Task:4

Tasks:

Debugging Scripts :

Objective: Write a buggy shell script and use debugging to identify and fix the issue using the "set" and their options. Check which options is used in which scenario, document it with all the information.

```
sunny28@Linux:~/Desktop/debug_script$ chmod -x debug_script
chmod: cannot access 'debug_script': No such file or directory
sunny28@Linux:~/Desktop/debug_script$ chmod -x debug_script.sh
sunny28@Linux:~/Desktop/debug_script$ bash -x debug_script.sh
+ set -x
+ calculteee_square
debug_script.sh: line 6: calculteee_square: command not found
+ set +x
the square of  is: 0
sunny28@Linux:~/Desktop/debug_script$ 
```

set -x

calculate_square() {

   result=$((number * number))

   echo "The square of $number is: $result"

}

 # Intentional bug: Misspelled variable name for the function call

calculteee_square

set +x

 # Call the function to calculate the square

calculate_square

Debugging with set -e:

set -e makes the script exit immediately if any command it runs exits with a non-zero status.

\#!/bin/bash

set -e

\# ... (rest of the script)

Running the script with set -e will terminate the script upon encountering the error in the misspelled function call.

Debugging with set -x:

set -x prints each command and its arguments to the standard error output before executing them.

\#!/bin/bash

set -x

Running the script with set -x will show the commands being executed, helping to identify where the script fails.

Debugging with set -u:

set -u treats unset variables as an error.

\#!/bin/bash

set -u

... (rest of the script)

Running the script with set -u will catch cases where variables are used before being assigned, improving script robustness.

```
sunny28@Linux:~/Desktop/debug_script$ chmod -x debug_script2.sh
sunny28@Linux:~/Desktop/debug_script$ bash -x debug_script2.sh
+ set -x
+ read -p 'enter a number ' number
enter a number 5
+ calculate_square
+ result=25
+ echo 'the square of 5 is: 25'
the square of 5 is: 25
+ set +x
sunny28@Linux:~/Desktop/debug_script$
```