


1. Write a Shell program to check the given number is even or odd.

CODE

```
function check_odd_even {  
    if [ $(( $1 % 2 )) -eq 0 ]; then  
        echo "$1 is even"  
    else  
        echo "$1 is odd"  
    fi  
}  
echo "Enter a number:"  
read num  
check_odd_even $num
```



```
bash: q8.sh: no such file or directory  
(base) sjcet@HP-Z238:~/Ajesh/Network/c2$ bash q8.sh  
Enter a number:  
10  
10 is even  
(base) sjcet@HP-Z238:~/Ajesh/Network/c2$ bash q8.sh  
Enter a number:  
5  
5 is odd  
(base) sjcet@HP-Z238:~/Ajesh/Network/c2$ bash q8.sh  
Enter a number:  
41  
41 is odd  
(base) sjcet@HP-Z238:~/Ajesh/Network/c2$ bash q8.sh  
Enter a number:  
26  
26 is even  
(base) sjcet@HP-Z238:~/Ajesh/Network/c2$
```

2. Write a Shell program to check a leap year.

CODE

```
echo -n "Enter year: "  
read year
```

```
if [ $((year % 4)) -eq 0 ] && [ $((year % 100)) -ne 0 ] || [ $((year % 400)) -eq  
0 ]; then  
    echo "$year is a leap year."  
else  
    echo "$year is not a leap year."  
fi
```

```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms$ bash q2.sh  
bash: q2.sh: No such file or directory  
(base) sjcet@HP-Z238:~/Ajesh/Network/programms$ bash q3.sh  
Enter year: 2056  
2056 is a leap year.  
(base) sjcet@HP-Z238:~/Ajesh/Network/programms$ bash q3.sh  
Enter year: 2024  
2024 is a leap year.  
(base) sjcet@HP-Z238:~/Ajesh/Network/programms$ bash q3.sh  
Enter year: 2022  
2022 is not a leap year.  
(base) sjcet@HP-Z238:~/Ajesh/Network/programms$
```

3. Write a Shell program to find the area and circumference of a circle.

CODE

```
echo "Enter the radius of the circle: "  
read radius
```

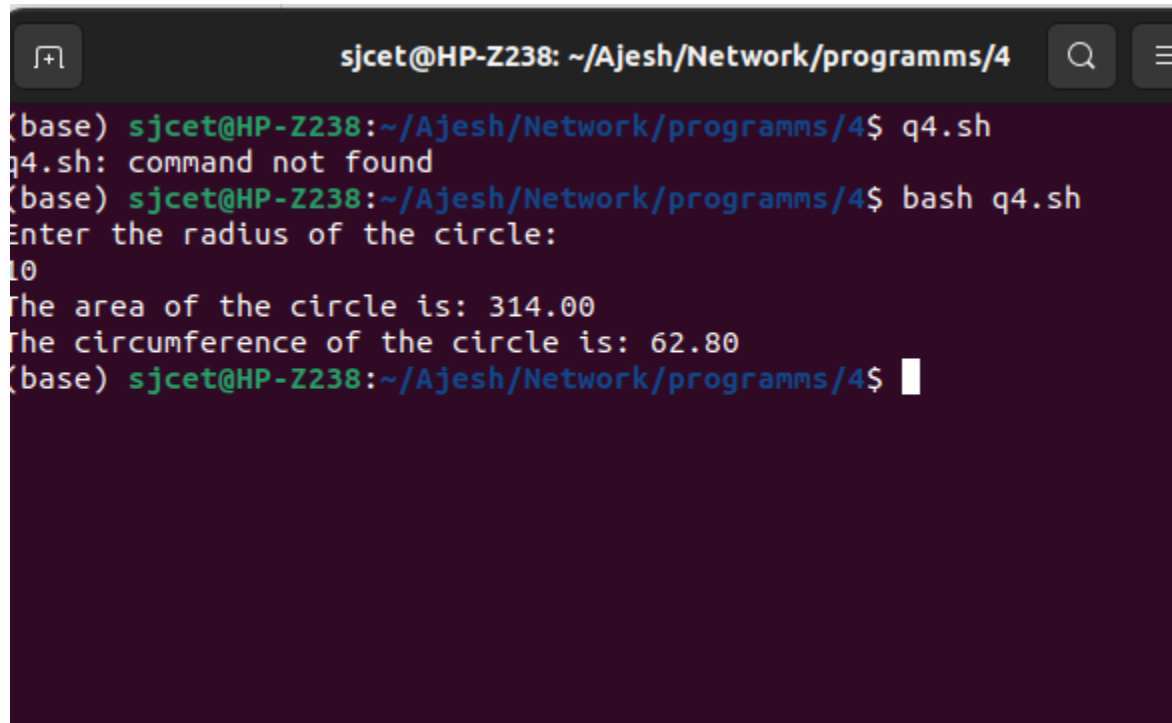
```
# Calculate the area of the circle  
area=$(echo "scale=2; 3.14 * ($radius^2)" | bc)
```

```
# Calculate the circumference of the circle
circumference=$(echo "scale=2; 2 * 3.14 * $radius" | bc)
```

```
# Print the results
```

```
echo "The area of the circle is: $area"
```

```
echo "The circumference of the circle is: $circumference"
```



```
sjcet@HP-Z238: ~/Ajesh/Network/programms/4
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/4$ q4.sh
q4.sh: command not found
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/4$ bash q4.sh
Enter the radius of the circle:
10
The area of the circle is: 314.00
The circumference of the circle is: 62.80
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/4$
```

4. Write a Shell program to check the given number and its reverse are same.

CODE

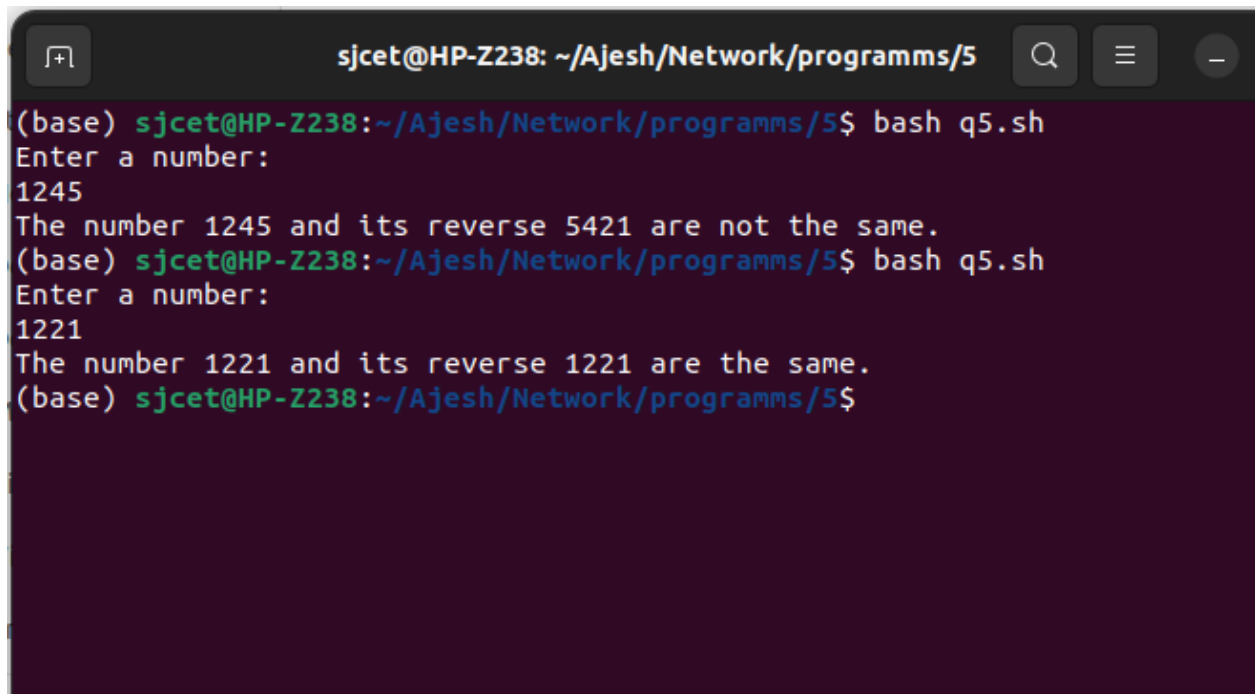
```
echo "Enter a number: "
```

```
read number
```

```
# Reverse the number
```

```
reverse=$(echo $number | rev)
```

```
# Check if the number and its reverse are the same
if [ "$number" -eq "$reverse" ]
then
    echo "The number $number and its reverse $reverse are the same."
else
    echo "The number $number and its reverse $reverse are not the same."
fi
```



```
sjcet@HP-Z238: ~/Ajesh/Network/programms/5
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/5$ bash q5.sh
Enter a number:
1245
The number 1245 and its reverse 5421 are not the same.
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/5$ bash q5.sh
Enter a number:
1221
The number 1221 and its reverse 1221 are the same.
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/5$
```

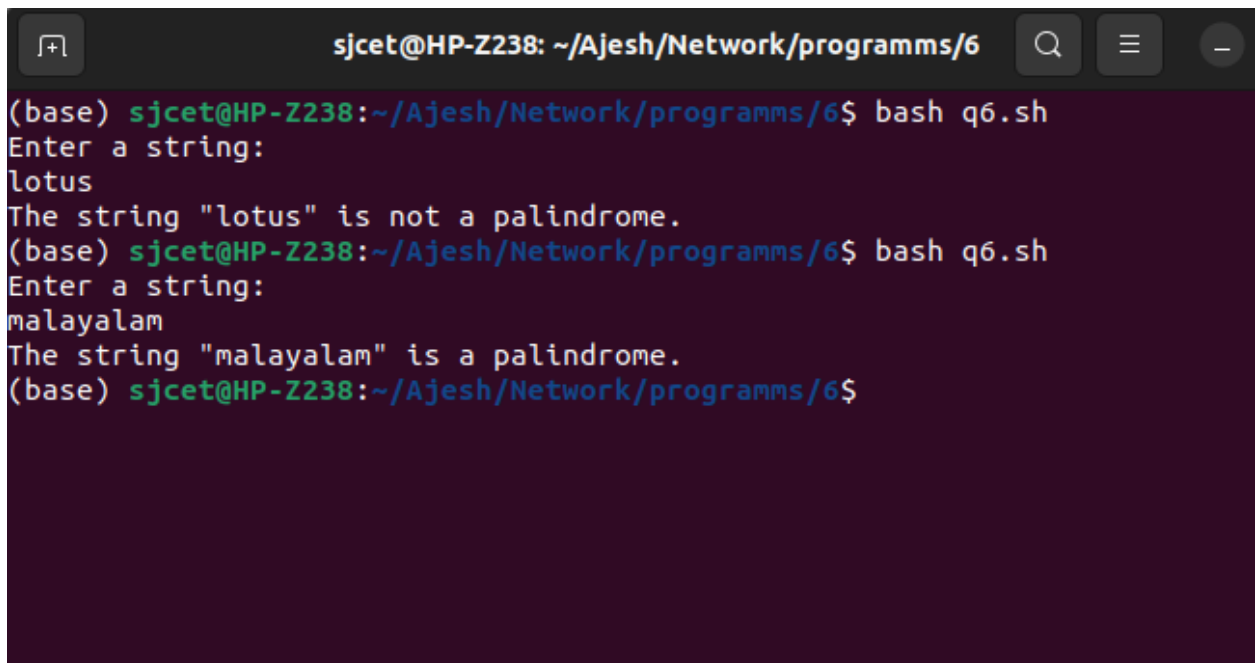
5. Write a Shell program to check the given string is palindrome or not.

CODE

```
echo "Enter a string: "
read string

# Reverse the string
reverse=$(echo "$string" | rev)
```

```
# Check if the string and its reverse are the same
if [ "$string" = "$reverse" ]
then
    echo "The string \"$string\" is a palindrome."
else
    echo "The string \"$string\" is not a palindrome."
fi
```

A terminal window with a dark background and light-colored text. The window title is 'sjcet@HP-Z238: ~/Ajesh/Network/programms/6'. The prompt is '(base) sjcet@HP-Z238:~/Ajesh/Network/programms/6\$'. The user enters 'bash q6.sh'. The script prompts 'Enter a string:' and the user enters 'lotus'. The script outputs 'The string "lotus" is not a palindrome.'. The user enters 'bash q6.sh' again. The script prompts 'Enter a string:' and the user enters 'malayalam'. The script outputs 'The string "malayalam" is a palindrome.'. The prompt returns to '(base) sjcet@HP-Z238:~/Ajesh/Network/programms/6\$'.

6. Write a Shell program to find the sum of odd and even numbers from a set of numbers.

CODE

```
echo "Enter a list of numbers separated by spaces: "
read -a numbers
```

```
# Initialize variables to store the sum of even and odd numbers
even_sum=0
```

```
odd_sum=0
```

```
# Loop through the numbers and add them to the appropriate sum  
for number in "${numbers[@]}"
```

```
do
```

```
    if [  $$(number \% 2)$  -eq 0 ]
```

```
    then
```

```
        even_sum=$((even_sum + number))
```

```
    else
```

```
        odd_sum=$((odd_sum + number))
```

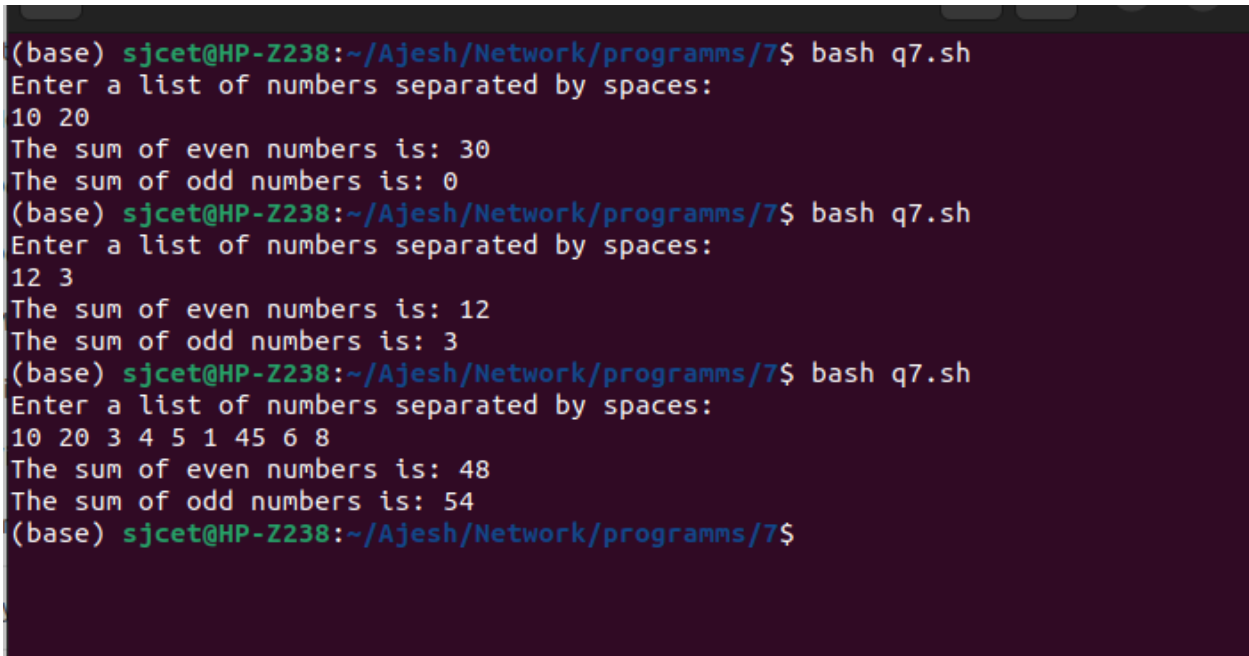
```
    fi
```

```
done
```

```
# Print the results
```

```
echo "The sum of even numbers is: $even_sum"
```

```
echo "The sum of odd numbers is: $odd_sum"
```

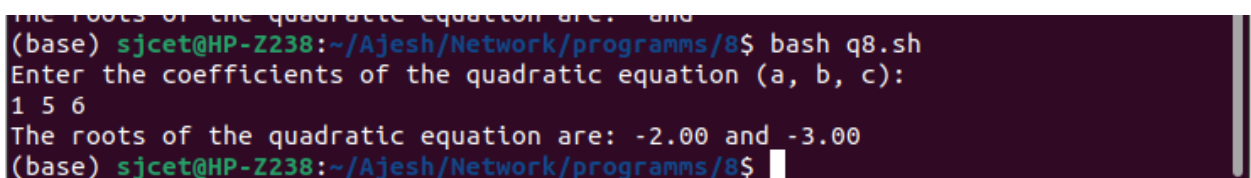


```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/7$ bash q7.sh  
Enter a list of numbers separated by spaces:  
10 20  
The sum of even numbers is: 30  
The sum of odd numbers is: 0  
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/7$ bash q7.sh  
Enter a list of numbers separated by spaces:  
12 3  
The sum of even numbers is: 12  
The sum of odd numbers is: 3  
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/7$ bash q7.sh  
Enter a list of numbers separated by spaces:  
10 20 3 4 5 1 45 6 8  
The sum of even numbers is: 48  
The sum of odd numbers is: 54  
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/7$
```

7. Write a Shell program to find the roots of a quadratic equation.

CODE

```
echo "Enter the coefficients of the quadratic equation (a, b, c): "  
read a b c  
  
# Calculate the discriminant  
discriminant=$((b*b - 4*a*c))  
  
# Check if the discriminant is negative (no real roots)  
if [ $discriminant -lt 0 ]  
then  
    echo "The quadratic equation has no real roots."  
else  
    # Calculate the roots  
    root1=$(echo "scale=2; (-$b + sqrt($discriminant)) / (2*$a)" | bc)  
    root2=$(echo "scale=2; (-$b - sqrt($discriminant)) / (2*$a)" | bc)  
  
    # Print the roots  
    echo "The roots of the quadratic equation are: $root1 and $root2"  
fi
```



The screenshot shows a terminal window with a dark background. The prompt is (base) sjcet@HP-Z238:~/Ajesh/Network/programms/8\$. The user has run bash q8.sh. The program prompts for coefficients a, b, c, and the user has entered 1 5 6. The program then outputs "The roots of the quadratic equation are: -2.00 and -3.00".

```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/8$ bash q8.sh  
Enter the coefficients of the quadratic equation (a, b, c):  
1 5 6  
The roots of the quadratic equation are: -2.00 and -3.00  
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/8$
```

8. Write a Shell program to check the given integer is Armstrong number or not.

CODE

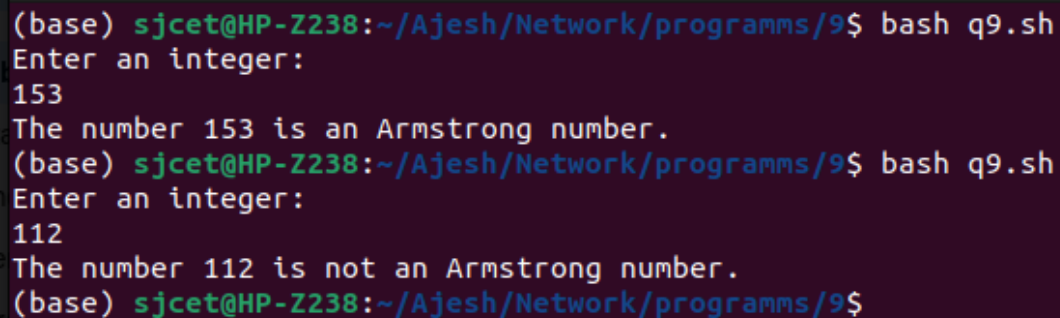
```
echo "Enter an integer: "  
read number
```

```
# Count the number of digits in the number
count=${#number}

# Initialize the sum to 0
sum=0

# Loop through the digits of the number and calculate the sum
for (( i=0; i<count; i++ ))
do
    digit=${number:i:1}
    sum=$((sum + digit**count))
done

# Check if the number is an Armstrong number
if [ "$sum" -eq "$number" ]
then
    echo "The number $number is an Armstrong number."
else
    echo "The number $number is not an Armstrong number."
fi
```



```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/9$ bash q9.sh
Enter an integer:
153
The number 153 is an Armstrong number.
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/9$ bash q9.sh
Enter an integer:
112
The number 112 is not an Armstrong number.
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/9$
```


9. **Write a Shell program to check the given integer is prime or not.**

CODE

```
echo "Enter an integer: "
read number

# Initialize the flag variable to 1
flag=1

# Check if the number is prime
for (( i=2; i<=number/2; i++ ))
do
    if [ $((number%i)) -eq 0 ]
    then
        flag=0
        break
    fi
done

# Output the result
if [ $number -eq 1 ]
then
    echo "1 is neither prime nor composite."
elif [ $flag -eq 1 ]
then
    echo "$number is a prime number."
else
    echo "$number is not a prime number."
fi
```

```
sjcet@HP-Z238: ~/Ajesh/Network/programms/10
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/10$ bash 10.sh
Enter an integer:
121
121 is not a prime number.
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/10$ bash 10.sh
Enter an integer:
26
26 is not a prime number.
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/10$
```

10. Write a Shell program to generate prime numbers between 1 and 50.

CODE

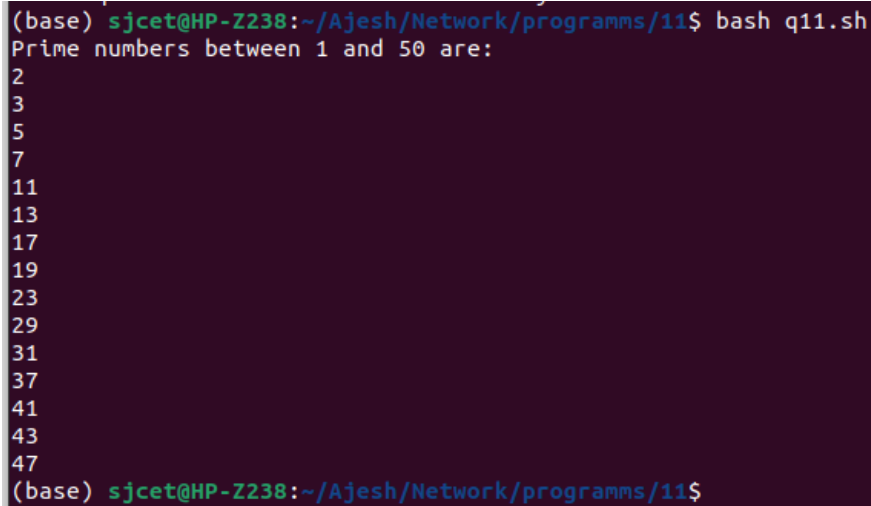
```
echo "Prime numbers between 1 and 50 are:"

# Check each number between 1 and 50 for primality
for (( number=2; number<=50; number++ ))
do
    flag=1

    for (( i=2; i<=number/2; i++ ))
    do
        if [ $((number%i)) -eq 0 ]
        then
            flag=0
            break
        fi
    done

    if [ $flag -eq 1 ]
    then
```

```
    echo $number
fi
done
```



```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/11$ bash q11.sh
Prime numbers between 1 and 50 are:
2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/11$
```

11. Write a Shell program to find the sum of square of individual digits of a number.

CODE

```
echo "Enter a number: "
read number
```

```
# Initialize the sum to 0
sum=0
```

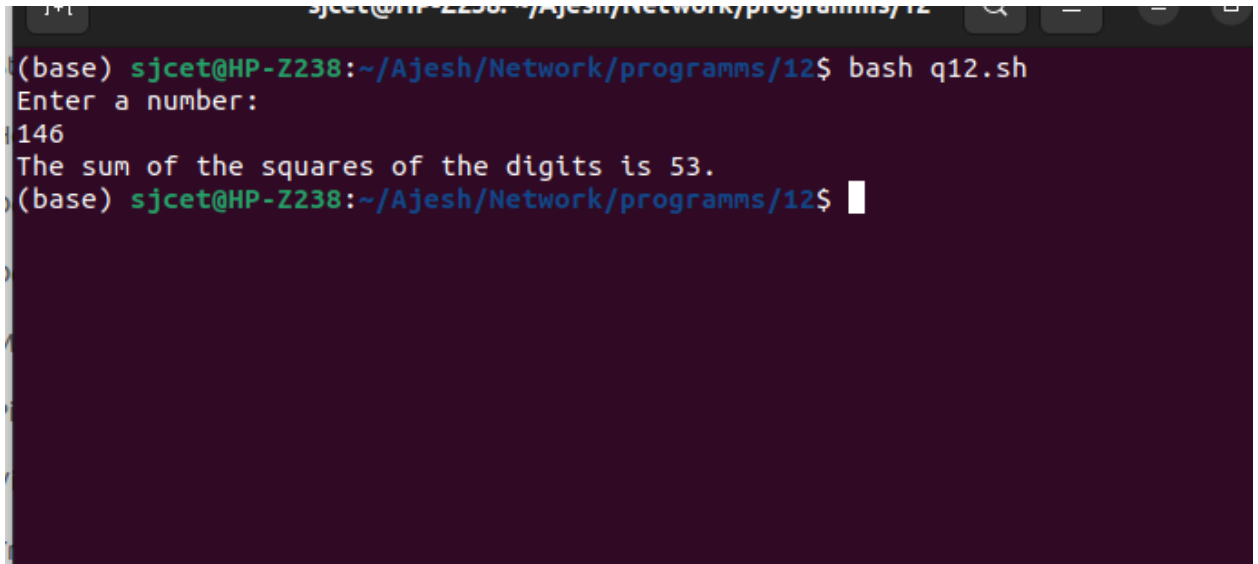
```
# Loop through the digits of the number and calculate the sum of their
squares
```

```
while [ $number -ne 0 ]
do
    digit=$((number % 10))
    sum=$((sum + digit * digit))
```

```
    number=$((number / 10))  
done
```

```
# Output the result
```

```
echo "The sum of the squares of the digits is $sum."
```



```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/12$ bash q12.sh  
Enter a number:  
146  
The sum of the squares of the digits is 53.  
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/12$
```

12. Write a Shell program to count the number of vowels in a line of text.

CODE

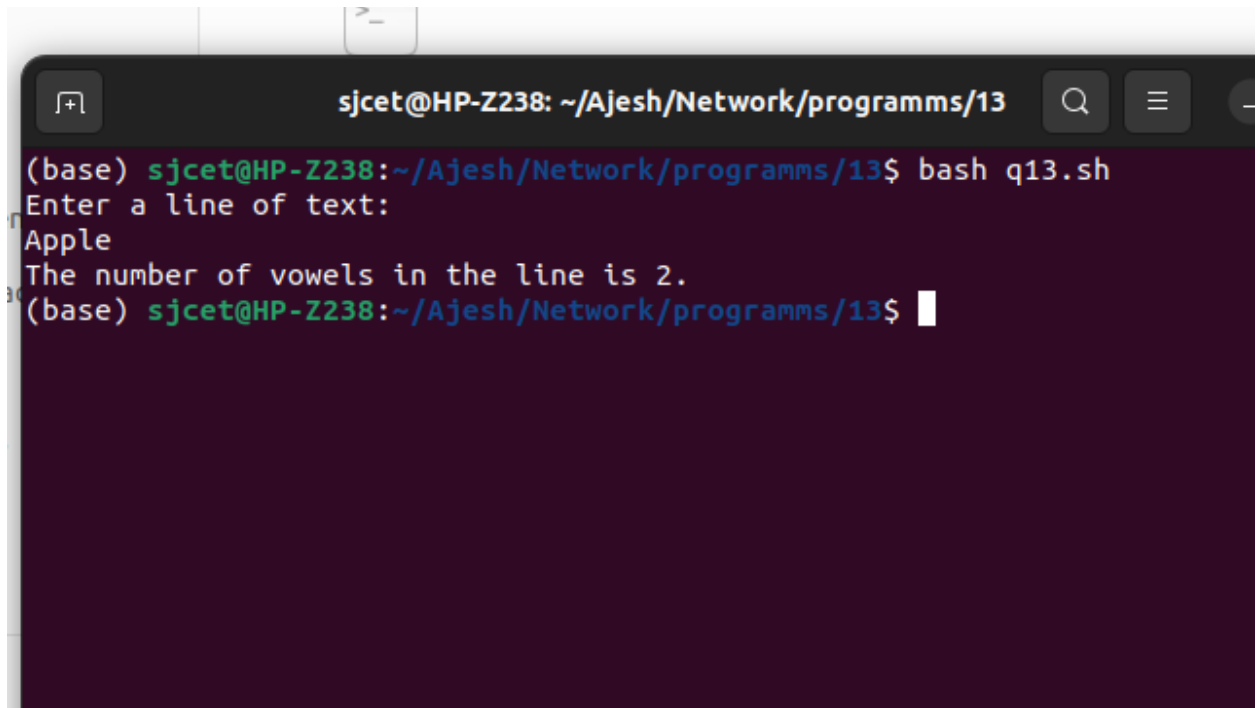
```
echo "Enter a line of text: "  
read line
```

```
# Initialize the vowel count to 0  
count=0
```

```
# Loop through each character of the line and check if it is a vowel  
for (( i=0; i<${#line}; i++ ))  
do
```

```
char=${line:$i:1}
if [[ $char == [aeiouAEIOU] ]]
then
    count=$((count + 1))
fi
done

# Output the result
echo "The number of vowels in the line is $count."
```

A terminal window with a dark purple background. The title bar shows the user 'sjcet' on a machine 'HP-Z238' in the directory '~/Ajesh/Network/programms/13'. The prompt is '(base)'. The user enters 'bash q13.sh'. The script prompts 'Enter a line of text:' and the user enters 'Apple'. The script outputs 'The number of vowels in the line is 2.'. The prompt returns to '(base)'.

```
(base) sjcet@HP-Z238: ~/Ajesh/Network/programms/13$ bash q13.sh
Enter a line of text:
Apple
The number of vowels in the line is 2.
(base) sjcet@HP-Z238: ~/Ajesh/Network/programms/13$
```

13. Write a Shell program to display student grades.

CODE

```
declare -A grades=(
    [Alice]=90
```

```

[Bob]=80
[Charlie]=70
[David]=60
[Emma]=50
)

# Loop through the student names and output their grades
for name in "${!grades[@]}"
do
    echo "$name: ${grades[$name]}%"
Done

```



```

sjcet@HP-Z238: ~/Ajesh/Network/programms/14
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/14$ bash q14.sh
Charlie: 70%
David: 60%
Emma: 50%
Alice: 90%
Bob: 80%
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/14$

```

14. Write a Shell program to find the smallest and largest numbers from a set of numbers.

CODE

```

echo "Enter a list of numbers separated by spaces: "
read numbers

```

```

# Convert the input string to an array of numbers

```

```
IFS=' ' read -ra nums <<< "$numbers"
```

```
# Initialize the min and max variables to the first number in the array
```

```
min=${nums[0]}
```

```
max=${nums[0]}
```

```
# Loop through the remaining numbers in the array and update min and  
max as needed
```

```
for num in "${nums[@]}"
```

```
do
```

```
    if (( num < min )); then
```

```
        min=$num
```

```
    fi
```

```
    if (( num > max )); then
```

```
        max=$num
```

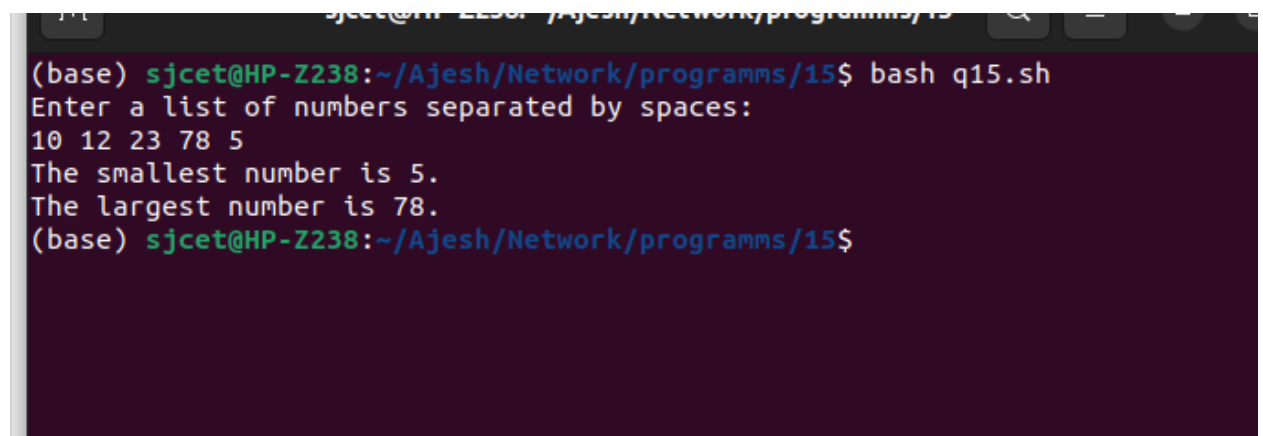
```
    fi
```

```
done
```

```
# Output the result
```

```
echo "The smallest number is $min."
```

```
echo "The largest number is $max."
```

A terminal window with a dark purple background. The prompt is (base) sjcet@HP-Z238:~/Ajesh/Network/programms/15\$. The user has run bash q15.sh. The script prompts "Enter a list of numbers separated by spaces:" and the user has entered "10 12 23 78 5". The script outputs "The smallest number is 5." and "The largest number is 78.".

```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/15$ bash q15.sh
Enter a list of numbers separated by spaces:
10 12 23 78 5
The smallest number is 5.
The largest number is 78.
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/15$
```

15. Write a Shell program to find the smallest digit from a number.

CODE

```
echo "Enter a number: "  
read num  
  
# Initialize the min variable to the first digit of the number  
min=${num:0:1}  
  
# Loop through the remaining digits of the number and update min as  
# needed  
for (( i=1; i<${#num}; i++ ))  
do  
    digit=${num:$i:1}  
    if (( digit < min )); then  
        min=$digit  
    fi  
done  
  
# Output the result  
echo "The smallest digit in $num is $min."
```



A terminal window titled 'sjcet@HP-Z238: ~/Ajesh/Network/programms/16' showing the execution of a script named 'q16.sh'. The script prompts for a number and outputs the smallest digit. The first run uses the input '1456' and outputs 'The smallest digit in 1456 is 1.'. The second run uses the input '5469' and outputs 'The smallest digit in 5469 is 4.'.

```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/16$ bash q16.sh  
Enter a number:  
1456  
The smallest digit in 1456 is 1.  
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/16$ bash q16.sh  
Enter a number:  
5469  
The smallest digit in 5469 is 4.  
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/16$
```


16. Write a Shell program to find the sum of all numbers between 50 and 100, which are divisible by 3 and not divisible by 5.

CODE

```
sum=0
```

```
# Loop through the numbers between 50 and 100
```

```
for (( num=50; num<=100; num++ ))
```

```
do
```

```
    # Check if the number is divisible by 3 and not divisible by 5
```

```
    if (( num % 3 == 0 && num % 5 != 0 )); then
```

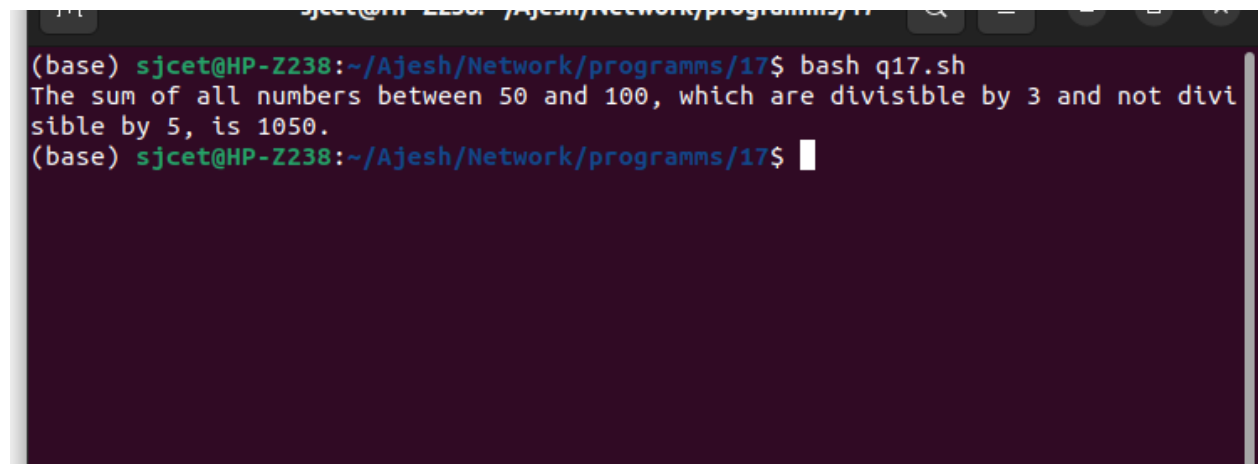
```
        sum=$((sum + num))
```

```
    fi
```

```
done
```

```
# Output the result
```

```
echo "The sum of all numbers between 50 and 100, which are divisible by 3  
and not divisible by 5, is $sum."
```

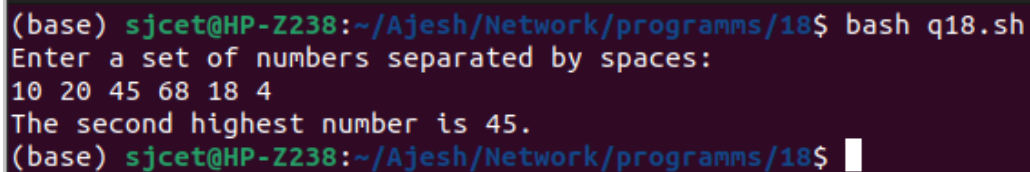
A terminal window with a dark purple background. The prompt is (base) sjcet@HP-Z238:~/Ajesh/Network/programms/17\$. The user has entered bash q17.sh. The output is "The sum of all numbers between 50 and 100, which are divisible by 3 and not divisible by 5, is 1050." followed by a new line and the prompt (base) sjcet@HP-Z238:~/Ajesh/Network/programms/17\$.

```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/17$ bash q17.sh
The sum of all numbers between 50 and 100, which are divisible by 3 and not divi
sible by 5, is 1050.
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/17$
```

17. Write a Shell program to find the second highest number from a set of numbers.

CODE

```
echo "Enter a set of numbers separated by spaces: "  
read numbers  
  
# Convert the space-separated string to an array  
arr=($numbers)  
  
# Sort the array in descending order  
sorted_arr=$(echo "${arr[@]}" | tr " " "\n" | sort -rn))  
  
# Output the second highest number  
echo "The second highest number is ${sorted_arr[1]}."
```



```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/18$ bash q18.sh  
Enter a set of numbers separated by spaces:  
10 20 45 68 18 4  
The second highest number is 45.  
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/18$
```

18. Write a Shell program to find the sum of digits of a number using function.

CODE

```
# Define the function to calculate the sum of digits  
sum_of_digits() {  
    num=$1  
    sum=0  
    while [ $num -gt 0 ]
```

```

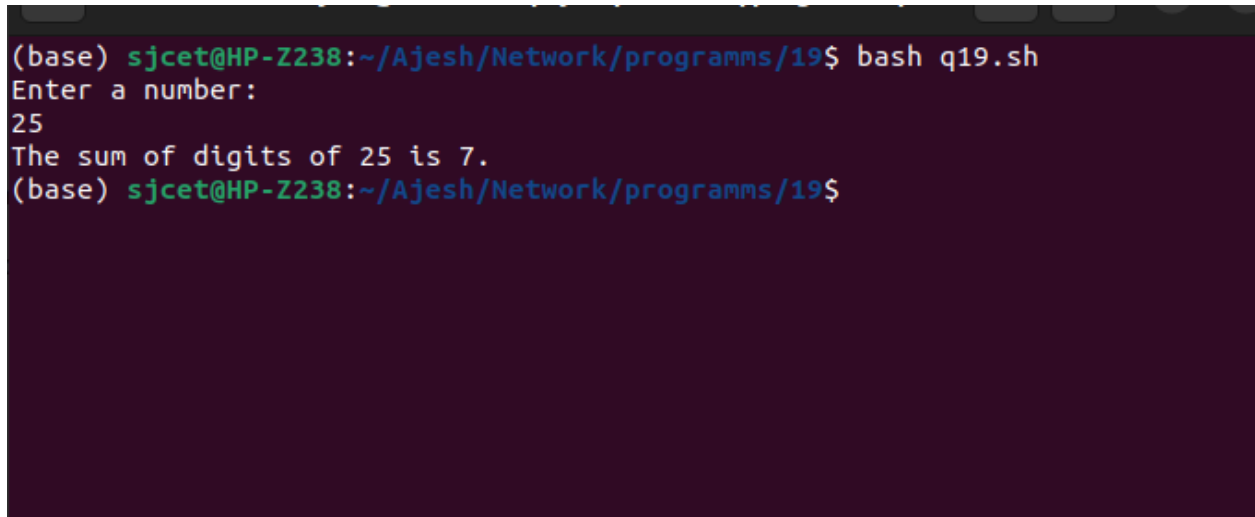
do
    digit=$((num % 10))
    sum=$((sum + digit))
    num=$((num / 10))
done
echo $sum
}

# Prompt the user to enter a number
echo "Enter a number: "
read num

# Call the function to calculate the sum of digits
result=$(sum_of_digits $num)

# Output the result
echo "The sum of digits of $num is $result."

```



```

(base) sjcet@HP-Z238:~/Ajesh/Network/programms/19$ bash q19.sh
Enter a number:
25
The sum of digits of 25 is 7.
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/19$

```

19. Write a Shell program to print the reverse of a number using function.

CODE

```
# Define the function to reverse a number
```

```
reverse_number() {  
    num=$1  
    rev=0  
    while [ $num -gt 0 ]  
    do  
        digit=$((num % 10))  
        rev=$((rev * 10 + digit))  
        num=$((num / 10))  
    done  
    echo $rev  
}
```

```
# Prompt the user to enter a number
```

```
echo "Enter a number: "
```

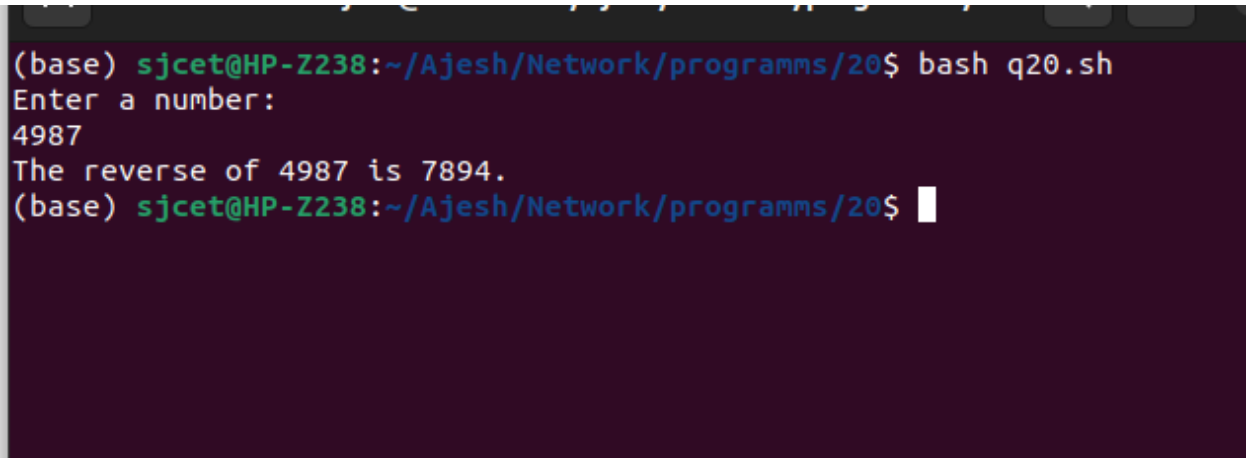
```
read num
```

```
# Call the function to reverse the number
```

```
result=$(reverse_number $num)
```

```
# Output the result
```

```
echo "The reverse of $num is $result."
```



```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/20$ bash q20.sh  
Enter a number:  
4987  
The reverse of 4987 is 7894.  
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/20$
```

20. Write a Shell program to find the factorial of a number using for loop.

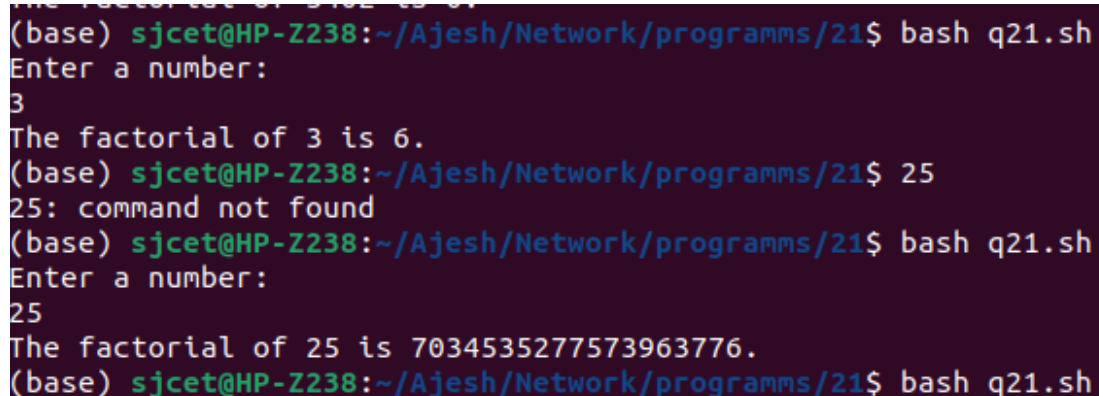
CODE

```
# Prompt the user to enter a number
echo "Enter a number: "
read num

# Initialize the factorial to 1
factorial=1

# Calculate the factorial using a for loop
for (( i=1; i<=$num; i++ ))
do
    factorial=$((factorial * i))
done

# Output the result
echo "The factorial of $num is $factorial."
```



```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/21$ bash q21.sh
Enter a number:
3
The factorial of 3 is 6.
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/21$ 25
25: command not found
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/21$ bash q21.sh
Enter a number:
25
The factorial of 25 is 7034535277573963776.
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/21$ bash q21.sh
```

21. Write a Shell program to generate Fibonacci series.

CODE

```
# Prompt the user to enter the number of terms to generate
echo "Enter the number of terms to generate: "
```

```
read num
```

```
# Initialize the first two terms of the series
```

```
a=0
```

```
b=1
```

```
# Output the first two terms
```

```
echo -n "$a $b"
```

```
# Generate the rest of the series using a loop
```

```
for (( i=3; i<=$num; i++ ))
```

```
do
```

```
    # Calculate the next term
```

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

```
    # Output the next term
```

```
    echo -n " $c"
```

```
    # Shift the values of a and b to prepare for the next iteration
```

```
    a=$b
```

```
    b=$c
```

```
done
```

```
Echo
```

```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/22$ bash q22.sh
Enter the number of terms to generate:
6
0 1 1 2 3 5
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/22$
```

22. Write a shell script, which receives two filenames as arguments. It checks whether the two files contents are same or not. If they are same then second file is deleted.

CODE

```
if [ $# -ne 2 ]; then
    echo "Usage: $0 file1 file2"
    exit 1
fi

if cmp -s "$1" "$2"; then
    echo "The contents of $1 and $2 are the same."
    rm "$2"
else
    echo "The contents of $1 and $2 are different."
fi
```

```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/23$ bash q23.sh
Usage: q23.sh file1 file2
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/23$
```


23. Write a Menu driven Shell script that Lists current directory, Prints Working Directory, displays Date and displays Users logged in

CODE

```
echo "Select an option:"  
echo "1. List current directory"  
echo "2. Print working directory"  
echo "3. Display date"  
echo "4. Display users logged in"
```

```
read option
```

```
case $option in
```

```
1)
```

```
ls -l
```

```
;;
```

```
2)
```

```
pwd
```

```
;;
```

```
3)
```

```
date
```

```
;;
```

```
4)
```

```
who
```

```
;;
```

```
*)
```

```
echo "Invalid option selected"
```

```
;;
```

```
Esac
```

```

(base) sjcet@HP-Z238:~/Ajesh/Network/programms/24$ bash q24.sh
Select an option:
1. List current directory
2. Print working directory
3. Display date
4. Display users logged in
1
total 4
-rw-rw-r-- 1 sjcet sjcet 315 Apr  3 15:12 q24.sh
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/24$
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/24$ bash q24.sh
Select an option:
1. List current directory
2. Print working directory
3. Display date
4. Display users logged in
2
/home/sjcet/Ajesh/Network/programms/24
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/24$ bash q24.sh
Select an option:
1. List current directory
2. Print working directory
3. Display date
4. Display users logged in
3
Monday 03 April 2023 03:12:49 PM IST
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/24$ bash q24.sh
Select an option:
1. List current directory
2. Print working directory
3. Display date
4. Display users logged in
4
sjcet      tty2          2023-04-03 14:49 (tty2)
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/24$

```

24.Shell script to check executable rights for all files in the current directory, if a file does not have the execute permission then make it executable.

CODE

```
find . -type f | while read file; do
```

```
if [ ! -x "$file" ]; then
```

```
    chmod +x "$file"
```

```
    echo "Made $file executable"
```

```
fi
```

done

```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/25$ bash q25.sh
Made ./q25.sh executable
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/25$
```

25. Write a Shell program to generate all combinations of 1, 2, and 3 using loop.

CODE

```
or i in 1 2 3; do
  for j in 1 2 3; do
    for k in 1 2 3; do
      echo "$i$j$k"
    done
  done
done
```

```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/26$ bash q26.sh
111
112
113
121
122
123
131
132
133
211
212
213
221
222
223
231
232
233
311
312
313
321
322
323
331
332
333
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/26$
```

26. Write a Shell program to create the number series.

```
1
2      3
4      5      6
7      8      9      10
```

CODE

```
count=1
```

```

for (( i=1; i<=4; i++ ))
do
    for (( j=1; j<=i; j++ ))
    do
        echo -n "$count "
        count=$((count+1))
    done
    echo ""
done

```



```

sjcet@HP-Z238: ~/Ajesh/Network/programms/27
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/27$ bash q27.sh
1
2 3
4 5 6
7 8 9 10
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/27$

```

27. Write a Shell program to create Pascal's triangle.

CODE

```

#!/bin/bash

# set the number of rows
echo "Enter the number of rows to generate for Pascal's triangle:"
read rows

# initialize the first row
row=1
echo $row

```

```

# loop over the remaining rows
for ((i=1; i<$rows; i++)); do
    # initialize the row with the left-most element
    prev_row=($row)
    row=${prev_row[0]}

    # loop over the remaining elements in the row
    for ((j=1; j<=i; j++)); do
        # calculate the current element
        current=$((prev_row[j-1] + prev_row[j]))

        # append the current element to the row
        row="$row $current"
    done

    # print the row
    echo $row
done

```

```

(base) sjcet@HP-Z238:~/Ajesh/Network/programms/28$ bash q28.sh
Enter the number of rows to generate for Pascal's triangle:
3
1
1 1
1 2 1
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/28$ bash q28.sh
Enter the number of rows to generate for Pascal's triangle:
5
1
1 1
1 2 1
1 3 3 1
1 4 6 4 1
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/28$

```

28. Write a Decimal to Binary Conversion Shell Script

CODE

```
#!/bin/bash
```

```
# Prompt the user for the decimal number to convert
```

```
echo "Enter a decimal number: "
```

```
read decimal
```

```
# Convert the decimal number to binary
```

```
binary=""
```

```
while [ $decimal -gt 0 ]; do
```

```
    remainder=$((decimal % 2))
```

```
    binary="$remainder$binary"
```

```
    decimal=$((decimal / 2))
```

```
done
```

```
# Print the binary number
```

```
echo "The binary equivalent is: $binary"
```



```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/29$ bash q29.sh
Enter a decimal number:
23
The binary equivalent is: 10111
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/29$ bash q29.sh
Enter a decimal number:
9
The binary equivalent is: 1001
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/29$
```

29. Write a Shell Script to Check Whether a String is Palindrome or not

CODE

```
#!/bin/bash

# Prompt the user for the string to check
echo "Enter a string: "
read string

# Reverse the string
reverse=$(echo $string | rev)

# Check if the string is equal to its reverse
if [ "$string" == "$reverse" ]; then
    echo "$string is a palindrome."
else
    echo "$string is not a palindrome."
Fi
```

```

(base) sjcet@HP-Z238:~/Ajesh/Network/programms/30$ bash q30.sh
Enter a string:
Apple
Apple is not a palindrome.
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/30$ bash q30.sh
Enter a string:
Malayalam
Malayalam is not a palindrome.
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/30$ bash q30.sh
Enter a string:
malayalam
malayalam is a palindrome.
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/30$ █

```

30. Write a shell script to find out the unique words in a file and also count the occurrence of each of these words.

CODE

```
#!/bin/bash
```

```
# Prompt the user for the file name
```

```
echo "Enter the file name: "
```

```
read file
```

```
# Check if the file exists
```

```
if [ ! -f "$file" ]; then
```

```
    echo "File not found."
```

```
    exit 1
```

```
fi
```

```
# Convert the contents of the file to lowercase and replace all non-
alphanumeric characters with spaces
```

```
contents=$(tr '[:upper:]' '[:lower:]' < $file | sed 's/[^a-z0-9]/ /g')
```

```
# Create an array of words from the file contents
```

```
words=($contents)
```

```
# Loop through the array of words and count their occurrences
```

```
declare -A count
```

```
for word in "${words[@]}"; do
```

```
    if [ -n "$word" ]; then
```

```
        ((count[$word]++))
```

```
    fi
```

```
done
```

```
# Print the unique words and their counts
```

```
echo "Unique words in $file:"
```

```
for word in "${!count[@]}"; do
```

```
    echo "$word: ${count[$word]}"
```

```
done
```

```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/31$ bash q31.sh
Enter the file name:
q31.sh
Unique words in q31.sh:
non: 1
not: 1
file: 10
with: 1
the: 9
count: 5
enter: 1
all: 1
lower: 1
if: 3
in: 3
and: 3
done: 2
do: 2
found: 1
echo: 4
fi: 2
spaces: 1
9: 1
1: 1
an: 1
s: 1
n: 1
g: 1
f: 1
contents: 4
a: 2
lowercase: 1
bash: 1
through: 1
exists: 1
words: 6
z0: 1
array: 2
then: 2
their: 2
for: 3
check: 1
loop: 1
```

31. Write a shell script to get the total count of the word "Linux" in all the ".txt" files and also across files present in subdirectories.

CODE

```
#!/bin/bash

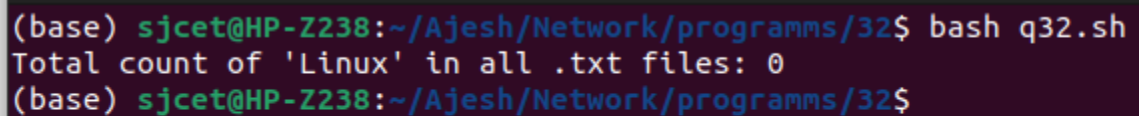
# Set the search directory
search_dir="."

# Find all ".txt" files in the search directory and its subdirectories
files=$(find "$search_dir" -type f -name "*.txt")

# Initialize the count
count=0

# Loop through each file and count the occurrences of "Linux"
for file in $files; do
    occurrences=$(grep -o "Linux" "$file" | wc -l)
    count=$((count + occurrences))
done

# Print the total count
echo "Total count of 'Linux' in all .txt files: $count"
```



```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/32$ bash q32.sh
Total count of 'Linux' in all .txt files: 0
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/32$
```

32. Write a shell script to validate password strength. Here are a few assumptions for the password string.

Length – minimum of 8 characters.

Contain both alphabet and number.

Include both the small and capital case letters.

CODE

```
#!/bin/bash
```

```
# Prompt user to enter password
```

```
echo "Please enter a password: "
```

```
read password
```

```
# Check if password length is at least 8 characters
```

```
if [ ${#password} -lt 8 ]
```

```
then
```

```
    echo "Password must have a minimum of 8 characters."
```

```
    exit 1
```

```
fi
```

```
# Check if password contains both alphabet and number
```

```
if ! [ "$password" =~ [[:alpha:]] && "$password" =~ [[:digit:]] ]
```

```
then
```

```
    echo "Password must contain both alphabet and number."
```

```
    exit 1
```

```
fi
```

```
# Check if password includes both small and capital case letters
```

```
if ! [ "$password" =~ [[:lower:]] && "$password" =~ [[:upper:]] ]
```

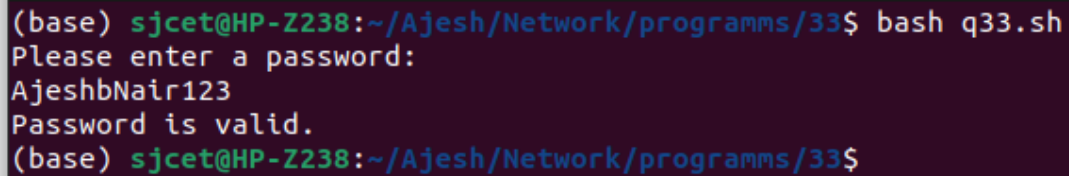
```
then
```

```
    echo "Password must include both small and capital case letters."
```

```
    exit 1
```

```
fi
```

```
# If all checks pass, password is valid
echo "Password is valid."
exit 0
```

A terminal window with a dark purple background. The prompt is (base) sjcet@HP-Z238:~/Ajesh/Network/programms/33\$. The user enters bash q33.sh. The script prompts "Please enter a password:", the user enters "AjeshbNair123", and the script outputs "Password is valid." followed by the prompt (base) sjcet@HP-Z238:~/Ajesh/Network/programms/33\$.

```
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/33$ bash q33.sh
Please enter a password:
AjeshbNair123
Password is valid.
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/33$
```

33. Write a shell script to print the count of files and subdirectories in the specified directory.

CODE

```
#!/bin/bash
```

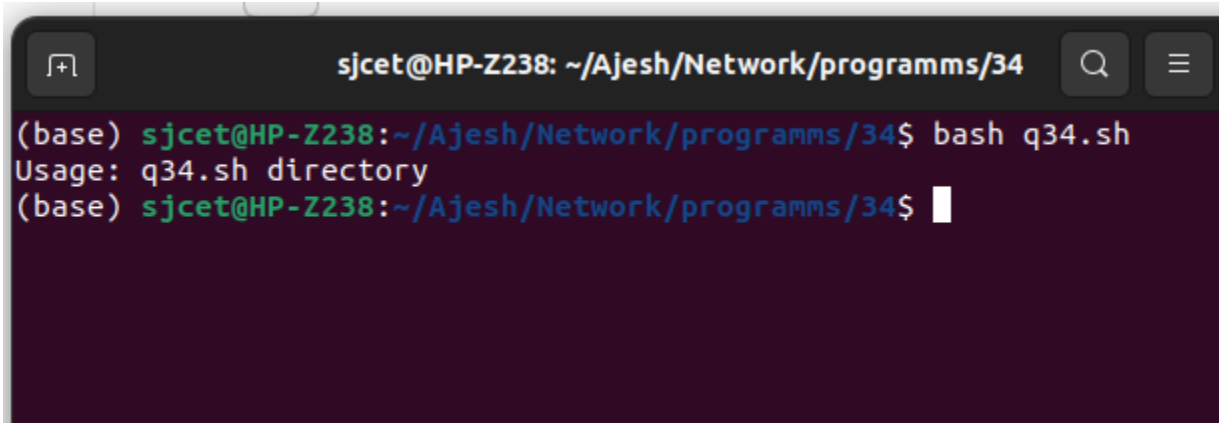
```
if [ $# -eq 0 ]; then
    echo "Usage: $0 directory"
    exit 1
fi
```

```
directory=$1
```

```
if [ ! -d $directory ]; then
    echo "Error: $directory is not a directory"
    exit 1
fi
```

```
num_files=$(find $directory -maxdepth 1 -type f | wc -l)
num_dirs=$(find $directory -maxdepth 1 -type d | wc -l)
```

```
echo "Number of files in $directory: $num_files"
echo "Number of directories in $directory: $((num_dirs - 1))"
```

A terminal window with a dark background. The title bar shows 'sjcet@HP-Z238: ~/Ajesh/Network/programms/34'. The prompt is '(base) sjcet@HP-Z238:~/Ajesh/Network/programms/34\$'. The user has entered 'bash q34.sh'. The output is 'Usage: q34.sh directory'. The prompt is now '(base) sjcet@HP-Z238:~/Ajesh/Network/programms/34\$' with a cursor.

```
sjcet@HP-Z238: ~/Ajesh/Network/programms/34
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/34$ bash q34.sh
Usage: q34.sh directory
(base) sjcet@HP-Z238:~/Ajesh/Network/programms/34$
```

34. Write a shell script to reverse the list of strings and reverse each string further in the list.

CODE

```
#!/bin/bash
```

```
# prompt user for list of strings
```

```
read -p "Enter a list of strings separated by spaces: " string_list
```

```
# split string_list into an array of strings
```

```
read -a strings <<< "$string_list"
```

```
# reverse the array of strings
```

```

for (( i=${#strings[@]}-1; i>=0; i-- ))
do
    # reverse each string in the array
    reversed_string=""
    for (( j=${#strings[i]}-1; j>=0; j-- ))
    do
        reversed_string="$reversed_string${strings[i]:$j:1}"
    done

    # replace the original string with the reversed string
    strings[i]=$reversed_string
done

# print the reversed and reversed strings
echo "${strings[@]}"

```

```

(base) sjcet@HP-Z238: ~/Ajesh/Network/programms/35$ bash q35.sh
Enter a list of strings separated by spaces: abin
niba
(base) sjcet@HP-Z238: ~/Ajesh/Network/programms/35$ bash q35.sh
Enter a list of strings separated by spaces: abin abhi ajesh
niba ihba hseja
(base) sjcet@HP-Z238: ~/Ajesh/Network/programms/35$

```