Assignment-7

1. What is a bash shell script? Give one example

   A **bash shell script** is a text file containing a sequence of commands that are executed by the Bash shell interpreter line-by-line. It automates repetitive tasks and simplifies complex command sequences by grouping them into a single executable script.

   Ex:-

   ```
   #!/bin/bash
   echo "Hello, user!"
   echo "Today is: $(date)"
   echo "Have a great day!"
   ```

2. Write a simple shell script to print “Hello World”.

   ```
   #!/bin/bash
   echo "Hello World"
   ```

3. What is the purpose of comments (#) in a shell script?

   The purpose of comments (#) in a shell script is to provide human-readable explanations or annotations within the script, which are ignored by the shell interpreter during execution. Comments help make the script more understandable and maintainable by documenting the code's logic, purpose, and any important details. They also assist in troubleshooting and collaboration by clarifying the intent behind code segments.

4. How do you declare variables (int, float, double, string, Boolean, and char in a shell script?

   ```
   #!/bin/bash

   declare -i myint=10
   mystring="Hello"
   is_true=1
   mychar="A"

   echo "Integer: $myint"
   echo "String: $mystring"
   echo "Boolean (1=true, 0=false): $is_true"
   echo "Char: $mychar"
   ```

5. Write a shell script to display the current date and time of the system.

```
current_date_time=$(date +"%Y-%m-%d %H:%M:%S")
echo "Current date and time: $current_date_time"
```

6. Explain the difference between a constant and a variable in bash script.

Aspect	Variable	Constant
Definition	A named storage whose value can change during execution	A named storage whose value does not change once set
Mutability	Mutable; value can be reassigned anytime	Immutable; value cannot be changed after being set
Declaration	var=value(simple assignment)	Usereadonlyordeclare -rkeywords
Example	name="John"	readonly MAX_VALUE=100
Usage	Store data that may vary during script execution	Store fixed values like configuration constants
Enforcement	No restriction on reassignment	Attempts to change cause an error
Typical use cases	Loop counters, user input, temporary data	Fixed parameters, constants like Pi, max limits

7. Write a shell script to read two integer number from the user and compute the sum of both the number.

```
#!/bin/bash
read -p "Enter first integer: " num1
```

```
read -p "Enter second integer: " num2
sum=$((num1 + num2))
echo "Sum of $num1 and $num2 is: $sum"
```

8. What is the use of source command in shell scripting?

The **purpose of the source command** in shell scripting is to execute commands from a file within the current shell environment, rather than starting a new subshell. This means any variables, functions, or environment changes made by the sourced script affect the current shell session immediately.

Key points about source command:

- Runs the script in the current shell process.
- Changes environment variables and functions for the current shell.
- Useful for loading configuration files, setting environment variables, or defining functions without spawning a new shell.

9. How can you debug a shell script? Give two methods.

1) Using the set command with the -x option: Running the script with set -x enabled causes the shell to print each command and its arguments to the terminal as they are executed. This helps in tracking the script's flow and catching errors in command execution. You can enable it temporarily in the script by placing set -x before the section to debug and disable it with set +x afterward. Alternatively, invoke the script with bash -x script.sh or add -x to the shebang line like #!/bin/bash -x for tracing the whole script.

2) Using verbose mode with the -v option: This prints each line of the script to the terminal as it is read, showing what the shell is processing. Similar to -x, this can be enabled in the shebang or with the set -v command inside the script. It helps to see the raw script lines before execution.

10. Write a bash script to create and delete a file.

```
#!/bin/bash
file_name="myfile.txt"
touch "$file_name"
echo "File '$file_name' created."
rm "$file_name"
echo "File '$file_name' deleted."
```