**1. Write a shell script for implementing Voting System:**

echo "Enter your Name"

read name

echo "Enter your Age"

read age

if [ $age -gt 17 ]; then

echo "Whom you want to vote?"

read vote

echo "Mr. $name, you have voted $vote "

else

echo "Sorry Mr. $name, you are not eligible to vote"

fi

**Output:**

**2. Write a shell script to find sum and average of 3 No’s:**

echo "Enter 1st No."

read a

echo "Enter 2nd No."

read b

echo "Enter 3nd No."

read c

sum=$(($a + $b + $c))

avg=`expr $sum / 3`

echo "The sum is $sum"

echo "The average is $avg"

**Output:**

**3. Write a shell script to find a no. is even or odd:**

echo "Enter the no.:"

read a

if [ $((a%2)) -eq 0 ]; then

echo "The number $a is Even"

else

echo "The number $a is Odd"

fi

**Output:**

**4. Write a shell script to find a no. is prime or not:**

echo "Enter the number:"

read n

x=true

if [ "$n" -lt 2 ]; then

x=false

else

for ((i=2; i\*i<=n; i++)); do

if (( n % i == 0 )); then

x=false

break

fi

done

fi

if [ "$x" = true ]; then

echo "$n is a prime number"

else

echo "$n is not a prime number"

fi

**5. Write a shell script to find factorial of a number:**

echo "Enter the number:"

read num

fact=1;

for((i=1;i<=num;i++));do

fact=$((fact\*i))

done

echo "The factorial is $fact"

**Output:**

**6. Write a shell script to print multiplication table of a no. :**

echo "Which number table you want to print?"

read num

for i in 1 2 3 4 5 6 7 8 9 10

do

echo "$num x $i = $((i\*num))"

done

**Output:**

**7. Write a shell script to find out the addition of given number’s digits:**

echo "Enter a number: "

read num

sum=0

while [ $num -gt 0 ]; do

digit=$(( num % 10 ))

sum=$(( sum + digit ))

num=$(( num / 10 ))

done

echo "Sum of digits: $sum"

**Output:**

**8. Write a shell program to print number pattern upto 9:**

for (( i=1; i<=9; i++ ))

do

for (( j=1; j<=i; j++ ))

do

echo -n "$j"

done

echo

done

**Output:**

**9. Write a shell program to print reverse number pattern:**

for (( i=9; i>=1; i-- ))

do

for (( j=1; j<=i; j++ ))

do

echo -n "$j"

done

echo

done

**Output:**

**10. Write a shell program to print number pyramid pattern:**

for (( i=1; i<=9; i++ ))

do

for (( j=i; j<9; j++ ))

do

echo -n " "

done

for (( j=1; j<=i; j++ ))

do

echo -n "$j "

done

echo

done

**11. Write a shell program for Implementation of Array:**

declare -a arr

echo "Enter the number of elements:"

read n

echo "Enter $n elements:"

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

read element

arr[i]=$element

done

echo "Elements of the array:"

for element in "${arr[@]}"; do

echo "$element"

done

**12. Write a shell script to implement string:**

declare -a str

echo "Enter the string::"

read n

echo "$n"

**Output:**

**13. Write a shell script to check String is Palindrome or not:**

declare -a str

echo "Enter the string"

read s

len=${#s}

rev=""

for (( i=$len-1;i>=0;i--))do

rev="$rev${s:i:1}"

done

if [ "$s" == "$rev" ];then

echo "String $s is Palindrome"

else

echo "String $s is not Palindrome"

fi

**Output:**

**14. Write a shell script to reverse a string:**

declare -a str

echo "Enter the string"

read s

len=${#s}

rev\_st=""

for (( i=$len-1;i>=0;i--))do

rev="$rev${s:i:1}"

done

echo "The reversed string is $rev"

**Output:**

**15. Write a shell script to concatinate two strings:**

declare -a str

echo "Enter the first string"

read s1

echo "Enter the second string"

read s2

st="$s1$s2"

echo "The concatinated string is $st"

**Output:**

**16. Write a shell script to find the length of a string:**

declare -a str

echo "Enter the string:"

read s

echo "The length of string is: ${#s}"

**Output:**

**17. Write a shell script to remove the first occurrence of a substring from a string:**

echo "Enter the string:"

read s1

awk '{ sub(/.!!/, ""); sub(/kumari./, ""); print }' <<< "$s1"

**Output:**

**18. Write a shell script to find the no. of lines in all .txt files in a directory:**

DIR="/mnt/c/Users/basant/os\_lab"

if [ ! -d "$DIR" ]; then

echo "Directory $DIR does not exist."

exit 1

fi

for FILE in "$DIR"/\*.txt; do

if [ -f "$FILE" ]; then

echo "Processing $FILE"

LINE\_COUNT=$(wc -l < "$FILE")

echo "$FILE has $LINE\_COUNT lines."

else

echo "No .txt files found in $DIR."

fi

done

**19. Write a shell script to implement FCFS Algorithm:**

read -p "Enter number of processes: " n

declare -a pid at bt wt tat ct

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

pid[i]=$i

read -p "Enter Arrival Time for Process $((i+1)): " at[i]

read -p "Enter Burst Time for Process $((i+1)): " bt[i]

done

for ((i=0; i<n-1; i++)); do

for ((j=0; j<n-i-1; j++)); do

if (( at[j] > at[j+1] )); then

tmp=${at[j]}; at[j]=${at[j+1]}; at[j+1]=$tmp

tmp=${bt[j]}; bt[j]=${bt[j+1]}; bt[j+1]=$tmp

tmp=${pid[j]}; pid[j]=${pid[j+1]}; pid[j+1]=$tmp

fi

done

done

wt[0]=0

ct[0]=$((at[0] + bt[0]))

tat[0]=$((ct[0] - at[0]))

for ((i=1; i<n; i++)); do

if (( ct[i-1] < at[i] )); then

ct[i]=$((at[i] + bt[i]))

else

ct[i]=$((ct[i-1] + bt[i]))

fi

tat[i]=$((ct[i] - at[i]))

wt[i]=$((tat[i] - bt[i]))

done

echo -e "\nProcess\tAT\tBT\tWT\tTAT"

total\_wt=0

total\_tat=0

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

echo -e "$((pid[i]+1))\t${at[i]}\t${bt[i]}\t${wt[i]}\t${tat[i]}"

total\_wt=$((total\_wt + wt[i]))

total\_tat=$((total\_tat + tat[i]))

**Output:**

**20. Case Study 1: A company wants to enforce strong passwords for its employees. You need to write a shell script that checks if a given password meets the following criteria: At least 8 characters long, Contains at least one uppercase letter, one lowercase letter, one digit, and one special character. Input: A password string Output: "Strong password" or "Weak password"**

echo "Enter your password:"

read password

if [[ ${#password} -lt 8 ]]; then

echo "Weak password"

elif ! echo "$password" | grep -q '[A-Z]'; then

echo "Weak password"

elif ! echo "$password" | grep -q '[a-z]'; then

echo "Weak password"

elif ! echo "$password" | grep -q '[0-9]'; then

echo "Weak password"

elif ! echo "$password" | grep -q '[^a-zA-Z0-9]'; then

echo "Weak password"

else

echo "Strong password"

fi

**Output:**

**21. Case Study 2: A script that searches for a given word in a file and replaces it with another word. Input: Find: error, Replace with: warning Output: "Replaced all occurrences of 'error' with 'warning' in the file.**

grep "error" ques2.txt | awk '{gsub(/error/, "warning"); print}' > temp.txt && mv temp.txt ques2.txt

**Ques2.txt before running Script:**

**Ques2.txt after running Script:**

**22. Pipeline Case Study 1: You are a system administrator for a large company, and you have been given a task to generate a report from system logs. The system logs are stored in the file /var/log/syslog, and they contain various messages related to system processes, authentication, and other services. Your manager wants a summary report that includes: The total number of log entries in the file. The number of entries related to authentication failures. The top 5 processes that generated the most log entries. A count of unique IP addresses that attempted to log in to the system. A list of failed SSH login attempts sorted by frequency. You must accomplish this task using a single shell command pipeline.**

echo "Total log entries: $(wc -l < /var/log/syslog)" && \

echo "Authentication failures: $(grep -i 'authentication failure' /var/log/syslog | wc -l)" && \

echo "Top 5 processes by log entries:" && \

awk '{for(i=1;i<=NF;i++) if ($i ~ /^\[?[a-zA-Z0-9\_.-]+(\[[0-9]+\])?:$/) {gsub(/\[.\*\]|:/,"",$i); print $i}}' /var/log/syslog | \

sort | uniq -c | sort -nr | head -5 && \

echo "Unique IPs that attempted login: $(grep -Eo '([0-9]{1,3}\.){3}[0-9]{1,3}' /var/log/syslog | sort | uniq | wc -l)" && \

echo "Failed SSH login attempts sorted by frequency:" && \

grep "Failed password" /var/log/syslog | grep -Eo 'from ([0-9]{1,3}\.){3}[0-9]{1,3}' | awk '{print $2}' | \

sort | uniq -c | sort -nr

**23. Pipeline Case Study 2: Write a shell script that extracts all currently logged-in users, removes duplicates, sorts them alphabetically, and displays them.**

who | awk '{print $1}' | sort | uniq

**Output:**

**24. Pipeline Case Study 3: Write a script to display the most frequently used command from the .bash\_history file.**

n=$(awk '{print $1}' ~/.bash\_history | sort | uniq -c | sort -nr | head -1)

echo "Most frequently used command: $n"

**Output:**