1. Calculadora simple: Crea un script que solicite al usuario dos números y luego le permita elegir entre diferentes operaciones matemáticas (suma, resta, multiplicación, división). El script debería mostrar el resultado de la operación elegida.

```
#!/bin/bash

# Solicitar al usuario dos números
read -p "Introduce el primer número: " num1
read -p "Introduce el segundo número: " num2

# Mostrar menú de opciones
echo "Selecciona una operación:"
echo "1. Suma"
echo "2. Resta"
echo "3. Multiplicación"
echo "4. División"
read -p "Opción: " opcion

# Realizar la operación seleccionada
case $opcion in
  1) resultado=$(echo "$num1 + $num2" | bc);;
  2) resultado=$(echo "$num1 - $num2" | bc);;
  3) resultado=$(echo "$num1 * $num2" | bc);;
```

```
    4) resultado=$(echo "scale=2; $num1 / $num2" | bc);;

    *) echo "Opción no válida"; exit 1;;

esac



# Mostrar el resultado

echo "El resultado es: $resultado"
```

2.Conversor de unidades: Desarrolla un script que convierta una cantidad dada en una unidad de medida (por ejemplo, metros) a otra unidad de medida (por ejemplo, pies). El usuario debe poder elegir las unidades de entrada y salida, y el script debe realizar la conversión.

```
#!/bin/bash

echo "Conversor de unidades"

echo "Introduce la cantidad a convertir:"
read cantidad

# Mostrar las opciones de unidades de entrada
echo "Selecciona la unidad de entrada:"
echo "1. Metros"
echo "2. Pies"
echo "3. Kilómetros"
echo "4. Millas"
read -p "Opción: " unidad_entrada

# Mostrar las opciones de unidades de salida
echo "Selecciona la unidad de salida:"
echo "1. Metros"
echo "2. Pies"
echo "3. Kilómetros"
echo "4. Millas"
read -p "Opción: " unidad_salida
```

```
# Realizar la conversión
case $unidad_entrada in
    1) # Metros
        case $unidad_salida in
            1) resultado=$cantidad;;
            2) resultado=$(echo "scale=2; $cantidad * 3.281" | bc);;
            3) resultado=$(echo "scale=2; $cantidad / 1000" | bc);;
            4) resultado=$(echo "scale=2; $cantidad / 1609" | bc);;
            *) echo "Opción no válida"; exit 1;;
        esac;;
    2) # Pies
        case $unidad_salida in
            1) resultado=$(echo "scale=2; $cantidad / 3.281" | bc);;
            2) resultado=$cantidad;;
            3) resultado=$(echo "scale=2; $cantidad / 3281" | bc);;
            4) resultado=$(echo "scale=2; $cantidad / 5280" | bc);;
            *) echo "Opción no válida"; exit 1;;
        esac;;
    3) # Kilómetros
        case $unidad_salida in
            1) resultado=$(echo "scale=2; $cantidad * 1000" | bc);;
            2) resultado=$(echo "scale=2; $cantidad * 3281" | bc);;
            3) resultado=$cantidad;;
            4) resultado=$(echo "scale=2; $cantidad / 1.609" | bc);;
            *) echo "Opción no válida"; exit 1;;
        esac;;
    4) # Millas
        case $unidad_salida in
            1) resultado=$(echo "scale=2; $cantidad * 1609" | bc);;
            2) resultado=$(echo "scale=2; $cantidad * 5280" | bc);;
            3) resultado=$(echo "scale=2; $cantidad * 1.609" | bc);;
            4) resultado=$cantidad;;
            *) echo "Opción no válida"; exit 1;;
        esac;;
    *) echo "Opción no válida"; exit 1;;
esac

# Mostrar resultado
echo "El resultado de la conversión es: $resultado"
```

3. Juego de adivinanza: Crea un juego de adivinanzas en el que el programa selecciona un número aleatorio y le pide al usuario que lo adivine. El programa debe proporcionar pistas como "Demasiado alto" o "Demasiado bajo" según corresponda, y seguirá solicitando al usuario que intente adivinar hasta que lo haga correctamente.

```
#!/bin/bash

# Generar un número aleatorio entre 1 y 100

numero_aleatorio=$((RANDOM % 100 + 1))



echo "Bienvenido al juego de adivinanza!"

echo "Intenta adivinar el número secreto (entre 1 y 100)"



intentos=0

adivinado=false



while [ "$adivinado" = false ]; do

    read -p "Introduce tu suposición: " suposicion

    ((intentos++))



    if [ "$suposicion" -lt "$numero_aleatorio" ]; then

        echo "Demasiado bajo. Intenta de nuevo."

    elif [ "$suposicion" -gt "$numero_aleatorio" ]; then
```

```
        echo "Demasiado alto. Intenta de nuevo."
    else
        echo "¡Felicidades! ¡Has adivinado el número en $intentos intentos!"
        adivinado=true
    fi
done
```

4. Generador de contraseñas: Escribe un script que genere una contraseña aleatoria con cierta longitud especificada por el usuario. El script debe incluir una combinación de letras mayúsculas, minúsculas, números y caracteres especiales.

```
#!/bin/bash

# Solicitar la longitud de la contraseña al usuario
read -p "Ingrese la longitud de la contraseña: " longitud

# Inicializar la variable de la contraseña
password=""

# Caracteres para construir la contraseña
caracteres="ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123
456789@#$%&"
```

```
# Longitud total de los caracteres disponibles

longitud_caracteres=${#caracteres}



# Generar la contraseña aleatoria

for (( i=0; i<longitud; i++ )); do

  # Generar un número aleatorio entre 0 y longitud_caracteres-1

  indice=$((RANDOM % longitud_caracteres))

  # Concatenar el carácter correspondiente al índice aleatorio a la contraseña

  password+="${caracteres:$indice:1}"

done



# Mostrar la contraseña generada

echo "La contraseña generada es: $password"
```

5. Copia de seguridad de archivos: Desarrolla un script que realice copias de seguridad de ciertos archivos o directorios especificados por el usuario. El script debería comprimir los archivos en un archivo de respaldo y almacenarlo en un directorio de destino especificado.

```
#!/bin/bash

# Solicitar al usuario la ruta del directorio o archivo a respaldar
read -p "Ingrese la ruta del directorio o archivo a respaldar: " ruta_origen

# Verificar si la ruta de origen es válida
if [ ! -e "$ruta_origen" ]; then
    echo "La ruta especificada no es válida."
```

```
    exit 1
fi

# Solicitar al usuario la ruta del directorio de destino
read -p "Ingrese la ruta del directorio de destino para el respaldo: " directorio_destino

# Verificar si la ruta de destino es válida
if [ ! -d "$directorio_destino" ]; then
    echo "La ruta de destino especificada no es válida."
    exit 1
fi

# Obtener el nombre base del archivo o directorio a respaldar
nombre_base=$(basename "$ruta_origen")

# Crear el nombre del archivo de respaldo con la fecha y hora actual
fecha=$(date +"%Y-%m-%d_%H-%M-%S")
archivo_respaldo="${nombre_base}_${fecha}.tar.gz"

# Comprimir el archivo o directorio de origen en un archivo de respaldo
tar -czf "${directorio_destino}/${archivo_respaldo}" "$ruta_origen"

# Verificar si la operación de respaldo fue exitosa
if [ $? -eq 0 ]; then
    echo "Respaldo completado correctamente."
else
    echo "Ocurrió un error durante el respaldo."
fi
```