTC+0200)
```

Ecrire une commande qui permette d'afficher uniquement le mois (ex : mai).

▷ **Question 9.** La commande suivante est-elle correcte ? Si oui, expliquez ce qu'elle fait, si non, proposer une solution corrigée.

```
grep 'l\'examen' doc.txt
```

▷ **Question 10.** Que réalise la commande suivante ?

```
sed 's/\([a-z]*\)\[[0-9]*\)/\2 \1/'
```

⋆ Exercice 2. Organisation du planning des soutenances de projet CSH. (2.5 pts)

On dispose d'un fichier `binomes.csv` dont voici un extrait :

```
"KAMGAING, BOINNOT";G3
"LORENZI,KOMUSSIDI";"G3,G5"
"KLEIN,TRUCHI";"G3,G5"
"NGUYEN, SOMBE";"G2,G4"
"BOULDOIRE, DIEZ";"G4,G2"
"BOHR, BUZEAU","G1,G5"
"FOULON, GIRARDIN";"G2,G4"
"RIGAULT, BEN FAIDA";G2
"GUERNIER, ROUSSET";G3
"DOMARIN, LARROQUE";G3
"PRUVOST, VITALE";G2
"BARONE,CAMANINI";G2
```

Chaque ligne indique le nom des élèves et leur(s) groupe(s).

Donnez une commande permettant, en une seule ligne, de :

1. compter combien de binômes ont au moins un membre dans le groupe 2
2. compter combien de binômes ont au moins un membre dans le groupe 4 mais aucun membre dans le groupe 2
3. donner les noms des binômes dont au moins un membre est du groupe 3 (sans enlever les double quotes)
4. la même chose, en enlevant les caractères double quotes
5. donner les noms des binômes qui n'appartiennent qu'à un seul groupe

⋆ Exercice 3. Gestion de versions. (2.5 pts)

Vous travaillez sur votre ordinateur portable qui est déconnecté du réseau. Vous êtes dans une copie de travail Subversion, dont le dépôt est sur une machine actuellement inaccessible.
Parmi les commandes suivantes de Subversion, indiquez celles qui ne fonctionneront pas :
`add log revert status update commit`

Vous avez un autre projet sur votre ordinateur portable, celui-ci versionné par Darcs. Là encore, le dépôt principal est sur une autre machine, inaccessiblepour le moment.
Parmi les commandes suivantes de Darcs, indiquez celles qui ne fonctionneront pas :
`add changes revert whatsnew record push pull`

⋆ Exercice 4. Correction d'un projet de compilation.(3.5 pts)

On demande à des élèves de programmer un compilateur. Ce compilateur doit entre autres signaler un certain nombres de `WARNING` dans des cas précis.
Pour évaluer ces compilateurs, on met en place une suite de tests : on crée un fichier source pour chaque cas de `WARNING` que doit prendre en compte un compilateur. Ces fichiers sources sont dans le répertoire `warnings`.
On écrit un script qui prend en argument un exécutable et qui le teste sur chaque fichiere source du répertoire `warnings`. On vérifie alors que l'exécutable affiche bien `WARNING` lors de la compilation de chaque fichier source. Ce script enregistre le nom des fichiers sources pour lesquels l'exécutable n'a pas signale de `WARNING`, et attribue une note en fonction du nombre de test réussis.
Le script contient des erreurs : chaque ligne où apparait le commentaire `# ERREUR` nécessite une correction, indiquez-les (il y en a 7).

```
#/bin/sh      # ERREUR 1

# tester si le bon nombre d'arguments a été donné

if [ $* != 1 ]     # ERREUR 2
then
 echo "donner le nom de l'executable en argument"
 exit 1
fi

# récupérer la liste des fichiers sources, et créer un fichier pour
# stocker les résultats

note = 0                 # ERREUR 3
fichiers="ls warnings"   # ERREUR 4
rm -f $1.rapport
touch $1.rapport

# pour chaque fichier...
for f in $fichiers
do

 # exécuter l'exécutable dessus et récupére la sortie

 ./$1 warnings/$f > output

 # si la sortie affiche un WARNING incrémenter la note

 grep  -q "WARNING" output
 if [ $? -eq 0 ]
 then
  note=‘expr $note + 1‘
 else

  # si erreur, indiquer le nom du fichier et recopier
  # la sortie du programme dans le rapport

  echo "Erreur avec le fichier $f" >> $1.rapport
  echo output >> $1.rapport           # ERREUR 5
 fi
done

# si le rapport ne contient pas d'erreurs, afficher "parfait"

grep -q "^Erreur$" $1.rapport     # ERREUR 6
if [ $? -eq 1 ]
then
 echo "Parfait !" >> $1.rapport
fi

# ajouter la note de l'exécutable dans le rapport

echo note >> $1.rapport    # ERREUR 7

# nettoyage

rm output
```

⋆ Exercice 5. (Petit) Script shell. (3 pts)

Ecrivez un script shell qui réalise la compilation d'un fichier source C dont le nom est passé en argument, sans extension. Cette commande devra vérifier,

1. que le fichier existe,
2. qu'il correspond bien à un fichier source écrit en langage C (on supposera pour cela qu'un fichier C contient la fonction `main()`).

⋆ Exercice 6. La commande sed. (3.5 pts)

On rappelle que sous `/home/1A/` se trouve l'ensemble des noms de login de chacun d'entre-vous.

La commande `ls -l` affiche (ici pour l'utilisateur `andrew`) l'ensemble des fichiers et répertoires sous la forme :

```
-rw-r--r-- 1 andrew esial1      867 2010-03-21 14:47 mystere.c
```

Soit le script `examen.sh` suivant :

```
#! /bin/sh
# commande examen.sh
rep=/home/1A/

n=‘ls -l $rep | grep "^-.*" | sed ’s/^\([^ ]*\) *\([0-9]*\) *\([^ ]*\).*/\3/’‘
echo $n
```

▷ **Question 1.** Que fait ce script ? Qu'imprime-t-il ?

▷ **Question 2.** On souhaite modifier ce script afin qu'il supprime tous les fichiers `core` qui se trouveraient sous le *home directory* des étudiants d'Esial1 après avoir affiché sa taille. Le cas échéant, le script indiquera :

```
etudiant andrew : fichier core inexistant
```

Réécrivez le script de telle sorte que :

1. il vérifie que les droits du fichier `core`, s'il existe, sont positionnés de sorte que ce fichier soit entièrement accessible pour le groupe *others*,
2. il affiche la taille de ce fichier `core` s'il existe et le supprime ensuite, ou imprime, le cas échéant le message cité précédemment. On rappelle que la taille d'un fichier est donnée par le 5ième champ d'une ligne retournée par `ls -l` (valeur 867 pour le fichier mystère.c).

Rappel de quelques options de grep.

```
grep  - print lines matching a pattern

 -c, --count
     Suppress normal output; instead print a count of matching  lines
     for  each  input  file.

 -q, --quiet, --silent
     Quiet;  do  not  write  anything  to  standard  output.    Exit
     immediately  with  zero status if any match is found, even if an
     error was detected.

 -v, --invert-match
     Invert the sense of matching, to select non-matching lines.
```