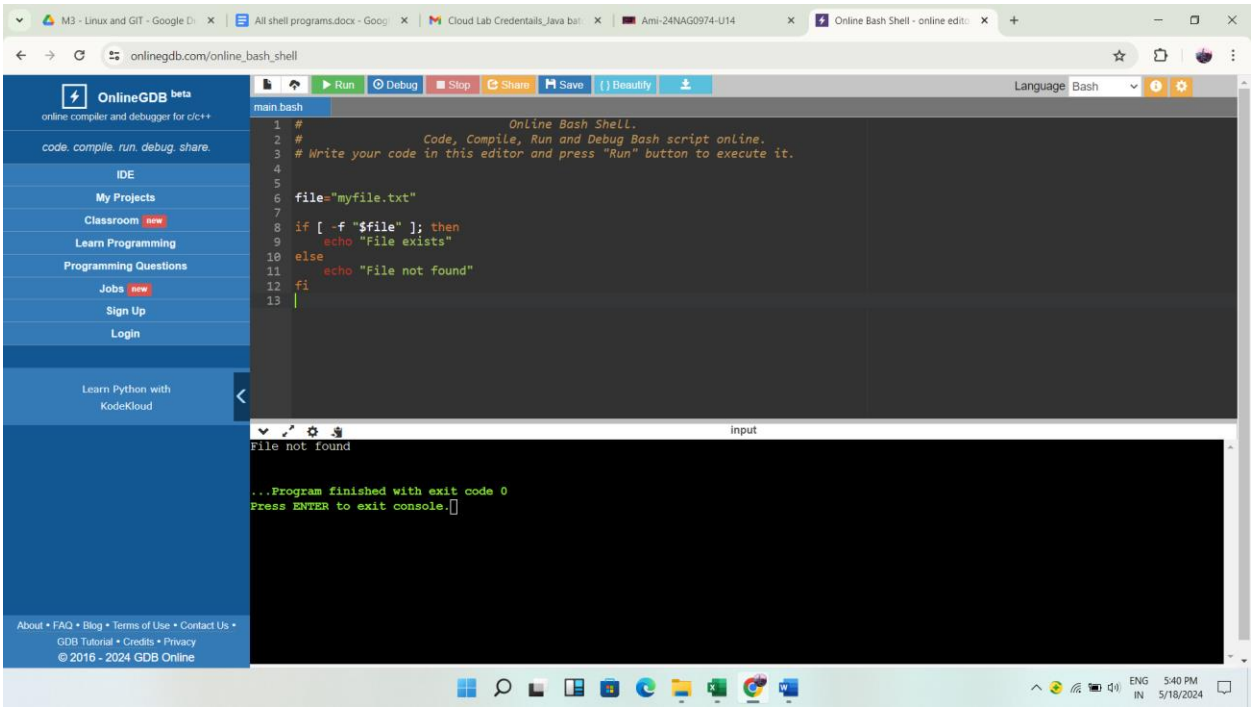
Assignment Shell-Scripting

Que 1.

Ensure the script checks if a specific file (e.g., myfile.txt) exists in the current directory. If it exists, print "File exists", otherwise print "File not found".

Ans:

```
file="myfile.txt"


if [ -f "$file" ]; then
   echo "File exists"
else
   echo "File not found"
fi
```

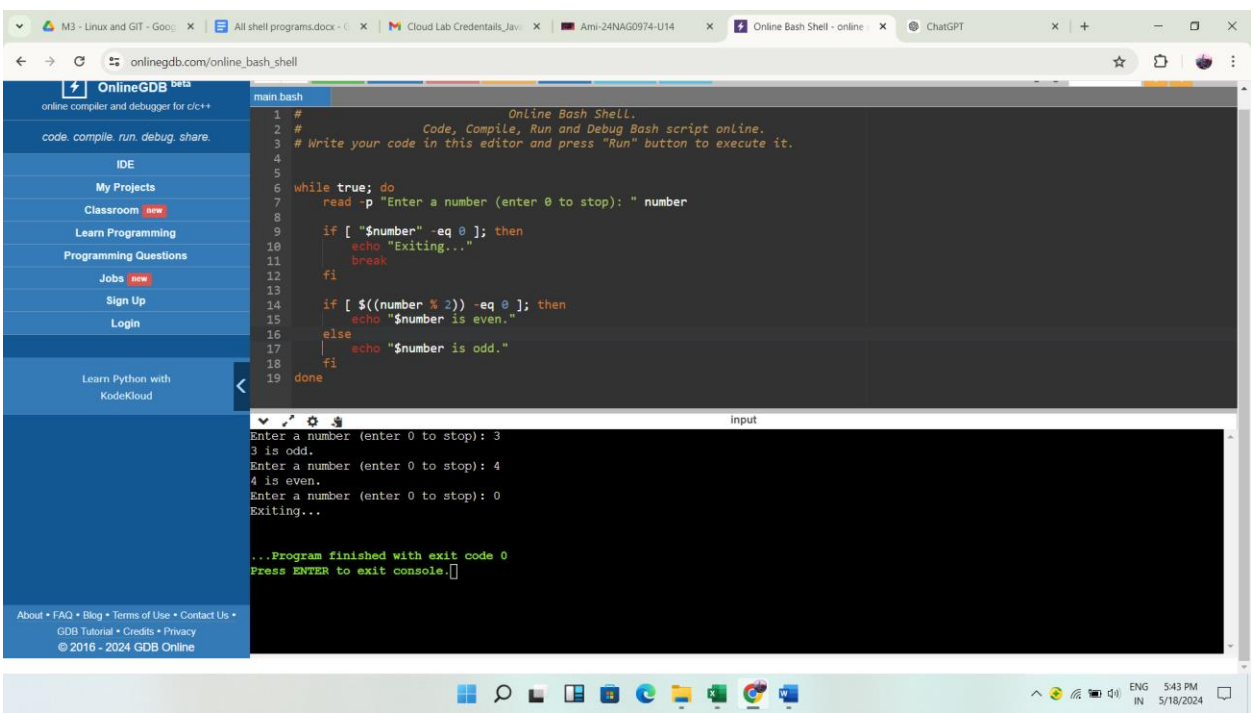


Que 2.

Write a script that reads numbers from the user until they enter '0'. The script should also print whether each number is odd or even.

Ans.

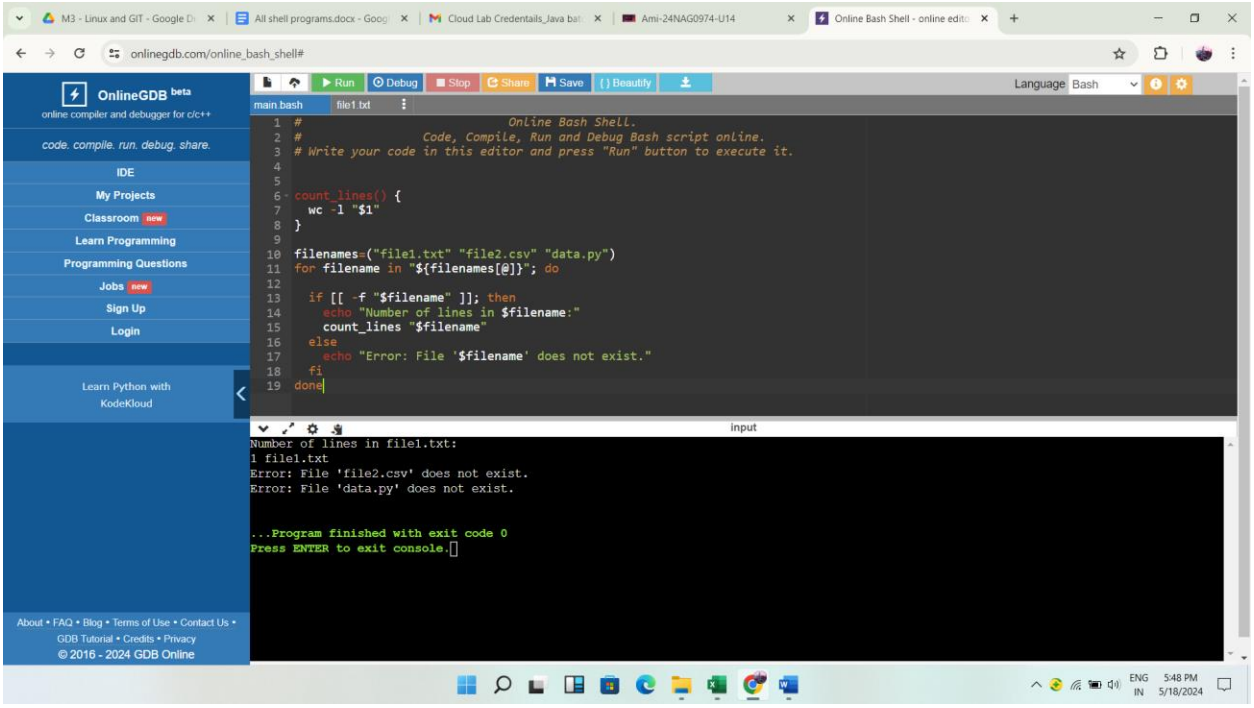
```
while true; do
   read -p "Enter a number (enter 0 to stop): " number

   if [ "$number" -eq 0 ]; then
      echo "Exiting..."
      break
   fi

   if [ $((number % 2)) -eq 0 ]; then
      echo "$number is even."
   else
      echo "$number is odd."
   fi
done
```



Que 3. Create a function that takes a filename as an argument and prints the number of lines in the file. Call this function from your script with different filenames.

Ans:

```
count_lines() {
  wc -l "$1"
}

filenames=("file1.txt" "file2.csv" "data.py")
for filename in "${filenames[@]}"; do

  if [[ -f "$filename" ]]; then
    echo "Number of lines in $filename:"
    count_lines "$filename"
  else
    echo "Error: File '$filename' does not exist."
  fi
done
```

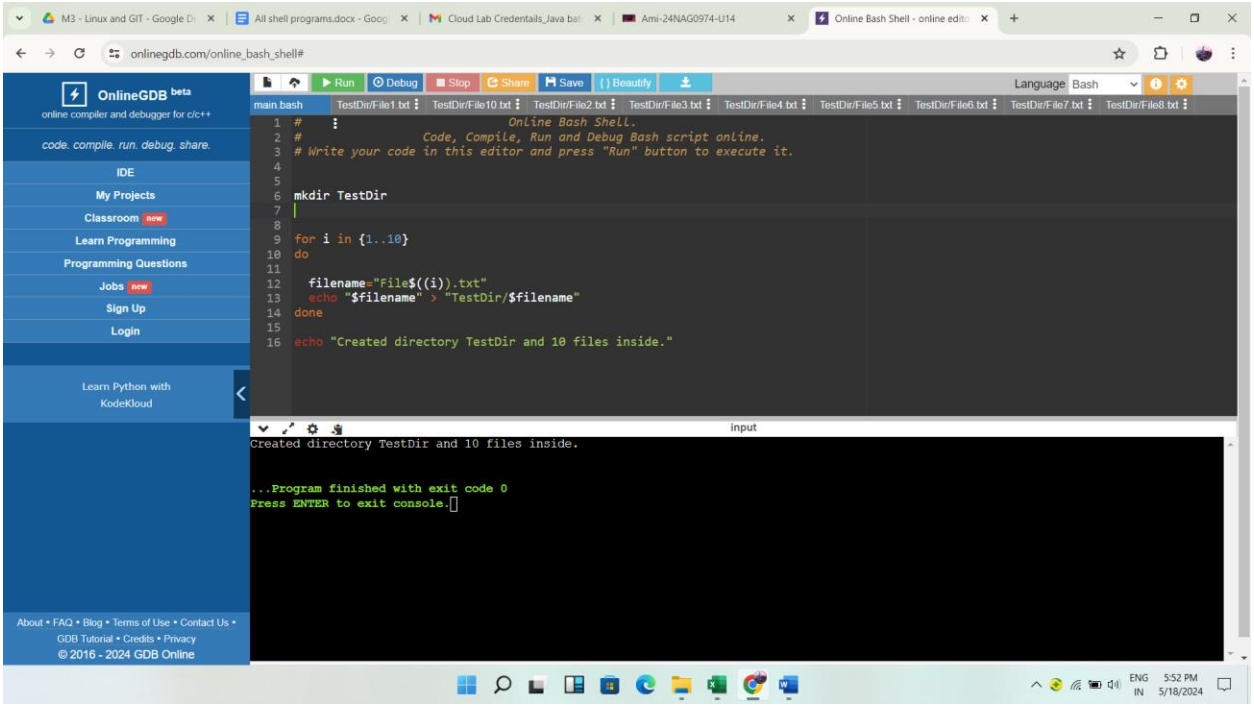


Que 4.

Write a script that creates a directory named TestDir and inside it, creates ten files named File1.txt, File2.txt, ... File10.txt. Each file should contain its filename as its content (e.g., File1.txt contains "File1.txt").

Ans.

```
mkdir TestDir



for i in {1..10}

do


  filename="File$((i)).txt"

  echo "$filename" > "TestDir/$filename"

done


echo "Created directory TestDir and 10 files inside."
```



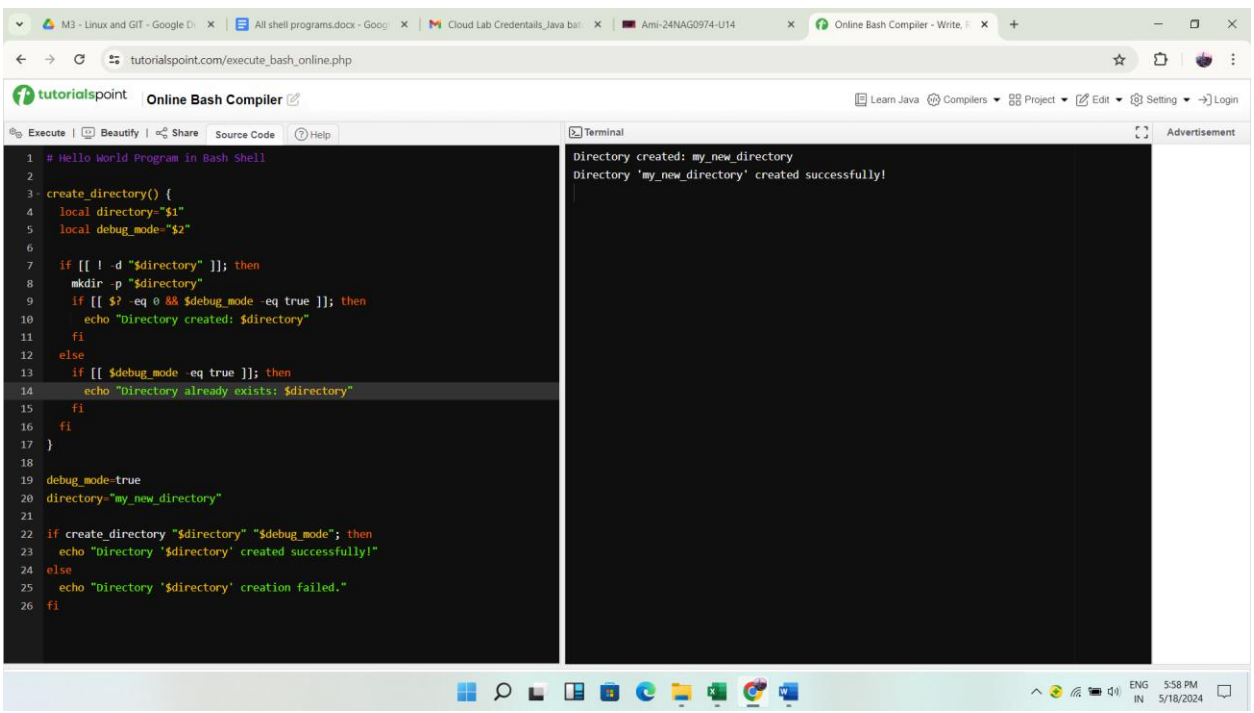
Que 5. Modify the script to handle errors, such as the directory already existing or lacking permissions to create files.

Add a debugging mode that prints additional information when enabled.

Ans.

```
create_directory() {
  local directory="$1"
  local debug_mode="$2"

  if [[ ! -d "$directory" ]]; then
    mkdir -p "$directory"
    if [[ $? -eq 0 && $debug_mode -eq true ]]; then
      echo "Directory created: $directory"
    fi
  else
    if [[ $debug_mode -eq true ]]; then
      echo "Directory already exists: $directory"
```

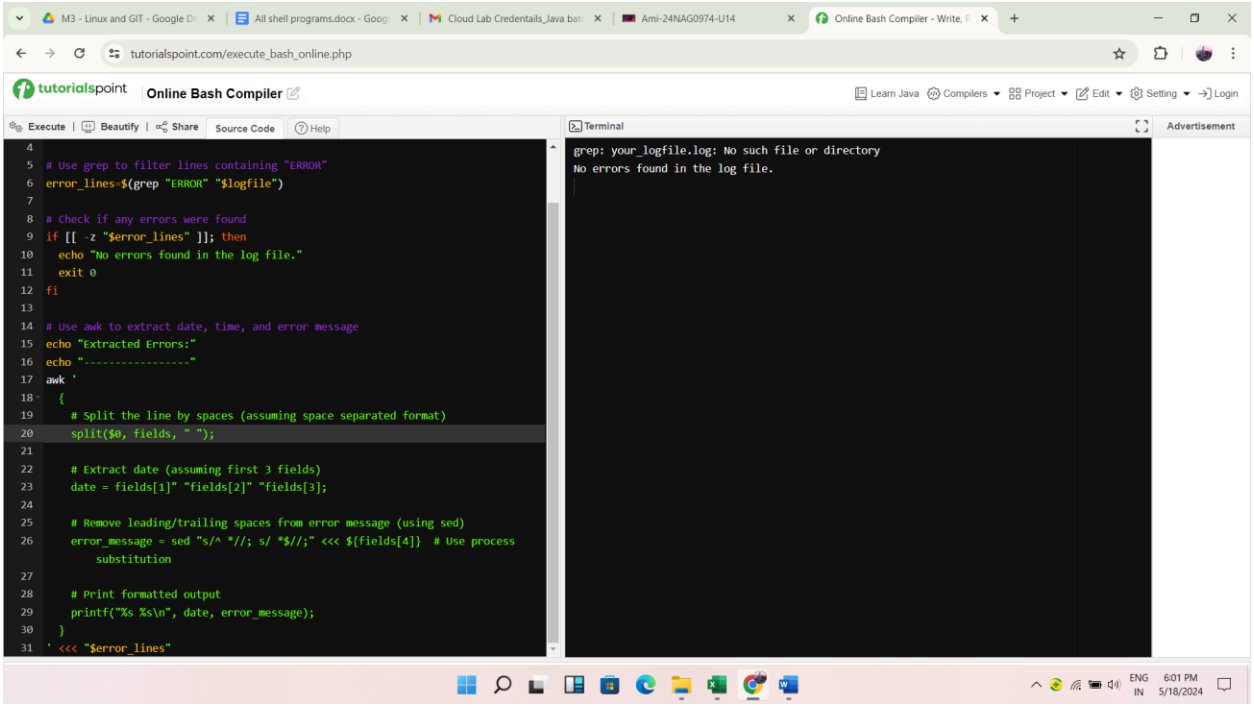
```
    fi
  fi
}

debug_mode=true
directory="my_new_directory"

if create_directory "$directory" "$debug_mode"; then
  echo "Directory '$directory' created successfully!"
else
  echo "Directory '$directory' creation failed."
fi
```



Que. 6 Given a sample log file, write a script using grep to extract all lines containing "ERROR". Use awk to print the date, time, and error message of each extracted line.

Data Processing with sed.

```
error_lines=$(grep "ERROR" "$logfile")

# Check if any errors were found
if [[ -z "$error_lines" ]]; then
  echo "No errors found in the log file."
  exit 0
fi

# Use awk to extract date, time, and error message
echo "Extracted Errors:"
echo "-----------------"
awk '
  {
    # Split the line by spaces (assuming space separated format)
    split($0, fields, " ");

    # Extract date (assuming first 3 fields)
    date = fields[1]" "fields[2]" "fields[3];

    # Remove leading/trailing spaces from error message (using sed)
    error_message = sed "s/^ *//; s/ *$//;" <<< ${fields[4]}  # Use process substitution

    # Print formatted output
    printf("%s %s\n", date, error_message);
  }
' <<< "$error_lines"
```



Que 7. Create a script that takes a text file and replaces all occurrences of "old_text" with "new_text". Use sed to perform this operation and output the result to a new file.

Ans.

```
if [ $# -ne 2 ]; then
  echo "Usage: $0 <input_file> <new_text>"
  exit 1
fi

input_file="$1"
new_text="$2"

if [ ! -f "$input_file" ]; then
  echo "Error: File '$input_file' does not exist."
  exit 1
fi
```

```
output_file="${input_file%.txt}.replaced.txt"

sed -i "s/old_text/$new_text/g" "$input_file"

if [ $? -eq 0 ]; then
  echo "Successfully replaced 'old_text' with '$new_text' in '$input_file'."
  echo "Output saved to: $output_file"
else
  echo "Error: Failed to replace text in '$input_file'."
fi
```