Task No: 5 Date:	Write a Spark application to perform word count in the input file. **Tools: APACHE SPARK**	CO2

Task 4.1: Apache Spark Installation
Aim:
To download, install and configure the Apache Spark in Windows operating system.

Procedure:

1. Download and install Java Development Kit (JDK) version 8 or higher, and ensure the RAM size, least 8 GB.
2. Download and install Python latest version from https://www.python.org/
3. Visit the Apache Spark website at https://spark.apache.org/downloads.html to download the latest stable release of Spark. [figure 1]
4. Type the following command in the command prompt to check the java and python version:
    - java --version
    - python --version
5. Create a new folder named Spark in the root of your C: drive and locate the Spark file you downloaded.
6. Right-click the file and extract it to C:\ApacheSpark using the tool you have on your system
7. Configure Environment Variables

    JAVA_HOME = C:\Program Files\Java\jdk1.8.0_201
    SPARK_HOME = C:\apps\opt\spark-3.5.0-bin-hadoop3
    HADOOP_HOME = C:\apps\opt\spark-3.5.0-bin-hadoop3

    PATH=%PATH%;%SPARK_HOME%\bin;%JAVA_HOME%\bin

8. Launch Spark, To start Spark, enter the command
    - C:\Spark\spark-2.4.5-bin-hadoop2.7\bin>spark-shell
    - If you set the environment path correctly, you can type spark-shell to launch Spark.
    - Finally, the Spark logo appears, and the prompt displays the Scala shell.
    - Open a web browser and navigate to http://localhost:4040/.

9. Download winutils.exe for Hadoop 3.3 using the link https://github.com/kontext-tech/winutils/tree/master/hadoop-3.3.0/bin and copy it to %SPARK_HOME%\bin folder. Winutils differ for each Hadoop version

10. Open command prompt, and go to bin directory of spark home, then type spark-shell command
    C:\Spark\spark-2.4.5-bin-hadoop2.7\bin>spark-shell

Scala Program:

Execute in interpreter mode:

```
scala> val data=sc.textFile("sparkdata.txt");

scala> data.collect;

scala> val splitdata = data.flatMap(line => line.split(""));

scala> splitdata.collect;

scala> val mapdata = splitdata.map(word => (word,1));

scala> mapdata.collect;

scala> val reducedata = mapdata.reduceByKey(_+_);

scala> reducedata.collect;
```

Thus:

Thus Apache Spark downloaded, installed, configured and executed the Spark application to perform word count in the input file, successfully.

Figure 1

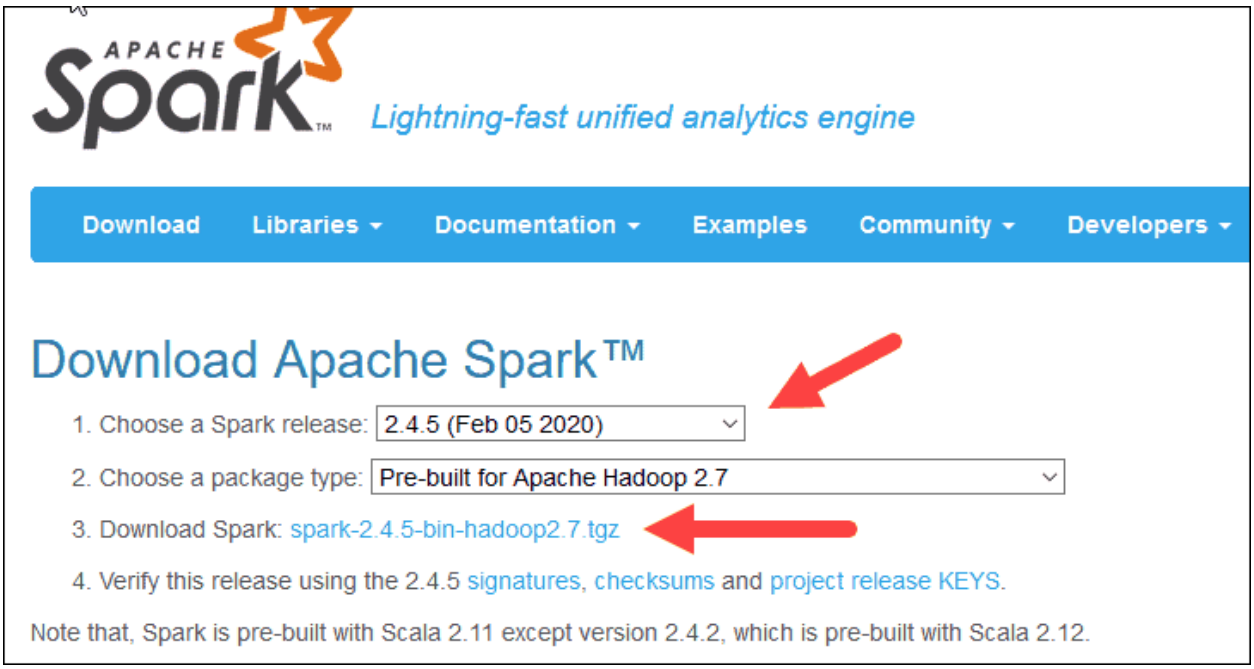


Figure 2

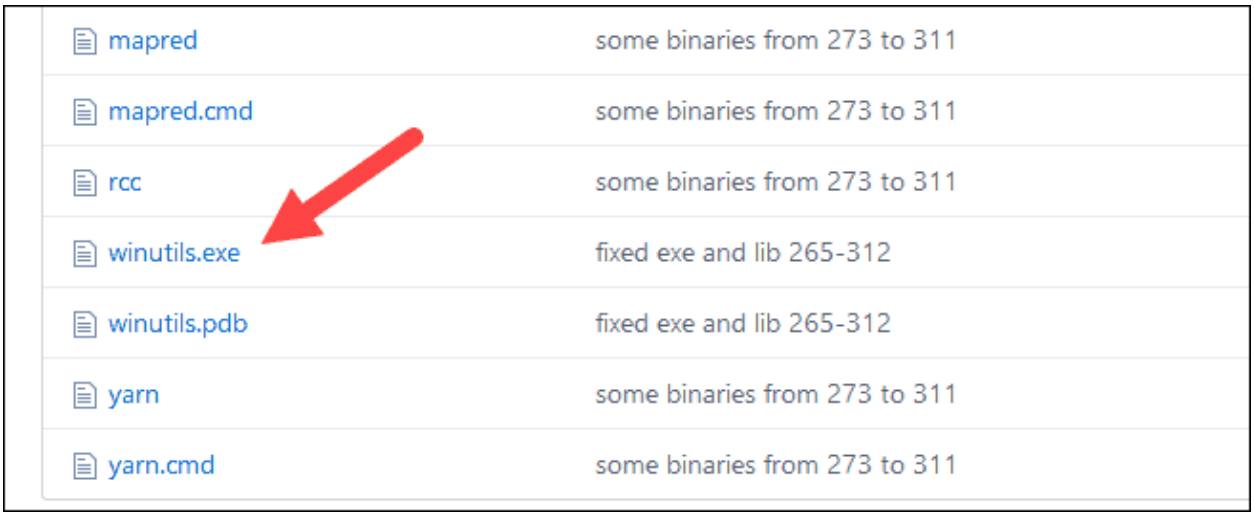


Figure 3

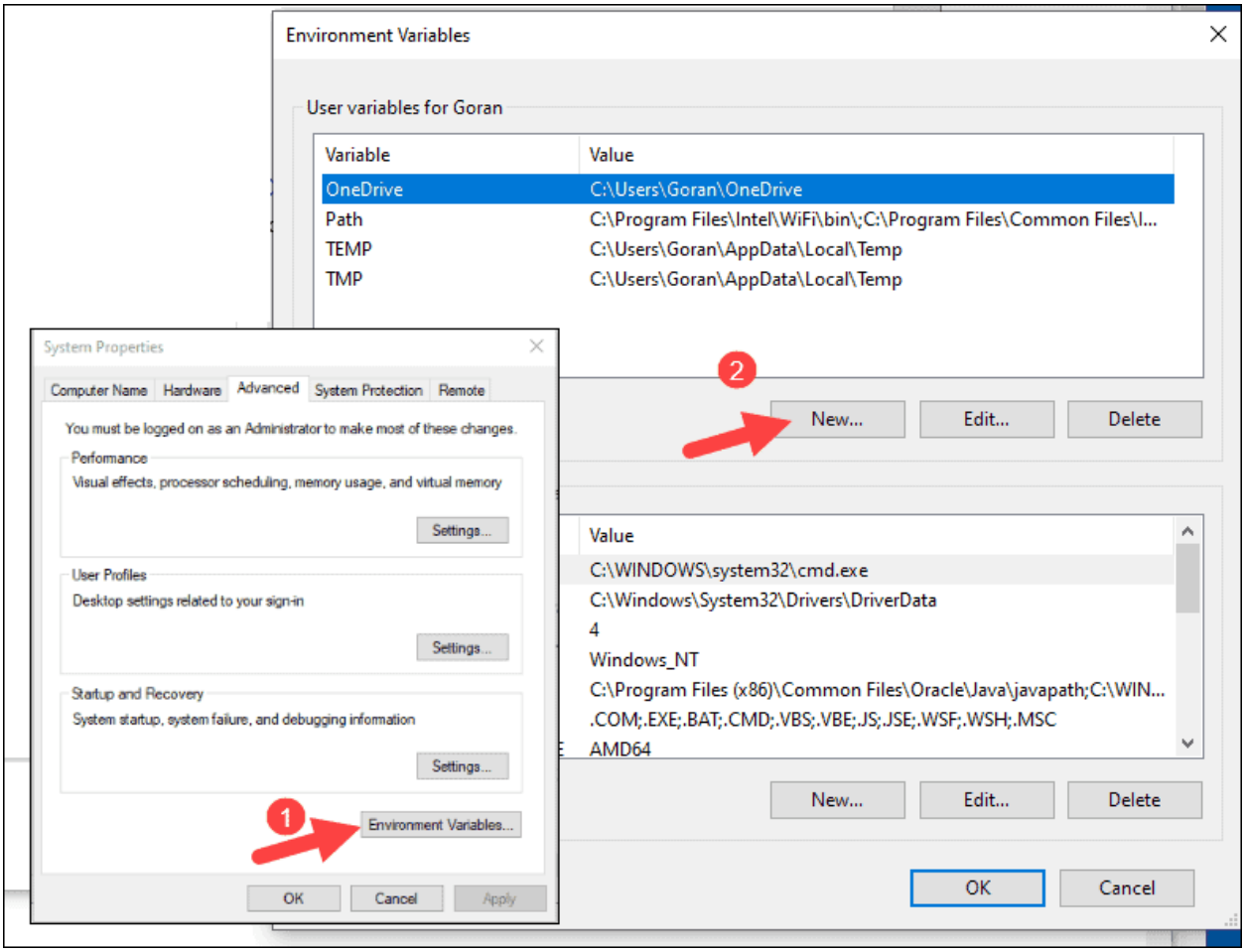


Figure 4

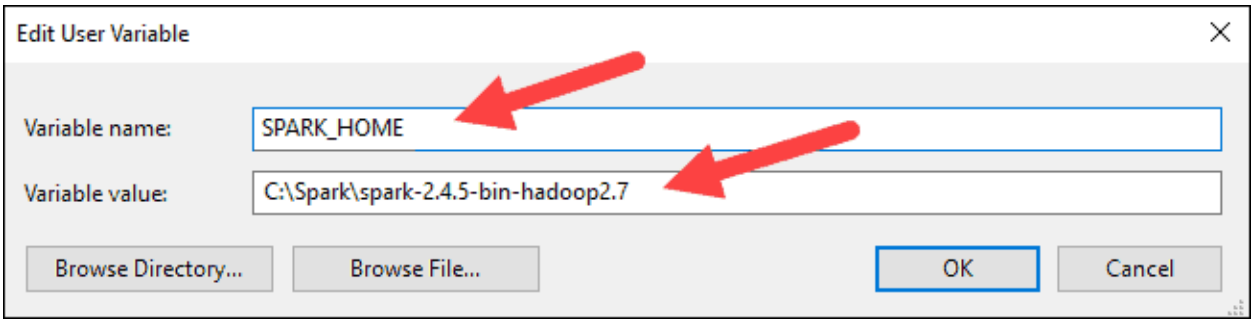


Figure 5

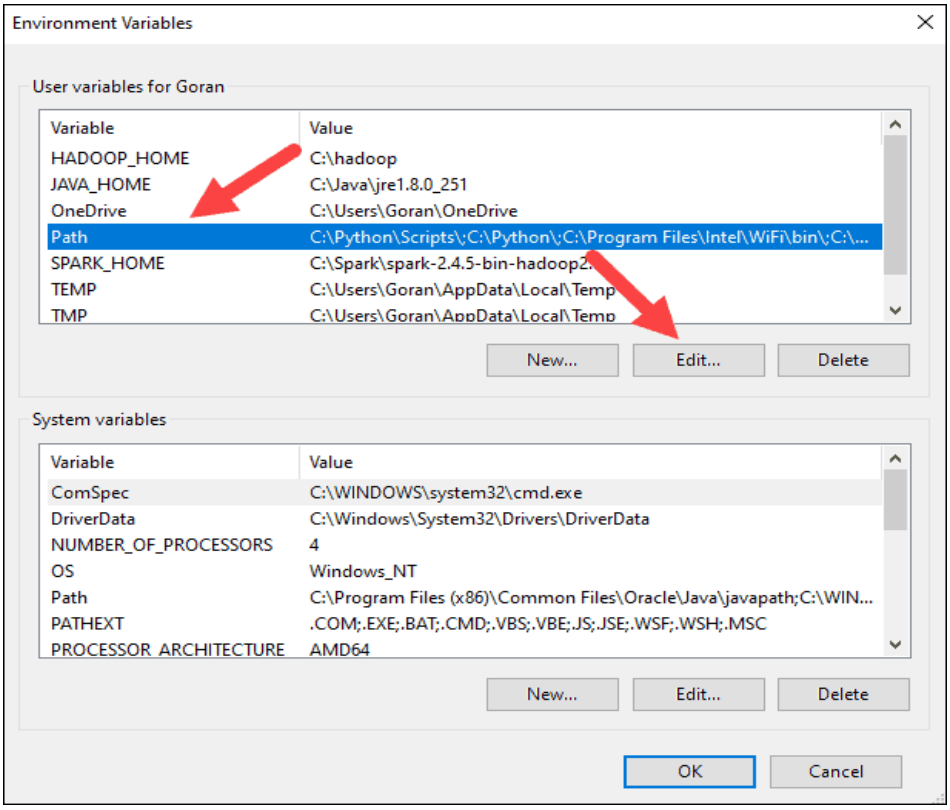
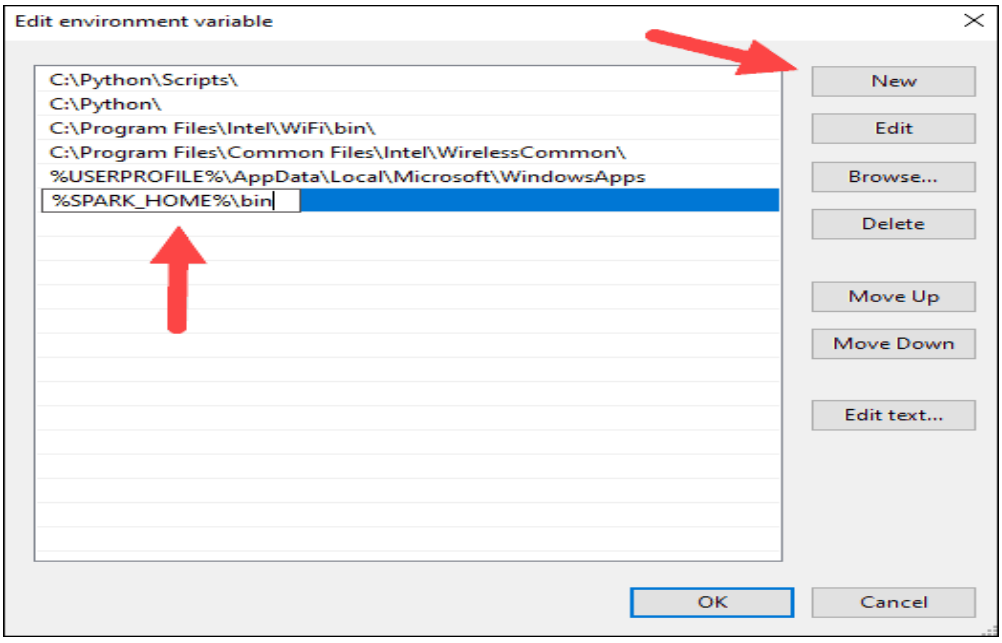


Figure 6



Task 4.2: Spark application to perform word count

Aim:
To implement the spark application to perform the word count using pyspark

Procedure:

1. Check the spark installation, environmental variables setup
2. Import necessary libraries from pyspark
3. Create a SparkConf and SparkContext (or SparkSession):
4. Load the input data
5. Read the input file and Calculating words count
6. Perform the word count operation
7. Save the output, Stop the SparkContext or SparkSession
8. Stopping Spark-Session and Spark context

Program:

```
import findspark

findspark.init()

from pyspark.sql import SparkSession

spark = SparkSession.builder\
```

```
                .master("local")\
                .appName('Firstprogram')\
                .getOrCreate()
sc=spark.sparkContext
text_file = sc.textFile("firstprogram.txt")
counts = text_file.flatMap(lambda line: line.split(" ")) \
                    .map(lambda word: (word, 1)) \
                   .reduceByKey(lambda x, y: x + y)
output = counts.collect()
for (word, count) in output:
   print("%s: %i" % (word, count))
sc.stop()
spark.stop()
```

File Name: firstprogram.txt

Chennai formerly known as Madras, is the capital city of Tamil Nadu, the southernmost Indian state.

Output:

Chennai: 1

Formerly: 1

Known: 1

as: 1

Madras: 1

Is: 1

the: 2

Capital: 1

City: 1

Of: 1

Tamil: 1

Nadu: 1

southernmost: 1

Indian: 1

state.: 1

Result:

Thus the Program to count the words in the given file using Pyspark application was implemented successfully.