1)

a) Convierta el contenido de un archivo de texto a mayúsculas y guarde el resultado sobre el mismo archivo

tr a-z A-Z < $1 > aux.txt

mv aux.txt $1

b) Reemplace los dígitos del contenido de un archivo por un carácter dado como parámetro.

tr 0-9 $1 < $2 > aux.txt

mv aux.txt $2

c) Implemente el comando “tac”, que genera la inversa de un archivo de texto, es decir, la última línea primero y así sucesivamente

for linea in $(cat $1);

do

tail -n 1 $1 >> aux.txt

sed '$d' $1 >> borrado.txt

mv borrado.txt $1

done

mv aux.txt $1

2) Realice un listado recursivo de los archivos del directorio HOME de un usuario y guarde la información en un archivo

ls -R /home > ejercicio2.txt

3) Liste, uno a la vez, todos los archivos mayores de 100K en el directorio HOME de un usuario. De al usuario la opción de eliminar o comprimir el archivo (usando el comando read), luego proceda a mostrar el siguiente. Escriba en un archivo de log los nombres de todos los archivos eliminados y la fecha. Puede utilizar el comando ls o find.

for archivo in $(find /home -size +100k -type f);

do

echo "¿Desea eliminar o comprimir el archivo? $archivo"

read var

case $var in

eliminar)

nombreCompleto=$(basename "$archivo")

dia=$(date +"%d/%m/%Y")

echo "Nombre del archivo eliminado: $nombreCompleto Fecha de

elminación:$dia" >> ejercicio3.log

rm $archivo

;;

comprimir)

gzip $archivo

;;

esac

done

4) Dada una lista de archivos, escriba un script que basado en el tipo (extensión) de cada uno de ellos (.gz, .bz2, .zip, .tar), invoque automáticamente el comando apropiado para descomprimirlo (gunzip, bunzip2, unzip, tar). Si un archivo no está comprimido, el script debe mostrar un mensaje y continuar con el siguiente archivo.

for archivo in $(find $1 -type f); do

case $archivo in

\*.gz)

gunzip $archivo

echo "El archivo $archivo fue descomprimido con exito"

;;

\*.bz2)

bunzip2 $archivo

echo "El archivo $archivo fue descomprimido con exito"

;;

\*.zip)

unzip $archivo

echo "El archivo $archivo fue descomprimido con exito"

;;

\*.tar)

tar $archivo

echo "El archivo $archivo fue descomprimido con exito"

;;

\*)

echo "El archivo $archivo no era un archivo comprimido"

;;

esac

done

5) Un directorio contiene archivos cuyos nombres poseen mayúsculas, minúsculas y espacios. Escriba un script que convierta todos los nombres de archivos en minúsculas y los espacios en ’\_’. Informe cuántos archivos se renombraron. Nota: puede utilizar el comando tr.

IFS='

'

contador=0

for archivo in $(find $1 -type f); do

nombre\_archivo=$(basename $archivo)

sin\_espacios=${nombre\_archivo// /\_}

sin\_mayusculas=$(echo $sin\_espacios|tr 'A-Z' 'a-z')

ruta\_archivo=$(dirname $archivo)

nueva\_ruta="$ruta\_archivo/$sin\_mayusculas"

if [ $archivo != $nueva\_ruta ]; then

let contador++

fi

mv $archivo $nueva\_ruta

done

echo "Se renombraron $contador archivos"

6) Implemente un script reciba un nombre de directorio, correspondiente al directorio de código fuente de un proyecto Java, y construya los siguientes reportes:

a) Cantidad de clases/interfaces importadas (diferentes) en cada archivo de código fuente

cant = 0

cd $1

for i in $(ls $1); do

cant = $(cat $i | grep -c import)

echo "$i- >" $cant >> ~/Documentos/ejerciciosSO/result.txt

done

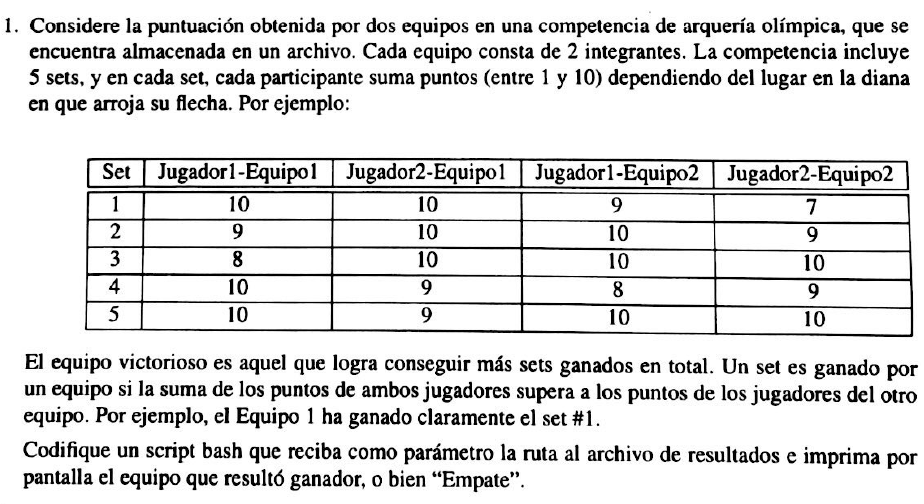
SIN PROBAR

b) no lo hice

7)

8)

**Parcial 2018**



Solucion entendiendo que pide quien gana cada set:

Function Juego {

Datos = $(cat $1) // El $1 hace referencia al parametro pasado

for line in $Datos; do

nroset = $(echo “$line” | cut -d “ “ -f1)

Equipo1Jugador1 = $(echo “$line” | cut -d “ “ -f2)

Equipo1Jugador2 = $(echo “$line” | cut -d “ “ -f3)

Equipo2Jugador1 = $(echo “$line” | cut -d “ “ -f4)

Equipo2Jugador2 = $(echo “$line” | cut -d “ “ -f5)

Equipo1 = $($Equipo1Jugador1 + $Equipo1Jugador2)

Equipo2 = $($Equipo2Jugador1 + $Equipo2Jugador2)

if [$Equipo1 -gt $Equipo2]; then

echo “gana el equipo nro1 en el set $nroset”

elif [$Equipo1 -eq $Equipo2]; then

echo “Hay empate en el setn $nroset”

else

echo “Gana el equipo nro 2 en el set $nroset”

fi

done

}

//PROBAR

Solución entendiendo que pide quien gana el partido:

Function Juego {

Datos = $(cat $1) // El $1 hace referencia al parametro pasado

let PuntosEq1 = 0

let PuntosEq2 = 0

for line in $Datos; do

nroset = $(echo “$line” | cut -d “ “ -f1)

Equipo1Jugador1 = $(echo “$line” | cut -d “ “ -f2)

Equipo1Jugador2 = $(echo “$line” | cut -d “ “ -f3)

Equipo2Jugador1 = $(echo “$line” | cut -d “ “ -f4)

Equipo2Jugador2 = $(echo “$line” | cut -d “ “ -f5)

Equipo1 = $($Equipo1Jugador1 + $Equipo1Jugador2)

Equipo2 = $($Equipo2Jugador1 + $Equipo2Jugador2)

if [$Equipo1 -gt $Equipo2]; then

echo “gana el equipo nro1 en el set $nroset”

let PuntosEq1 ++

elif [$Equipo1 -eq $Equipo2]; then

echo “Hay empate en el setn $nroset”

else

echo “Gana el equipo nro 2 en el set $nroset”

let PuntosEq2 ++

fi

done

if [$PuntosEq1 -gt $PuntosEq2]; then

echo “ganó el equipo 1”

elif [$PuntosEq1 -eq $PuntosEq2]; then

echo “empataron”

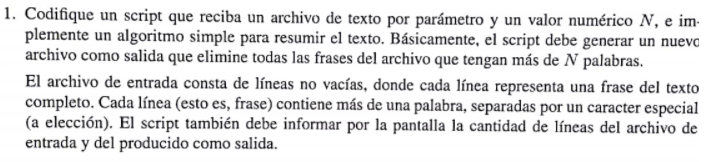
else

echo “ganó el equipo 2”

}

//PROBAR

**Recu 2018**



Function recu {

let cantEntrada = 0 /////// averiguar

let cantSalida = 0

Datos = $(cat $1) // Archivo pasado por parametro

N = $(cat $2) // valor de N pasado por parametro

for line in $Datos; do

cantpalabras = $(echo “$line” | wc -w)

if [$cantpalabras -lt $N]; then

echo $line >> nuevoarchivo.txt

let cantsalida++

fi

let cantentrada++

done

echo “el archivo entrada tiene $cantentrada lineas”

echo “el archivo salida tiene $cantsalida lineas”

}

//PROBAR

**Parcial 2015:**

**Se tiene un archivo con información de ciudades del mundo, con una estructura tabular donde cada fila posee una columna con el nombre de la ciudad, otra con el país, y columnas que guardan la temperatura media histórica de cada mes (en grados Celsius). Escriba un script que dado como parámetro un país retome el nombre y las temperaturas medias de sus ciudades expresadas en grados Farenheit. Considere que F = 1,8 \* C + 32.**

Function 2015{

original = $(cat temperatura.txt)

pais = $(cat $1)

resultado = ‘’

for line in $original; do

paislinea = $(echo “$line” | cut -d ‘ ‘ -f2)

if [$pais = $paislinea]; then

ciudad = $(echo “$line” | cut -d ‘ ‘ -f1)

let aux = 0

for j in {3..15}; do

let aux = aux + (echo “$line” | cut -d ‘ ‘ -f$j)

done

promedio = $( $aux / 12)

let promedioF = $promedio \* 1,8 + 32

resultado = $resultado ‘ ‘ $ciudad ‘ ‘ $promedioF

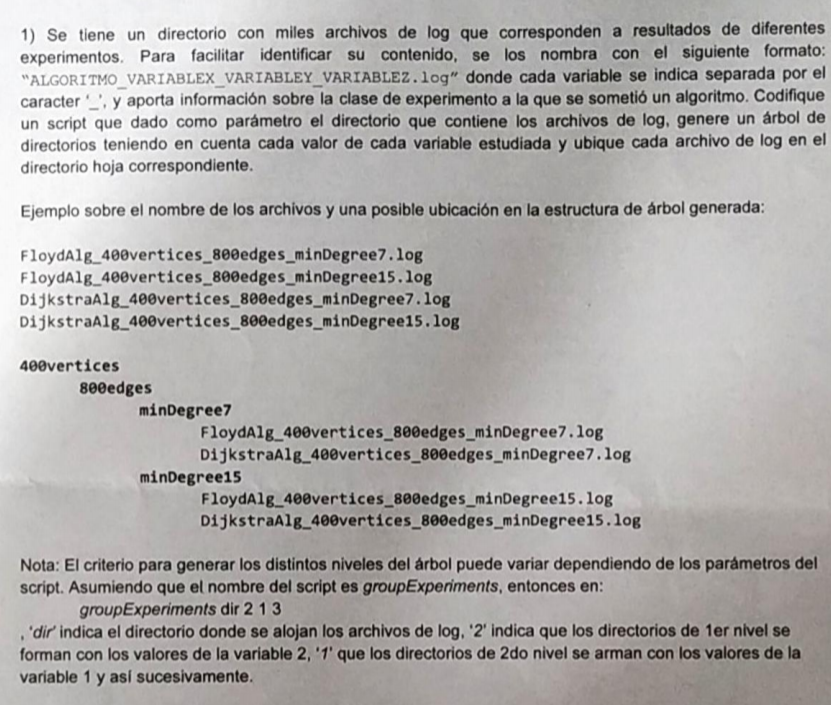
fi

done

echo $resultado > salida.txt

//PROBAR

**Parcial 2019:**

****

for loopfile in $(ls -l $1); do

path=$(echo $loopfile | cut -d '.' -f1 | cut -d '\_' -f$2) #1

path=$path"/"$(echo $loopfile | cut -d '.' -f1 | cut -d '\_' -f$3) #2

path=$path"/"$(echo $loopfile | cut -d '.' -f1 | cut -d '\_' -f$4) #3

mkdir -p $1/ $path #4

mv $1/$loopfile $1/$path #5

done

//PROBAR

#ls es para listar todos los archivos en $1 en donde $1 es el directorio que viene como parametro

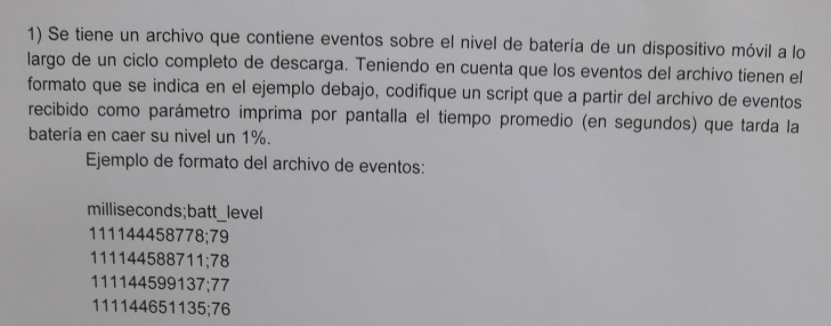
1 #se queda con el nombre del archivo, el d indica lo que separa las columnas y el f$2 indica cual de las 3 columnas es la que tiene que agarrar

2 # hago lo mismo, corto hasta el punto , y despues me quedo con la variable que venga

4#mkdir crea un directorio en la ruta, pero toda la ruta debe existir. -p crea la ruta entera

5#mueve el archivo al path creado

**Recu 2019**



firstsample = $(Head -2 $1 | Tail -1 | cut -d ‘;’ - f1)

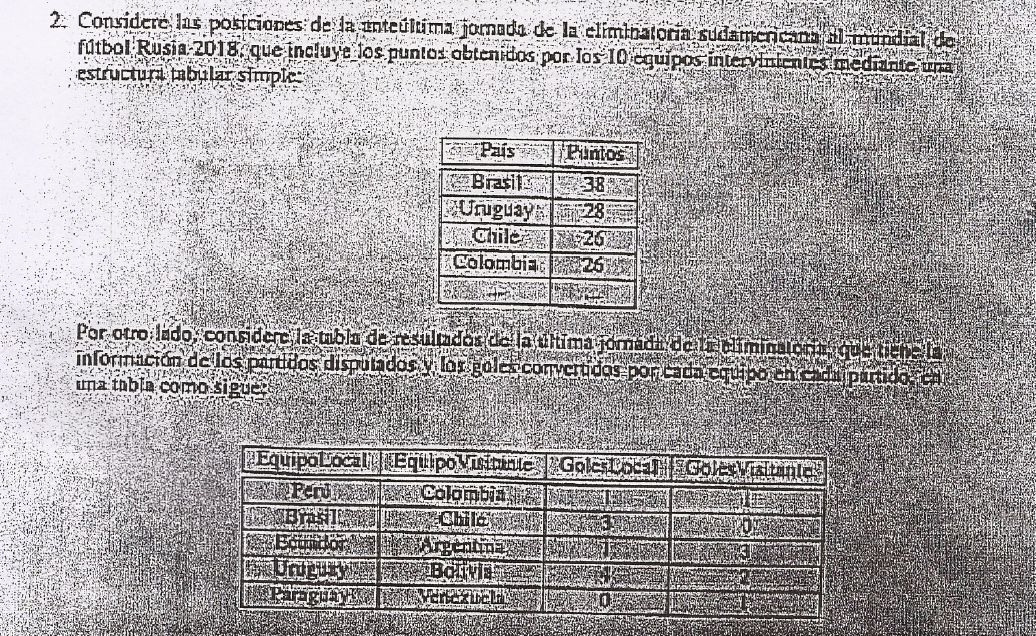
fastsample = $(Tail -1 $1 | cut -d ‘;’ -f1)

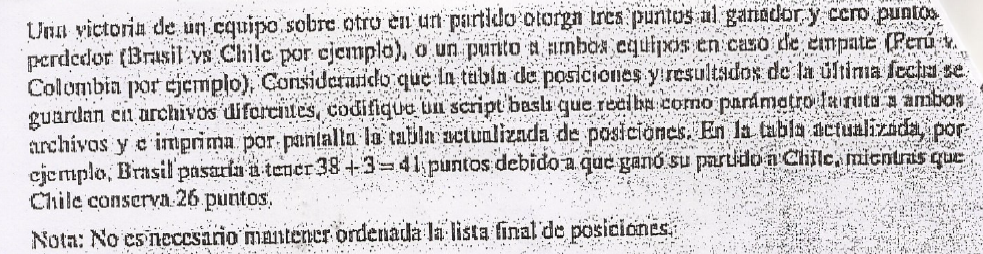
length = $(wc -l $1) o $(cat $1 | wc -l)

echo $(echo “ Scale = 2; ((fastsample - firstsample) / 100) / (length -1)” | bc)

REVISAR O PREGUNTAR ESTO

**Parcial 2017**





function mundial{

Acumulados = $(cat $1)

fecha = $(cat $2)

for line in $fecha; do

local = $(echo $line | cut -d ‘ ‘ -f1)

visitante = $(echo $line | cut -d ‘ ‘ -f2)

goleslocal = $(echo $line | cut -d ‘ ‘ -f3)

golesvisitante = $(echo $line | cut -d ‘ ‘ -f4)

lineaacumuladolocal = $(grep $local $acumulados)

lineaacumuladovisitante = $(grep $visitante $acumulados)

puntoslocal = $(echo $lineaacumuladolocal | cut -d ‘ ‘ -f2)

puntosvisitante = $(echo $lineaacumuladovisitante | cut -d ‘ ‘ -f2)

if [$goleslocal -gt $golesvisitante] then

let puntoslocal = puntoslocal + 3

temp = $local ‘ ‘ $puntoslocal

sed ‘/$local/c $temp’ $acumulados

elif [$goleslocal -lt $golesvisitante] then

let puntosvisitante = puntosvisitante + 3

temp = $visitante ‘ ‘ $visitante

sed ‘/$visitante/c $temp’ $acumulados

else

let puntoslocal ++

let puntosvisitante ++

temp = $local ‘ ‘ $puntoslocal

sed ‘/$local/c $temp’ $acumulados

temp = $visitante ‘ ‘ $visitante

sed ‘/$visitante/c $temp’ $acumulados

fi

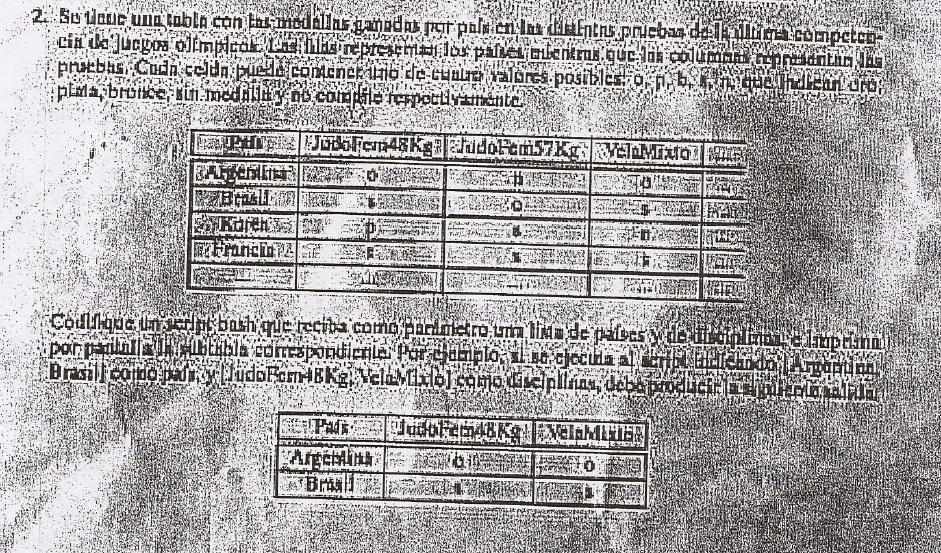
done

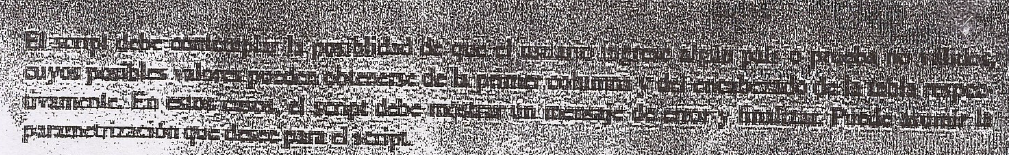
echo $acumulados

}

//PROBAR

**Parcial 2016:**





Fuction juegos{

for pais in $(echo $1 | tr ‘ ’ ‘\n’); do

cantlinea = $(grep -c $pais table.txt)

if [cantlinea -eq 0]; then

echo “ $pais no está en la tabla”

else

echo “pais a imprimir: $pais”

lineap = $(grep $pais tabla.txt)

linead = head -n 1 tabla.txt

resultado = echo ($pais ‘ ’)

disciplinas = $2

for disciplina in $(echo $2 | tr ‘ ‘ ‘\n’);do

let cant = 1

for i in $(echo $d | tr ‘ ‘ ‘\n’);do

if [i -eq $disciplina]

aux = $(echo $linea | cut -d ‘ ‘ -f $cant)

resultado = $resultado ‘ ‘ $aux

fi

let cant ++

done

done

fi

done

echo $resultado >> salida.txt } //PROBAR parametros: archivo paises primero y disciplinas despues.