Task 1: File Management Script

Write a Bash script that

- Creates a directory named "backup" in the user's home directory^
- Copies all .txt files from the current directory into the "backup" directory^
- Appends the current date and time to the filenames of the copied files.

```
#!/bin/bash

# Define the backup directory path
BACKUP_DIR="$HOME/backup"

# Create the backup directory if it doesn't exist
mkdir -p "$BACKUP_DIR"

# Get current date and time in format YYYY-MM-DD_HH-MM-SS
CURRENT_TIME=$(date +"%Y-%m-%d_%H-%M-%S")

# Loop through all .txt files in the current directory
for file in *.txt; do
# Check if any .txt files exist to avoid errors
if [[ -f "$file" ]]; then
   # Get the base filename without extension
   BASENAME=$(basename "$file" .txt)
   # Define new filename with date and time appended
   NEW_FILENAME="${BASENAME}_$CURRENT_TIME.txt"
   # Copy the file to the backup directory with the new name
   cp "$file" "$BACKUP_DIR/$NEW_FILENAME"
fi
```

echo "✅ Backup completed. All .txt files copied to $BACKUP_DIR with timestamp."

Task 2: System Health Check

Create a script thatg

- Checks the system's CPU and memory usage^
- Reports if the CPU usage is above 80% or if the available memory is below 20%^
- Logs the results to a file named system_health.log.

```
#!/bin/bash

LOG_FILE="system_health.log"
TIMESTAMP=$(date +"%Y-%m-%d %H:%M:%S")

CPU_IDLE=$(top -bn1 | grep "Cpu(s)" | awk '{print $8}' | cut -d "." -f1)
CPU_USAGE=$((100 - CPU_IDLE))

MEM_TOTAL=$(free -m | awk '/Mem:/ {print $2}')
MEM_AVAILABLE=$(free -m | awk '/Mem:/ {print $7}')
MEM_AVAILABLE_PERCENT=$((100 * MEM_AVAILABLE / MEM_TOTAL))

echo "[$TIMESTAMP] CPU Usage: ${CPU_USAGE}% | Available Memory: ${MEM_AVAILABLE_PERCENT}%" >> $LOG_FILE

if [ "$CPU_USAGE" -gt 80 ]; then
echo "[$TIMESTAMP] WARNING: High CPU usage - ${CPU_USAGE}%" >> $LOG_FILE
fi

if [ "$MEM_AVAILABLE_PERCENT" -lt 20 ]; then
echo "[$TIMESTAMP] WARNING: Low available memory - ${MEM_AVAILABLE_PERCENT}%" >> $LOG_FILE
fi
```

Task 3: User Account Mar

- Reads a list of usernames from a file (e.g., user_list.txt)^

- Creates a new user for each username^
- Generates a random password for each user and saves the username and password to a file named credentials.txt.

```
#!/bin/bash

USER_FILE="user_list.txt"
CRED_FILE="credentials.txt"
> "$CRED_FILE"

while IFS= read -r username || [ -n "$username" ]; do
if id "$username" &>/dev/null; then
   echo "User $username already exists. Skipping..."
 else
   PASSWORD=$(openssl rand -base64 12)
   useradd -m "$username"
   echo "$username:$PASSWORD" | chpasswd
   echo "$username:$PASSWORD" >> "$CRED_FILE"
   echo "User $username created."
fi
done < "$USER_FILE"

echo "✅ All users processed. Credentials saved to $CRED_FILE."
```

Task 4: Automated Backup

Create a script thatg

- Takes a directory path as input from the user^
- Compresses the directory into a .tar.gz file^
- Saves the compressed file with a name that includes the current date (e.g., backup_2023-08-20.tar.gz).

```
#!/bin/bash

read -p "Enter the directory path to back up: " dir_path

if [ ! -d "$dir_path" ]; then
  echo "❌ Directory does not exist: $dir_path"
  exit 1
fi

dir_name=$(basename "$dir_path")
current_date=$(date +%F)
backup_file="backup_${dir_name}_${current_date}.tar.gz"

 tar -czf "$backup_file" -C "$(dirname "$dir_path")" "$dir_name"

if [ $? -eq 0 ]; then
  echo "✅ Backup successful! File created: $backup_file"
else
  echo "❌ Backup failed."
fi
```

Task 5: Simple To-Do List

Create a Bash script thatg

- Implements a simple command-line to-do list^
- Allows the user to add tasks, view tasks, and remove tasks^
- Saves the tasks to a file (e.g., todo.txt).

```
#!/bin/bash

TODO_FILE="todo.txt"
touch "$TODO_FILE"

while true; do
 echo ""
 echo "=== Simple To-Do List ==="
```

```
 echo "1. View tasks"
 echo "2. Add a task"
 echo "3. Remove a task"
 echo "4. Exit"
 read -p "Choose an option [1-4]: " choice

case $choice in
    1)
        echo ""
        echo "--- Your Tasks ---"
        if [[ ! -s "$TODO_FILE" ]]; then
            echo "No tasks found."
        else
            nl -w2 -s'. ' "$TODO_FILE"
        fi
        ;;
    2)
        read -p "Enter the task: " task
        echo "$task" >> "$TODO_FILE"
        echo "✅ Task added!"
        ;;
    3)
        echo ""
        nl -w2 -s'. ' "$TODO_FILE"
        read -p "Enter the task number to remove: " num
        if [[ "$num" =~ ^[0-9]+$ ]]; then
            sed -i "${num}d" "$TODO_FILE"
            echo "🗑️ Task removed."
        else
            echo "❌ Invalid number."
        fi
        ;;
    4)
        echo "👋 Goodbye!"
        exit 0
        ;;
    *)
        echo "❌ Invalid option. Please enter 1-4."
        ;;
esac
done
```

```
# Task 7:  Text File Processing
Create a script thatg
- Takes a text file as input^
- Counts and displays the number of lines, words, and characters in the file^
- Finds and displays the longest word in the file.


   #!/bin/bash

   read -p "Enter the path to the text file: " file

   if [ ! -f "$file" ]; then
     echo "❌ File not found: $file"
     exit 1
   fi

   lines=$(wc -l < "$file")
   words=$(wc -w < "$file")
   chars=$(wc -m < "$file")

   longest=$(tr -cs '[:alnum:]' '[\n*]' < "$file" | awk '{ if (length > max) {
   max = length; word = $0 } } END { print word }')

   echo ""
   echo "📄 File: $file"
   echo "-------------------------------------"
   echo "📏 Lines       : $lines"
   echo "📝 Words       : $words"
   echo "🔠 Characters  : $chars"
   echo "🏆 Longest word: $longest"
```

Task 7: Text File Processing

Create a script thatg

- Takes a text file as input^
- Counts and displays the number of lines, words, and characters in the file^
- Finds and displays the longest word in the file.

```
#!/bin/bash

read -p "Enter the path to the text file: " file

if [ ! -f "$file" ]; then
  echo "❌ File not found: $file"
  exit 1
fi

lines=$(wc -l < "$file")
words=$(wc -w < "$file")
chars=$(wc -m < "$file")

longest=$(tr -cs '[:alnum:]' '[\n*]' < "$file" | awk '{ if (length > max) { max = length; word = $0 } } END { print word }')

echo ""
echo "📄 File: $file"
echo "-----------------------------------"
echo "📏 Lines       : $lines"
echo "📝 Words       : $words"
echo "🔠 Characters  : $chars"
echo "🏆 Longest word: $longest"
```