Q1.

```
#!/bin/bash
BACKUP_DIR="$HOME/backup"
mkdir -p "$BACKUP_DIR"
TIMESTAMP=$(date +"%Y-%m-%d_%H-%M-%S")
for FILE in *.txt; do
  if [[ -e "$FILE" ]]; then
      BASENAME=$(basename "$FILE" .txt)
          cp "$FILE" "$BACKUP_DIR/${BASENAME}_$TIMESTAMP.txt"
  fi
done

echo "Backup complete. Files copied to $BACKUP_DIR with timestamp."
```

Q2

```
#!/bin/bash
LOG_FILE="system_health.log"

# Get current date and time
TIMESTAMP=$(date +"%Y-%m-%d %H:%M:%S")

# Get CPU usage (user + system) as a percentage
CPU_USAGE=$(top -bn1 | grep "Cpu(s)" | awk '{print 100 - $8}')
CPU_USAGE_INT=${CPU_USAGE%.*}

# Get total and available memory in MB
MEM_TOTAL=$(free -m | awk '/Mem:/ {print $2}')
MEM_AVAILABLE=$(free -m | awk '/Mem:/ {print $7}')

# Calculate memory usage percentage
MEM_USAGE_PERCENT=$((100 - (MEM_AVAILABLE * 100 / MEM_TOTAL)))
```

```
# Initialize status messages
CPU_STATUS="OK"
MEM_STATUS="OK"

# Check CPU threshold
if [ "$CPU_USAGE_INT" -gt 80 ]; then
   CPU_STATUS="HIGH"
fi

# Check Memory threshold
if [ "$MEM_USAGE_PERCENT" -gt 80 ]; then
   MEM_STATUS="LOW"
fi

# Create log entry
LOG_ENTRY="$TIMESTAMP | CPU Usage: ${CPU_USAGE_INT}% [$CPU_STATUS] | Memory Usage: ${MEM_USAGE_PERCENT}% [$MEM_STATUS]"

# Write to log file
echo "$LOG_ENTRY" >> "$LOG_FILE"

# Optional: Print result to terminal
echo "$LOG_ENTRY"
```

Q3

```
#!/bin/bash

# Input and output files
USER_FILE="user_list.txt"
CRED_FILE="credentials.txt"
```

```
# Clear or create the credentials file
> "$CRED_FILE"

# Check if user list file exists
if [[ ! -f "$USER_FILE" ]]; then
  echo "User list file '$USER_FILE' not found."
  exit 1
fi

# Loop through each line (username)
while IFS= read -r USERNAME || [[ -n "$USERNAME" ]]; do
  # Skip empty lines
  [[ -z "$USERNAME" ]] && continue

  # Check if user already exists
  if id "$USERNAME" &>/dev/null; then
    echo "User $USERNAME already exists. Skipping."
    continue
  fi

  # Generate a random password (12 characters)
  PASSWORD=$(openssl rand -base64 12)

  # Create the user without home directory (-M) or with home (-m)
  useradd -m "$USERNAME"

  # Set the user's password
  echo "${USERNAME}:${PASSWORD}" | chpasswd

  # Log credentials
```

```
    echo "${USERNAME}:${PASSWORD}" >> "$CRED_FILE"

    echo "Created user: $USERNAME"
done < "$USER_FILE"

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

Q4

```
#!/bin/bash

# Prompt user for the directory path
read -rp "Enter the full path of the directory to back up: " DIR_PATH

# Check if the directory exists
if [[ ! -d "$DIR_PATH" ]]; then
    echo "Error: Directory '$DIR_PATH' does not exist."
    exit 1
fi

# Get base name of directory (e.g., /home/user/docs → docs)
DIR_NAME=$(basename "$DIR_PATH")

# Get current date
DATE=$(date +%F)  # Format: YYYY-MM-DD

# Define archive name
BACKUP_FILE="backup_${DIR_NAME}_${DATE}.tar.gz"

# Create the compressed archive
tar -czf "$BACKUP_FILE" -C "$(dirname "$DIR_PATH")" "$DIR_NAME"
```

```
# Notify the user
echo "Backup completed: $BACKUP_FILE"
```

Q5.

```
#!/bin/bash

# File to store tasks
TODO_FILE="todo.txt"

# Ensure the file exists
touch "$TODO_FILE"

# Display menu
show_menu() {
  echo "=== Simple To-Do List ==="
  echo "1) View tasks"
  echo "2) Add a task"
  echo "3) Remove a task"
  echo "4) Exit"
}

# View tasks
view_tasks() {
  echo "---- Your Tasks ----"
  if [[ ! -s "$TODO_FILE" ]]; then
    echo "No tasks yet!"
  else
    nl -w2 -s'. ' "$TODO_FILE"
  fi
  echo "--------------------"
}
```

```
# Add a task
add_task() {
  read -rp "Enter the task: " TASK
  echo "$TASK" >> "$TODO_FILE"
  echo "Task added."
}

# Remove a task
remove_task() {
  view_tasks
  read -rp "Enter the task number to remove: " TASK_NUM
  if [[ "$TASK_NUM" =~ ^[0-9]+$ ]]; then
    sed -i "${TASK_NUM}d" "$TODO_FILE" && echo "Task removed." || echo "Invalid task number."
  else
    echo "Please enter a valid number."
  fi
}

# Main loop
while true; do
  show_menu
  read -rp "Choose an option [1-4]: " CHOICE
  case $CHOICE in
    1) view_tasks ;;
    2) add_task ;;
    3) remove_task ;;
    4) echo "Goodbye!"; exit 0 ;;
    *) echo "Invalid option. Try again." ;;
  esac
  echo
```

```
done
```

Q6.

```
#!/bin/bash

# Input and log files
PACKAGE_FILE="packages.txt"
LOG_FILE="install_log.txt"

# Clear previous log
> "$LOG_FILE"

# Check if package file exists
if [[ ! -f "$PACKAGE_FILE" ]]; then
   echo "Error: '$PACKAGE_FILE' not found."
   exit 1
fi

# Detect available package manager
if command -v apt &> /dev/null; then
   PKG_MGR="apt"
   UPDATE_CMD="apt update -y"
   INSTALL_CMD="apt install -y"
elif command -v dnf &> /dev/null; then
   PKG_MGR="dnf"
   UPDATE_CMD="dnf check-update -y"
   INSTALL_CMD="dnf install -y"
elif command -v yum &> /dev/null; then
   PKG_MGR="yum"
   UPDATE_CMD="yum check-update -y"
   INSTALL_CMD="yum install -y"
```

```
else
  echo "No supported package manager found (apt, dnf, yum)."
  exit 1
fi

echo "Using package manager: $PKG_MGR"
echo "Logging installation status to: $LOG_FILE"
echo

# Update package list
echo "Updating package list..."
eval "$UPDATE_CMD" &>> "$LOG_FILE"

# Read package list and install
while IFS= read -r PACKAGE || [[ -n "$PACKAGE" ]]; do
  # Skip empty lines or comments
  [[ -z "$PACKAGE" || "$PACKAGE" =~ ^# ]] && continue

  echo "Installing: $PACKAGE"

  if sudo $INSTALL_CMD "$PACKAGE" &>> "$LOG_FILE"; then
    echo "[OK] Installed $PACKAGE" | tee -a "$LOG_FILE"
  else
    echo "[FAIL] Failed to install $PACKAGE" | tee -a "$LOG_FILE"
  fi
done < "$PACKAGE_FILE"

echo
echo "Installation complete. Check '$LOG_FILE' for details."
```

Q7.

```
#!/bin/bash

# Check if a file path was provided
if [[ -z "$1" ]]; then
   echo "Usage: $0 <text_file>"
   exit 1
fi

FILE="$1"

# Check if the file exists
if [[ ! -f "$FILE" ]]; then
   echo "Error: File '$FILE' not found."
   exit 1
fi

# Count lines, words, and characters
LINE_COUNT=$(wc -l < "$FILE")
WORD_COUNT=$(wc -w < "$FILE")
CHAR_COUNT=$(wc -m < "$FILE")

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

# Display the results
echo "File: $FILE"
echo "------------------------"
echo "Lines     : $LINE_COUNT"
echo "Words     : $WORD_COUNT"
echo "Characters: $CHAR_COUNT"
echo "Longest word: $LONGEST_WORD"
```