Ejercicios resueltos de Scripting

Hechos 16.31: Cree en el Señor Jesucristo, y serás salvo, tú y tu casa.

Ejercicio 1: 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
```

Forma de llamar al script: bash nombre_ejercicio1.sh nombre_archivo.extension_archivo
Nota 1: llamando al script con la opción -x se hace un paso a paso de lo que va ocurriendo: bash -x nombre_ejercicio1.sh
Nota 2: apretando Ctrl + c se aborta la ejecución de un script

Ejercicio 2: 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
```

Forma de llamar al script: bash nombre_ejercicio2.sh carácter nombre_archivo.extension_archivo

Ejercicio 3: Implemente el comando “tac”, que genera la inversa de un archivo de texto, es decir, la última línea primero y así sucesivamente.

```
IFS='
'
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
```

Forma de llamar al script: bash nombre_ejercicio3.sh nombre_archivo.extension_archivo

Ejercicio 4: 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
```

Forma de llamar al script: bash nombre_ejercicio4.sh

<u>Ejercicio 5:</u> 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.

```
IFS='
'
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
```

Forma de llamar al script: bash nombre_ejercicio5.sh

<u>Ejercicio 6:</u> 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.

```
IFS='
'
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
```

Forma de llamar al script: bash nombre_ejercicio6.sh ruta_directorio

<u>Ejercicio 7:</u> 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"
```

Forma de llamar al script: bash nombre_ejercicio7.sh ruta_directorio

<u>Ejercicio 8:</u> Muestre el árbol de directorios a partir de un directorio pasado como parámetro, de acuerdo a las siguientes variantes:
a) Mostrando sólo subdirectorios.
b) Mostrando subdirectorios y archivos.
c) Las dos anteriores, tomando como parámetro un valor entero que indica el nivel máximo para la navegación de un directorio/subdirectorio.
El comando debería mostrar por pantalla el árbol de directorios en modo texto, donde el desplazamiento a derecha del nombre de un directorio indica proporcionalmente su profundidad. Por ejemplo, la siguiente salida corresponde al listado del contenido del directorio "/" (los asteriscos indican cada entrada dentro de un subdirectorio):
*** boot**
*** grub**
***...**
*** dev**
*** etc**
*** ...**

a)

```
ricerca () {
for dir in $(find $1 -type d)
do
   if [ -d "$dir" ] ; then
      zz=0
      while [ $zz -ne $1 ]
      do
         echo -n " "
         zz=$(expr $zz + 1)
      done
      if [ -L "$dir" ] ; then
         echo "* $dir" $(ls -l $dir | sed 's/^.*'$dir' //')
      else
         echo "* $dir"
         numdir=$(expr $numdir + 1)
         if cd "$dir" ; then
            ricerca $(expr $1 + 1)
            cd ..
         fi
      fi
   fi
done
}
if [ $# -ne 0 ] ; then
  cd $1
fi
echo "Directorio inicial = $(pwd)"
numdir=0
ricerca 0
```

```
echo "Total directorios = $numdir"
```

Forma de llamar al script: bash nombre_ejercicio8a.sh ruta_directorio

b)

```
ricerca () {
for dir in $(echo *)
do
   zz=0
   while [ $zz -ne $1 ]
   do
      echo -n " "
      zz=$(expr $zz + 1)
   done
   if [ -L "$dir" ] ; then
      echo "* $dir" $(ls -l $dir | sed 's/^.*'$dir' //')
   else
      echo "* $dir"
      numdir=$(expr $numdir + 1)
      if [ -d "$dir" ] ; then
         if cd "$dir" ; then
            ricerca $(expr $1 + 1)
            cd ..
         fi
      fi
   fi
done
}
if [ $# -ne 0 ] ; then
  cd $1
fi
echo "Directorio inicial = $(pwd)"
numdir=0
ricerca 0
echo "Total directorios = $numdir"
```

Forma de llamar al script: bash nombre_ejercicio8b.sh ruta_directorio

c)

```
ricerca () {
for dir in $(echo *)
do
   if [[ $2 -lt $3 ]] ; then
      zz=0
      while [ $zz != $1 ]
      do
         echo -n " "
         zz=$(expr $zz + 1)
      done
      if [ -L "$dir" ] ; then
         echo "* $dir" $(ls -l $dir | sed 's/^.*'$dir' //')
```

```
        else
            echo "* $dir"
            numdir=$(expr $numdir + 1)
            if [ -d "$dir" ] ; then
                if cd "$dir" ; then
                    ricerca $(expr $1 + 1) $(expr $2 + 1) $3
                    cd ..
                fi
            fi
        fi
    fi
done
}
if [ $# != 0 ] ; then
  cd $1
fi
echo "Directorio inicial = $(pwd)"
numdir=0
ricerca 0 0 $2
echo "Total directorios = $numdir"
```

Forma de llamar al script: bash nombre_ejercicio8c.sh ruta_directorio número

<u>Ejercicio 9:</u> Implemente un script de shell que dada una ruta a un archivo de texto como parámetro, imprima por pantalla todas las palabras y su valor TF asociado. El algoritmo TF (Term Frequency) determina la importancia de cada una de las palabras de un documento de texto. Para esto, la importancia de una palabra p se determina contando la cantidad de veces que p aparece en el documento y dividiendo por el número de ocurrencias de la palabra que más se repite en el mismo. Por ejemplo, si un documento contiene p_1 (2 ocurrencias), p_2 (1 ocurrencia) y p_3 (3 ocurrencias), entonces:
$TF(p_1) = 2/3 = 0.66$
$TF(p_2) = 1/3 = 0.33$
$TF(p_3) = 3/3 = 1$

```
palabraMasSeRepite(){
   mayorCantidad=0;
   while read -ra linea;
   do
      for palabra in "${linea[@]}";
      do
         cantidad=$(tr ' ' '\n' < $1 | grep -c $palabra);
         if [ "$cantidad" \> "$mayorCantidad" ] ; then
            mayorCantidad=$cantidad
         fi
      done;
   done < $1
   echo $mayorCantidad
}
mayor=$(palabraMasSeRepite $1)
while read -ra linea;
do
   for palabra in "${linea[@]}";
   do
      cantidad=$(tr ' ' '\n' < $1 | grep -c $palabra);
      TF=$(echo "scale=2; $cantidad/$mayor" | bc)
      echo "TF($palabra)=$cantidad/$mayor=$TF" >> aux.txt
   done;
done < $1
uniq aux.txt
rm aux.txt
```

Forma de llamar al script: bash nombre_ejercicio9.sh nombre_archivo.extension_archivo

<u>Ejercicio 10:</u> Escriba un script de shell que calcule el número Fibonacci de un valor recibido como parámetro, de acuerdo a su definición recursiva:
fib(0)=1
fib(1)=1
fib(n)=fib(n-1)+fib(n-2)
a) El script debe guardar (sin repeticiones) los valores calculados en un archivo intermedio “computed” en el directorio home que asocie un valor individual con su Fibonacci. El script además debe utilizar los valores presentes en dicho archivo para evitar recalcular el Fibonacci de un número ya calculado. No considere tratamiento de errores. Se asume que los parámetros recibidos siempre tienen un valor asociado y del tipo correcto (no negativos).
b) Modifique el script para que calcule el Fibonacci para una lista de valores recibidos como parámetro bajo las mismas condiciones listadas anteriormente.
a)

```
Fibonacci(){
   fibonacci=0
   while read line
   do
      array=($line)
      for i in "${array[@]}"
      do
        valorAux=$(echo $i | cut --complement -d "(" -f1)
        valor=$(echo $valorAux | cut -d ")" -f1)
        if [ "$2" = "$valor" ] ; then
            fibonacci=$(echo $i | cut --complement -d "=" -f1)
        fi
        if [ $fibonacci -ne 0 ] ; then
           break;
        fi
      done
      if [ $fibonacci -ne 0 ] ; then
         break;
      fi
   done < $1
   echo $fibonacci
}
F=$(Fibonacci $1 $2)
if [ $F -eq 0 ] ; then
   for ((i=1; i<=$2; i++)); do
      incluido=$(Fibonacci $1 $i)
      if [ $incluido -eq 0 ] ; then
         n1=$(expr $i - 1)
         n2=$(expr $i - 2)
         Fn1=$(Fibonacci $1 $n1)
         Fn2=$(Fibonacci $1 $n2)
         F=$(expr $Fn1 + $Fn2)
```

```
            echo "Fibonacci($i)=$F" >> computed.txt
        fi
    done
fi
echo "El valor Fibonacci del valor $2 es $F"
```

Forma de llamar al script: bash nombre_ejercicio10a.sh computed.txt número

b)

```
Fibonacci(){
    fibonacci=0
    while read linea
    do
        array=($linea)
        for i in "${array[@]}"
        do
            valorAux=$(echo $i | cut --complement -d "(" -f1)
            valor=$(echo $valorAux | cut -d ")" -f1)
            if [ "$2" = "$valor" ] ; then
                fibonacci=$(echo $i | cut --complement -d "=" -f1)
            fi
            if [ $fibonacci -ne 0 ] ; then
                break;
            fi
        done
        if [ $fibonacci -ne 0 ] ; then
            break;
        fi
    done < $1
    echo $fibonacci
}
while read linea
do
    F=$(Fibonacci $1 $linea)
    if [ $F -eq 0 ] ; then
        for ((i=1; i<=$linea; i++)); do
            incluido=$(Fibonacci $1 $i)
            if [ $incluido -eq 0 ] ; then
                n1=$(expr $i - 1)
                n2=$(expr $i - 2)
                Fn1=$(Fibonacci $1 $n1)
                Fn2=$(Fibonacci $1 $n2)
                F=$(expr $Fn1 + $Fn2)
                echo "Fibonacci($i)=$F" >> computed.txt
            fi
        done
    fi
    echo "El valor Fibonacci del valor $linea es $F"
done < $2
```

Forma de llamar al script: bash nombre_ejercicio10b.sh computed.txt nombre_archivo.extension_archivo

<u>Ejercicio 11:</u> 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 Celcius). Escriba un script que dado como parámetro un país retorne el nombre y las temperaturas medias de sus ciudades expresadas en grados Farenheit. Considere que F = 1,8 * C + 32. (Ejercicio del Parcial 2015).

Formato del archivo para la resolución:

Ciudad	Pais	Temperatura
Necochea	Argentina	20
Tandil	Argentina	23
Ushuahia	Argentina	12
Madrid	España	15
Barcelona	España	24

```
IFS='
'
constante=1.8
cantLineas=$(cat $1 | wc -l)
cantLineas=$(expr $cantLineas - 1)
for linea in $(tail -n $cantLineas $1)
do
    pais=$(echo $linea | cut -d" " -f2)
    if [[ $pais = $2 ]]; then
        ciudad=$(echo $linea | cut -d" " -f1)
        celcius=$(echo $linea | cut -d" " -f3)
        temperatura=$(expr $constante*$celcius | bc)
        farenheit=$(expr $temperatura+32 | bc)
        echo "$ciudad $farenheit"
    fi
done
```

Forma de llamar al script: bash nombre_ejercicio11.sh nombre_archivo.extension_archivo nombre_pais

<u>Ejercicio 12:</u> Implemente un script que reciba como parámetro dos números enteros S y E y un archivo de texto, y que imprima por pantalla todas las líneas del archivo desde S inclusive hasta E inclusive. Por ejemplo, dados S = 3 y E = 5 del siguiente archivo:

text row 1
text row 2
text row 3
text row 4
text row 5
text row 6

el script debe mostrar por pantalla:

text row 3
text row 4
text row 5

Además, el script debe mostrar mensajes de error y abortar la impresión para cubrir “mínimamente” los siguientes tres casos: a) S es menor a 1, b) E es menor a S, c) E supera la cantidad de líneas del archivo. (Ejercicio del Parcialito 2018).

```
numeroLinea=1
S=$2
E=$3
cantidadLineasArchivo=$(cat $1 | wc -l)
if (( $S < 1 )) ; then
   echo "S es menor a 1"
else
   if (( $E < $S )) ; then
      echo "E es menor a S"
   else
      if (( $E > $cantidadLineasArchivo )) ; then
         echo "E supera la cantidad de lineas del archivo"
      else
IFS='
'
for linea in $(cat $1) ; do
   if (( $numeroLinea >= $S )) && (( $numeroLinea <= $E)) ; then
      echo $linea
   fi
   numeroLinea=$(expr $numeroLinea + 1)
   if (( $numeroLinea > $E )) ; then
      break;
   fi
done
      fi
```

```
    fi
fi
```

Forma de llamar al script: bash nombre_ejercicio12.sh nombre_archivo.extension_archivo

<u>Ejercicio 13:</u> Considere la puntuación obtenida por dos equipos en una competencia de arquería olímpica, que se encuentra almacenada en un archivo. Cada equipo consta de 2 integrantes. La competencia incluye 5 sets, y en cada set, cada participante suma puntos (entre 1 y 10) dependiendo del lugar en la diana en que arroja su flecha. Por ejemplo:

Set	Jugador1-Equipo1	Jugador2-Equipo1	Jugador1-Equipo2	Jugador2-Equipo2
1	10	10	9	7
2	9	10	10	9
3	8	10	10	10
4	10	9	8	9
5	10	9	10	10

El equipo victorioso es aquel que logra conseguir más sets ganados en total. Un set es ganado por un equipo si la suma de los puntos de ambos jugadores supera a los puntos de los jugadores del otro equipo. Por ejemplo, el Equipo 1 ha ganado claramente el set #1.

Codifique un script bash que reciba como parámetro la ruta al archivo de resultados e imprima por pantalla el equipo que resultó ganador, o bien “Empate”. (Ejercicio del Parcial 2018).

Formato del archivo para la resolución:

10	10	9	7
9	10	10	10
8	10	10	10
10	9	8	9
10	9	10	10

```
setGanadosEquipoUno=0
setGanadosEquipoDos=0
IFS='
'
for linea in $(cat $1)
do
   puntosEquipoUno=0
   puntosEquipoDos=0
   JugadorUnoEquipoUno=$(echo $linea  | cut -d" " -f1)
   JugadorDosEquipoUno=$(echo $linea  | cut -d" " -f2)
   JugadorUnoEquipoDos=$(echo $linea  | cut -d" " -f3)
   JugadorDosEquipoDos=$(echo $linea  | cut -d" " -f4)
   puntosEquipoUno=$(expr $JugadorUnoEquipoUno + $JugadorDosEquipoUno)
   puntosEquipoDos=$(expr $JugadorUnoEquipoDos + $JugadorDosEquipoDos)
   if [ $puntosEquipoUno -gt $puntosEquipoDos ] ; then
      setGanadosEquipoUno=$(expr $setGanadosEquipoUno + 1)
   else
      if [ $puntosEquipoDos -gt $puntosEquipoUno ] ; then
        setGanadosEquipoDos=$(expr $setGanadosEquipoDos + 1)
      fi
   fi
done
if [ $setGanadosEquipoUno -gt $setGanadosEquipoDos ] ; then
    echo "El equipo ganador es el equipo uno"
else
    if [ $setGanadosEquipoDos -gt $setGanadosEquipoUno ] ; then
       echo "El equipo ganador es el equipo dos"
    else
       echo "Los equipos empataron"
    fi
fi
```

Forma de llamar al script: bash nombre_ejercicio13.sh nombre_archivo.extension_archivo

<u>Ejercicio 14:</u> Se tiene un directorio con miles de archivos de log que corresponden a resultados de diferentes experimentos. Para facilitar identificar su contenido, se los nombra con el siguiente formato: "ALGORITMO_VARIABLEX_VARIABLEY_VARIABLEZ.log" donde cada variable se indica separada por el carácter "_", y aporta información sobre la clase de experimento a la que se sometió un algoritmo. Codifique un script que dado como parámetro el directorio que contiene los archivos de log, genere un árbol de directorios teniendo en cuenta cada valor de cada variable estudiada y ubique cada archivo de log en el directorio hoja correspondiente.

Ejemplo sobre el nombre de los archivos y una posible ubicación en la estructura del árbol generada:

FloydAlg_400vertices_800edges_minDegree7.log
FloydAlg_400vertices_800edges_minDegree15.log
DijkstraAlg_400vertices_800edges_minDegree7.log
DijkstraAlg_400vertices_800edges_minDegree15.log

400vertices
800edges
minDegree7
FloydAlg_400vertices_800edges_minDegree7.log
DijkstraAlg_400vertices_800edges_minDegree7.log
minDegree15
FloydAlg_400vertices_800edges_minDegree15.log
DijkstraAlg_400vertices_800edges_minDegree15.log

Nota: El criterio para generar los distintos niveles del árbol puede variar dependiendo de los parámetros del script. Asumiendo que el nombre del script es groupExperiments, entonces en: groupExperiments dir 2 1 3, "dir" indica el directorio donde se alojan los archivos de log, "2" indica que los directorios de 1er nivel se forman con los valores de la variable 2, "1" que los directorios de 2do nivel se arman con los valores de la variable 1 y así sucesivamente. (Ejercicio del Parcial 2019).

```
IFS='
'
for archivo in $(find $1 -type f)
do
    direccion=$(dirname $archivo)
    ALGORITMO=$(basename $archivo  | cut -d"_" -f1  | cut -d"." -f1)
    VARIABLEX=$(basename $archivo  | cut -d"_" -f$2 | cut -d"." -f1)
    VARIABLEY=$(basename $archivo  | cut -d"_" -f$3 | cut -d"." -f1)
    VARIABLEZ=$(basename $archivo  | cut -d"_" -f$4 | cut -d"." -f1)
    direccionNueva=$(echo "$direccion/$VARIABLEX/$VARIABLEY/$VARIABLEZ")
    mkdir -p $direccionNueva
    mv $archivo $direccionNueva
done
```

Forma de llamar al script: bash nombre_ejercicio14.sh ruta_directorio número número número

<u>Ejercicio 15:</u> Se tiene un directorio único con archivos de texto, donde cada archivo posee los cómputos finales de votos de los votantes de una ciudad dada. Cada archivo tiene una tabla de tres columnas, donde la primera es el nombre del candidato, la segunda el cargo al cual se postula (presidente, gobernador de pcia. X, etc.), y la tercera la cantidad de votos obtenidos. Los archivos no tienen un orden dado. Escriba un script que dada la ruta a dicho directorio, informe al usuario si hay balotaje o no a nivel presidencial y entre qué candidatos. Habrá balotaje si la cuenta total de votos del candidato presidencial más votado globalmente es mayor al 45%, o bien esta cuenta supera el 40% y además aventaja por 10% o más al segundo candidato presidencial más votado. (Ejercicio del Recuperatorio 2015).

Para la resolución del ejercicio se creó un directorio y en él se crean los archivos de las ciudades con el siguiente formato:

Arturo Rojas|Intendente|24500
Mauricio Macri|Presidente|24500
Maria Eugenia Vidal|Gobernador|24500
Facundo Lopez|Intendente|14300
Alberto Fernandez|Presidente|14300
Axel Kicillof|Gobernador|14300
Roberto Lavagna|Presidente|7500

```
pertenece(){
    votos=0
    for linea in $(cat $1); do
        nombreCandidato=$(echo $linea | cut -d"|" -f1)
        if [ $2 = $nombreCandidato ] ; then
            votos=$(echo $linea | cut -d"|" -f2)
            break;
        fi
    done
    echo $votos
}
cantVotosPresidente=0
> candidatos.txt
IFS='
'
for archivo in $(find $1 -type f); do
    for linea in $(cat $archivo); do
        presidente=$(echo $linea | cut -d"|" -f2)
        if [ $presidente = "Presidente" ] ; then
            nombreCandidato=$(echo $linea | cut -d"|" -f1)
            votos=$(echo $linea | cut -d"|" -f3)
            pertenece=$(pertenece candidatos.txt $nombreCandidato)
            if [ $pertenece -eq "0" ] ; then
                echo "$nombreCandidato|$votos" >> candidatos.txt
            else
                votosCandidato=$(expr $votos + $pertenece)
                reemplazar=$(echo "$nombreCandidato|$pertenece")
                por=$(echo "$nombreCandidato|$votosCandidato")
                sed -i "s/$reemplazar/$por/g" candidatos.txt
            fi
            cantVotosPresidente=$(expr $cantVotosPresidente + $votos)
        fi
    done
done
primero=0
segundo=0
for linea in $(cat candidatos.txt); do
    nombreCandidato=$(echo $linea | cut -d"|" -f1)
    votos=$(echo $linea | cut -d"|" -f2)
    if (( $votos > $primero )) ; then
        primero=$votos
        candidatoPrimero=$(echo "$nombreCandidato")
    else
        if (( $votos > $segundo )) ; then
            segundo=$votos
            candidatoSegundo=$(echo "$nombreCandidato")
        fi
    fi
done
porcCuarentaYCinco=$(expr $cantVotosPresidente*0.45 | bc)
porcCuarenta=$(expr $cantVotosPresidente*0.4 | bc)
porcDiez=$(expr $segundo*1.10 | bc)
```

```
primero=$(expr $primero*1.00 | bc)
if (( $(echo "$primero > $porcCuarentaYCinco" | bc) )) ; then
   echo "gano las elecciones $candidatoPrimero"
else
   if (( $(echo "$primero > $porcCuarenta" | bc)  )) && (( $(echo "$primero > $porcDiez" | bc)  )) ; then
          echo "gano las elecciones $candidatoPrimero"
   else
      echo "Hay balotaje entre los candidatos $candidatoPrimero y $candidatoSegundo"
   fi
fi
```

Forma de llamar al script: bash nombre_ejercicio15.sh ruta_directorio

<u>Ejercicio 16:</u> Codifique un script que reciba un archivo de texto como parámetro y un valor numérico N, e implemente un algoritmo simple para resumir el texto. Básicamente, el script debe generar un nuevo archivo como salida que elimine todas las frases del archivo que tengan más de N palabras.

El archivo de entrada consta de líneas no vacías, donde cada línea representa una frase del texto completo. Cada línea (esto es, frase) contiene más de una palabra, separadas por un carácter especial (a elección). El script también debe informar por la pantalla la cantidad de líneas del archivo de entrada y del producido como salida. (Ejercicio del Recuperatorio 2018).

```
> resumen.txt
CantidadLineasEntrada=$(cat $1 | wc -l)
IFS='
'
for linea in $(cat $1); do
   CantPalabras=$(echo $linea | grep -o " " | wc -l) #grep -o te muestra la palabra donde hizo match, sin el -0 te muestra la linea donde hizo match
   CantPalabras=$(expr $CantPalabras + 1)
   if (( $CantPalabras <= $2 )); then
      echo $linea >> resumen.txt
   fi
done
CantidadLineasSalida=$(cat resumen.txt | wc -l)
echo "La cantidad de lineas del archivo de entrada es $CantidadLineasEntrada"
echo "La cantidad de lineas del archivo de salida es $CantidadLineasSalida"
```

Forma de llamar al script: bash nombre_ejercicio16.sh nombre_archivo.extension_archivo número