Assignment-9

1. Write a shell script using if...else to check if a number is even or odd.

```
#!/bin/bash

echo "Enter a number:"
read num

remainder=$((num % 2))

if [ $remainder -eq 0 ]; then
   echo "$num is an even number."
else
   echo "$num is an odd number."
fi
```

2. Explain the difference between if and case statements in bash

Point	if Statement	case Statement
Purpose	Tests boolean/logical conditions	Matches one variable against multiple fixed patterns
Condition Types	Supports complex expressions, numeric and string tests	Matches strings or patterns
Syntax Complexity	Can become complex with manyelifbranches	Cleaner and more readable for multiple values
Use Case Example	Checking ranges, file existence, logical tests	Selecting action based on exact value or pattern

Point	if Statement	case Statement
Pattern Matching Support	No	Yes, supports wildcard and multiple pattern options

3. Write a script to find the largest of three numbers entered by the user.

```
#!/bin/bash

echo "Enter first number:"
read num1
echo "Enter second number:"
read num2
echo "Enter third number:"
read num3

if [ "$num1" -gt "$num2" ] && [ "$num1" -gt "$num3" ]; then
  echo "$num1 is the largest number."
elif [ "$num2" -gt "$num1" ] && [ "$num2" -gt "$num3" ]; then
  echo "$num2 is the largest number."
else
  echo "$num3 is the largest number."
fi
```

4. How do you use a for loop to traverse an array in bash? Give an example. The array is defined as arr=(123, “Abs”, -2.3, ‘A’, 23.56, 0).

In bash loop through an array's elements using a for loop with the syntax:

- for element in "${array[@]}"; do
  done

Ex:-

```
#!/bin/bash

arr=(123 "Abs" -2.3 'A' 23.56 0)

for item in "${arr[@]}"; do
 echo "$item"
done
```

5. **Write a shell script to loop through all files in the current directory and display their names.**

```
#!/bin/bash

for file in *; do
 if [ -f "$file" ]; then
  echo "$file"
 fi
done
```

6. **What is the difference between while and until loops in bash?**

Point	while Loop	until Loop
Condition check	Runswhilethe condition is true	Runsuntilthe condition becomes true
Loop execution	Executes as long as condition evaluates to true	Executes as long as condition evaluates to false
Typical use case	Repeat tasks when a condition remains true	Repeat tasks until a condition is met
Logic relation	Loop continues on true condition	Loop continues on false condition
Example condition	while [$count -le 5]; do ... done	until [$count -gt 5]; do ... done

7. Write a countdown timer script using a while loop.

```
#!/bin/bash

echo "Enter countdown time in seconds:"
read time
```

```
while [ $time -gt 0 ]
do
  echo "Time left: $time seconds"
  sleep 1
  ((time--))
done

echo "Countdown finished!"
```

8. How do you use break and continue statements in loops? Give examples.
   - break:

     Exits the current loop immediately.
     If nested loops exist, break n exits the n-th enclosing loop.

   - continue:

     Skips the remaining commands in the current loop iteration.
     Moves control to the next iteration of the loop.
     Can also use continue n to apply to outer loops when nested.

```
Ex:- i=0
while [[ $i -lt 5 ]]; do
 echo "Number: $i"
 ((i++))
 if [[ $i -eq 2 ]]; then
  break
 fi
done
echo 'All Done!'
```

9. Write a script to check if a file exists or not using the if and else loop.

```
#!/bin/bash
echo "Enter the filename:"
read filename

if [ -e "$filename" ]
then
  echo " File '$filename' exists."
else
  echo " File '$filename' does not exist."
fi
```

10. Write a script to calculate factorial of a number using for loop.

```
#!/bin/bash
echo "Enter a number:"
read num

fact=1

for (( i=1; i<=num; i++ ))
do
   fact=$((fact * i))
done

echo "The factorial of $num is $fact"
```