7. To perform data reduction operation using Weka

Name of Student		**Roll No.**	
Sign here to indicate that you have read all relevant material provided /available on Moodle while performing and writing this experiment		**Sign:**	

Late Submission Details (if any)

Reason(s) of late submission	**Date of practical performance**	**Date of practical submission**

References used

1	Name and author of reference book(s) with page nos.	
2	Name and roll nos. of the peers whose help you have taken (if any)	

Rubrics for assessment of Experiment:

Indicator	Poor	Average	Good
Timeliness Maintains Experiment deadline (3)	Experiment not done (0)	One or More than One week late (1-2)	Maintains deadline (3)
Completeness and neatness Complete all parts of Experiment (3)	N/A	< 80% complete (1-2)	100% complete (3)
Originality Extent of plagiarism (2)	Copied it from someone else (0)	At least try to implement but could not succeed (1)	Implemented (2)
Knowledge In depth knowledge of the Experiment (2)	Unable to answer any questions (0)	Unable to answer few questions (1)	Able to answer all questions (2)

Assessment Marks:

Timeliness	
Completeness and neatness	
Originality	
Knowledge	
Total	

Signature of Teacher with date:

Course, Subject & Experiment Details

Course & Branch	T.E. (ECS)	Estimated Time	02 Hours Per Week
Current Semester	Semester V	Subject Name	DWM
Chapter No. & Unit	3	Chapter Title	Data pre-processing
Experiment Type	Software Performance	Subject Code	ECC 604

Aim & Objective of Experiment

1. Understand data reduction operations

Expected Outcome of Experiment

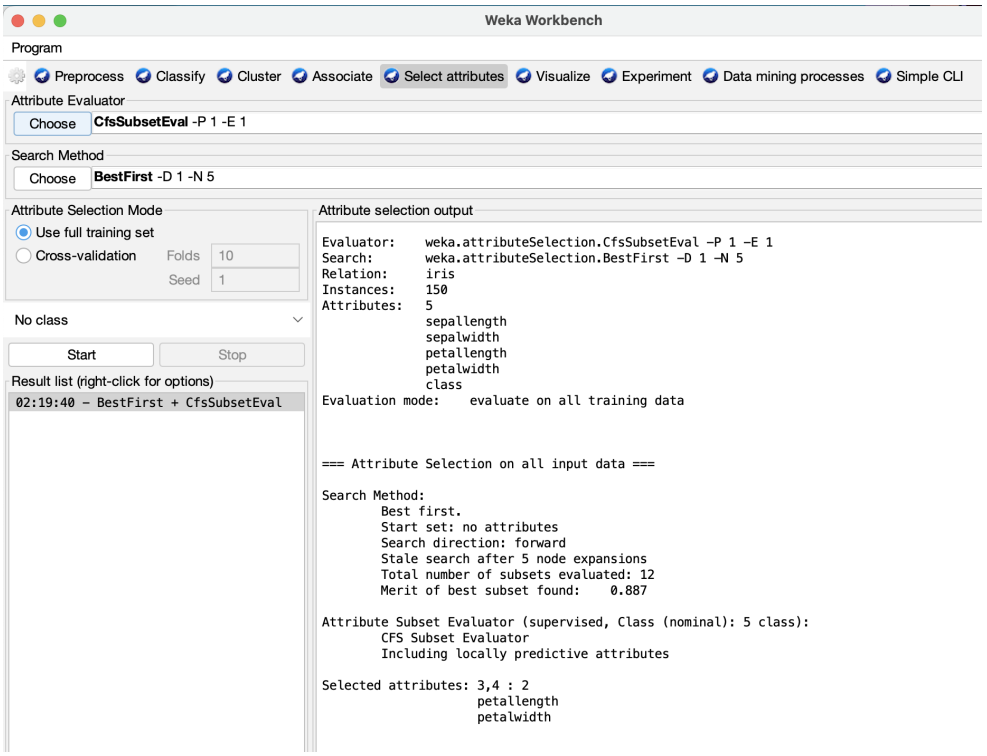
1. Demonstrate data reduction operation using Weka

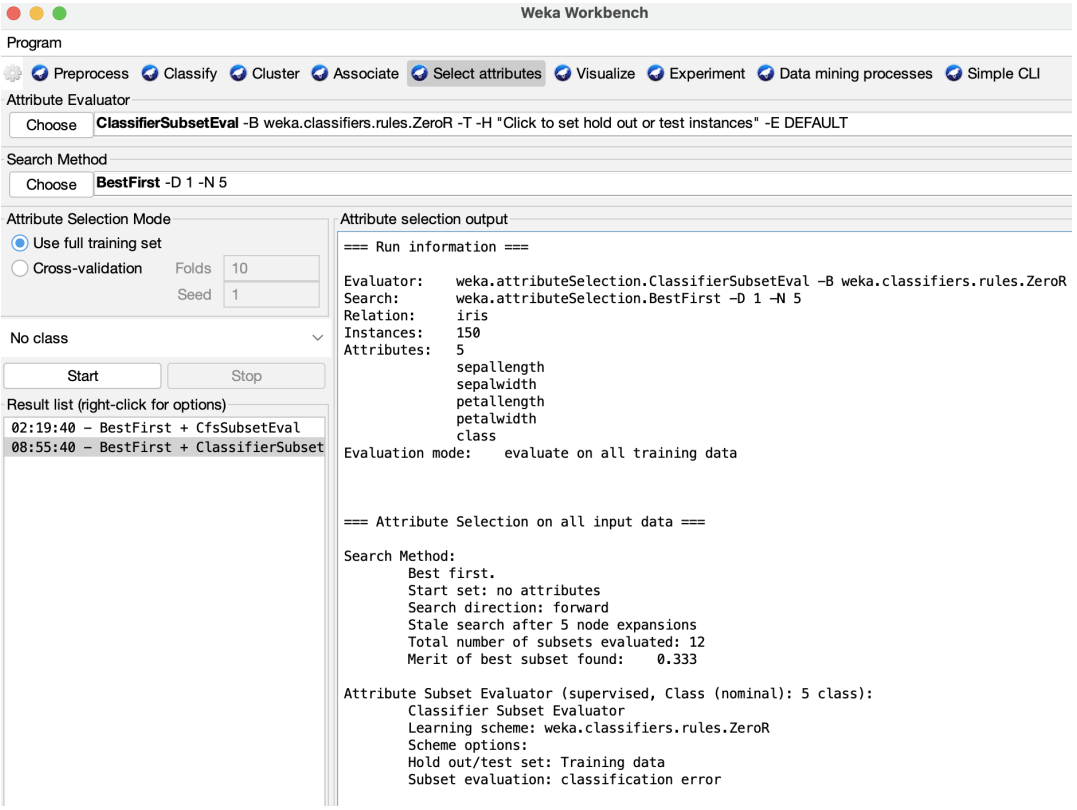
Theory and procedure of the experiment

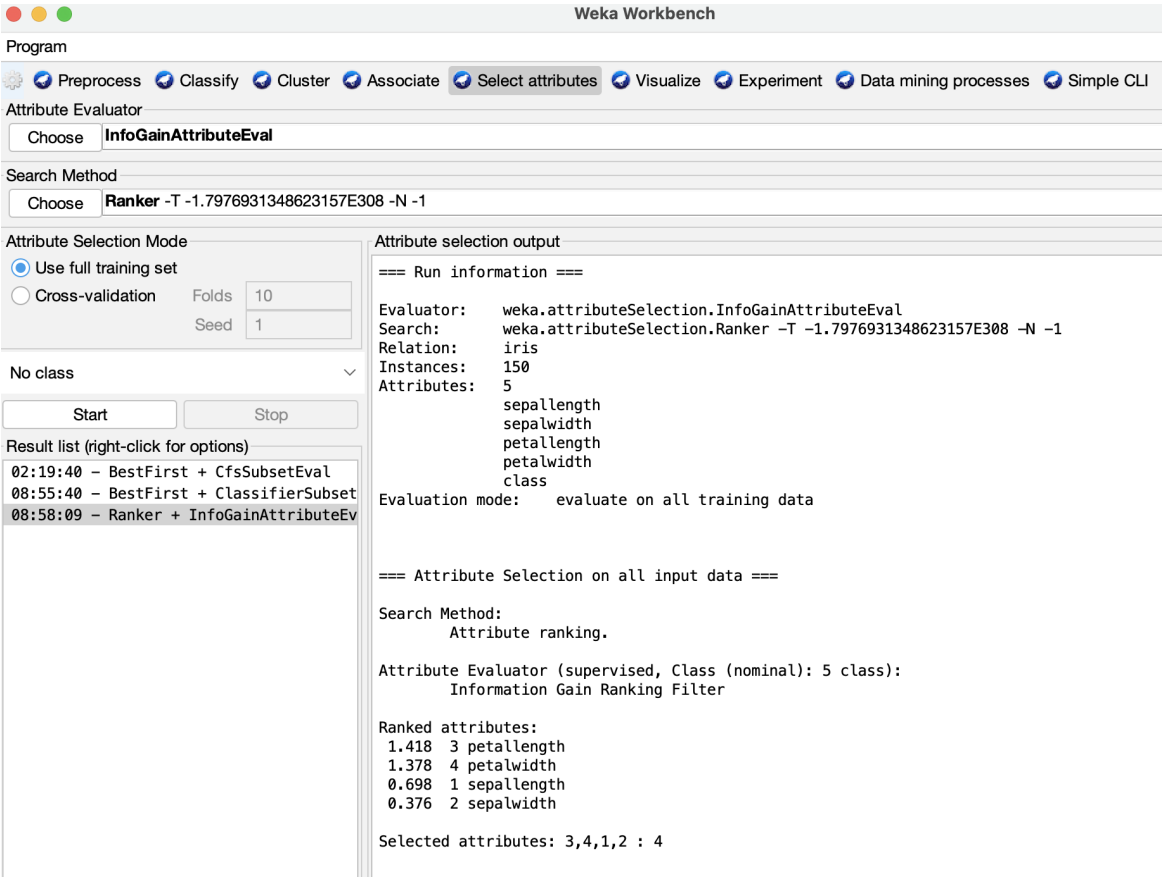
Solve Exercise 5, 6 and 7 from the following document:
https://drive.google.com/file/d/12uy7hEpYEUy5UdIJkRChWR0LUNmDYIjp/view?usp=share_link

Output

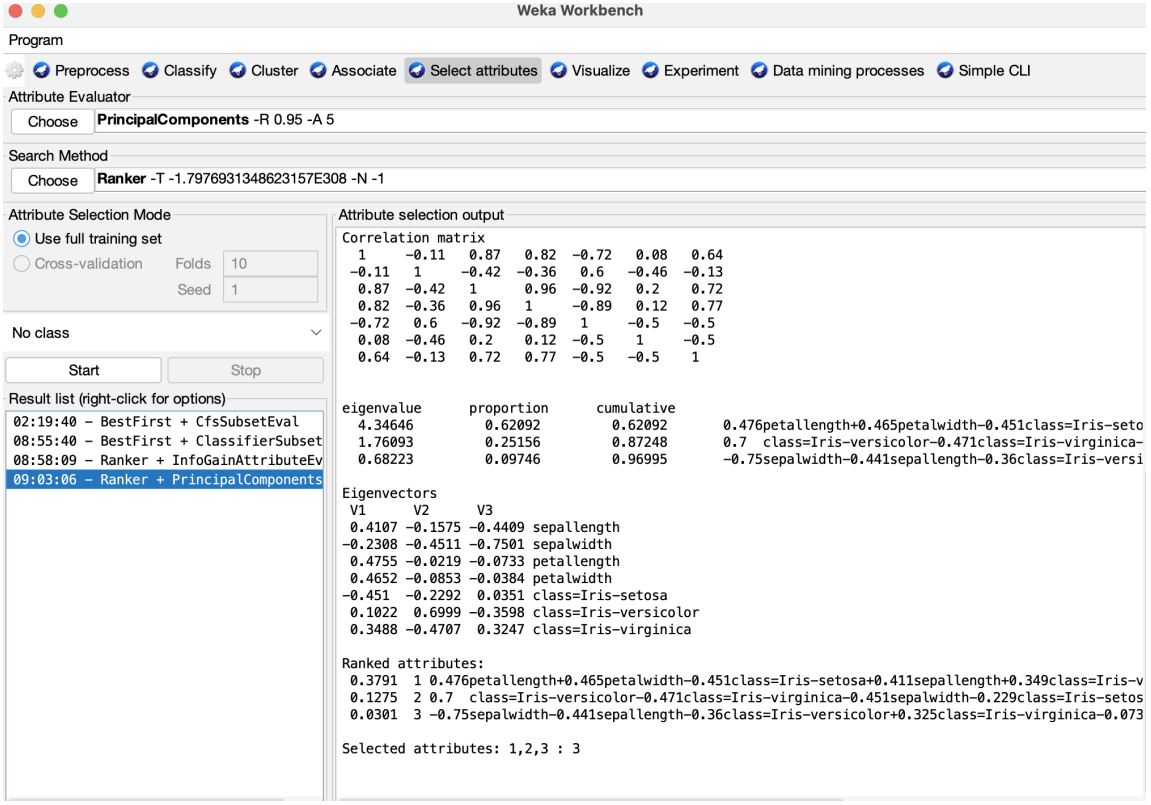
Attach the screenshots of output above exercises and comment on the result of each exercise 5

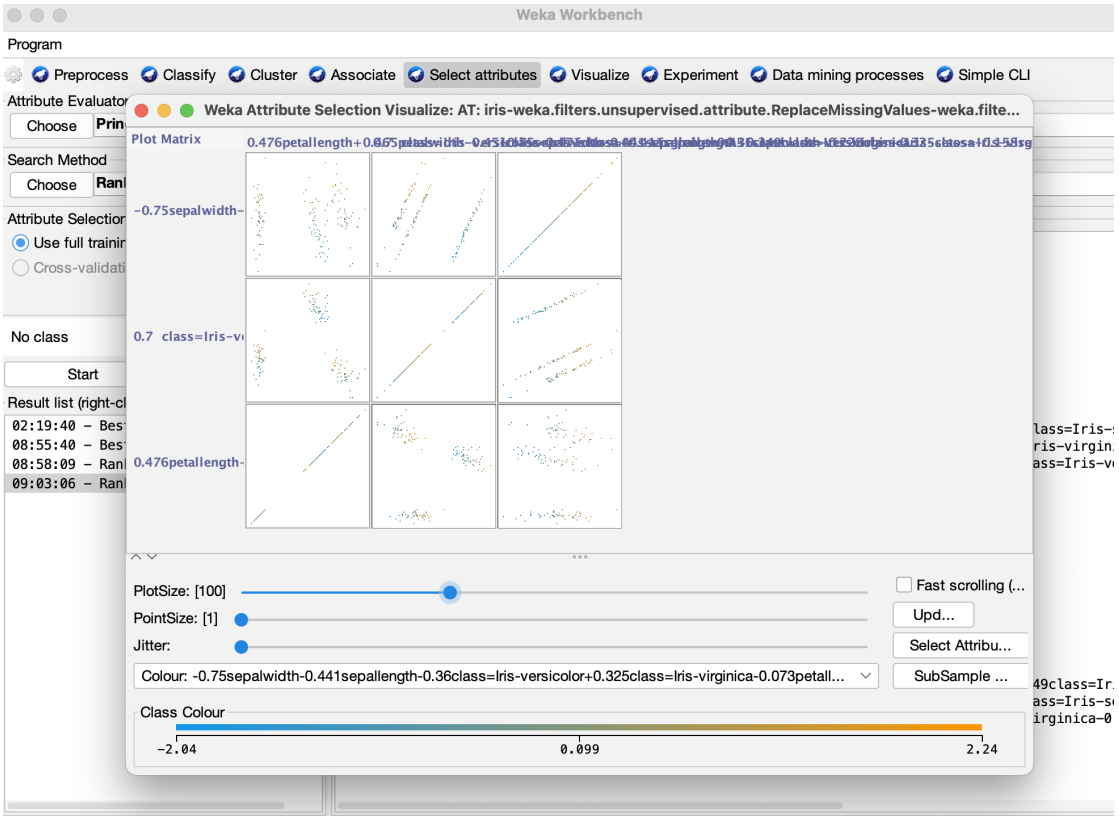



Weka Workbench
Program
Preprocess Classify Cluster Associate Select attributes Visualize Experiment Data mining processes Simple CLI
Attribute Evaluator
Choose ClassifierSubsetEval -B weka.classifiers.rules.ZeroR -T -H "Click to set hold out or test instances" -E DEFAULT
Search Method
Choose BestFirst -D 1 -N 5
Attribute Selection Mode
Use full training set
Cross-validation Folds 10
Seed 1
No class
Start Stop
Result list (right-click for options)
02:19:40 - BestFirst + CfsSubsetEval
08:55:40 - BestFirst + ClassifierSubset
Attribute selection output
=== Run information ===
Evaluator: weka.attributeSelection.ClassifierSubsetEval -B weka.classifiers.rules.ZeroR
Search: weka.attributeSelection.BestFirst -D 1 -N 5
Relation: iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth
class
Evaluation mode: evaluate on all training data
=== Attribute Selection on all input data ===
Search Method:
Best first.
Start set: no attributes
Search direction: forward
Stale search after 5 node expansions
Total number of subsets evaluated: 12
Merit of best subset found: 0.333
Attribute Subset Evaluator (supervised, Class (nominal): 5 class):
Classifier Subset Evaluator
Learning scheme: weka.classifiers.rules.ZeroR
Scheme options:
Hold out/test set: Training data
Subset evaluation: classification error

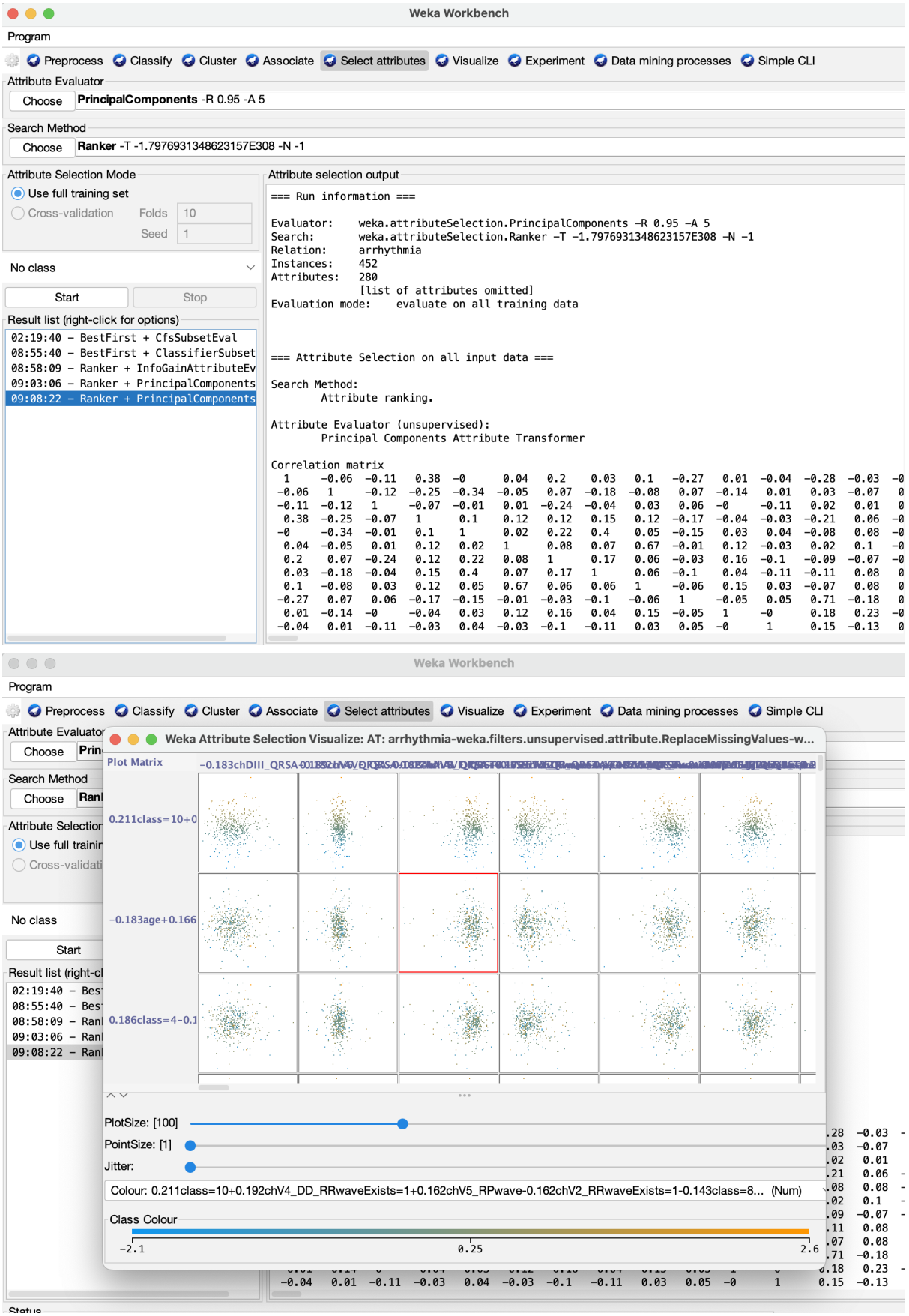

Weka Workbench
Program
Preprocess Classify Cluster Associate Select attributes Visualize Experiment Data mining processes Simple CLI
Attribute Evaluator
Choose InfoGainAttributeEval
Search Method
Choose Ranker -T -1.7976931348623157E308 -N -1
Attribute Selection Mode
Use full training set
Cross-validation Folds 10
Seed 1
No class
Start Stop
Result list (right-click for options)
02:19:40 - BestFirst + CfsSubsetEval
08:55:40 - BestFirst + ClassifierSubset
08:58:09 - Ranker + InfoGainAttributeEv
Attribute selection output
=== Run information ===
Evaluator: weka.attributeSelection.InfoGainAttributeEval
Search: weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1
Relation: iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth
class
Evaluation mode: evaluate on all training data
=== Attribute Selection on all input data ===
Search Method:
Attribute ranking.
Attribute Evaluator (supervised, Class (nominal): 5 class):
Information Gain Ranking Filter
Ranked attributes:
1.418 3 petallength
1.378 4 petalwidth
0.698 1 sepallength
0.376 2 sepalwidth
Selected attributes: 3,4,1,2 : 4

6


Weka Workbench
Program
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Experiment
Data mining processes
Simple CLI
Attribute Evaluator
Choose
PrincipalComponents -R 0.95 -A 5
Search Method
Choose
Ranker -T -1.7976931348623157E308 -N -1
Attribute Selection Mode
Use full training set
Cross-validation
Folds
10
Seed
1
No class
Start
Stop
Result list (right-click for options)
02:19:40 - BestFirst + CfsSubsetEval
08:55:40 - BestFirst + ClassifierSubset
08:58:09 - Ranker + InfoGainAttributeEv
09:03:06 - Ranker + PrincipalComponents
Attribute selection output
Correlation matrix
1 -0.11 0.87 0.82 -0.72 0.08 0.64
-0.11 1 -0.42 -0.36 0.6 -0.46 -0.13
0.87 -0.42 1 0.96 -0.92 0.2 0.72
0.82 -0.36 0.96 1 -0.89 0.12 0.77
-0.72 0.6 -0.92 -0.89 1 -0.5 -0.5
0.08 -0.46 0.2 0.12 -0.5 1 -0.5
0.64 -0.13 0.72 0.77 -0.5 -0.5 1
eigenvalue proportion cumulative
4.34646 0.62092 0.62092 0.476petallength+0.465petalwidth-0.451class=Iris-seto
1.76093 0.25156 0.87248 0.7 class=Iris-versicolor-0.471class=Iris-virginica-
0.68223 0.09746 0.96995 -0.75sepalwidth-0.441sepallength-0.36class=Iris-versi
Eigenvectors
V1 V2 V3
0.4107 -0.1575 -0.4409 sepallength
-0.2308 -0.4511 -0.7501 sepalwidth
0.4755 -0.0219 -0.0733 petallength
0.4652 -0.0853 -0.0384 petalwidth
-0.451 -0.2292 0.0351 class=Iris-setosa
0.1022 0.6999 -0.3598 class=Iris-versicolor
0.3488 -0.4707 0.3247 class=Iris-virginica
Ranked attributes:
0.3791 1 0.476petallength+0.465petalwidth-0.451class=Iris-setosa+0.411sepallength+0.349class=Iris-v
0.1275 2 0.7 class=Iris-versicolor-0.471class=Iris-virginica-0.451sepalwidth-0.229class=Iris-setos
0.0301 3 -0.75sepalwidth-0.441sepallength-0.36class=Iris-versicolor+0.325class=Iris-virginica-0.073
Selected attributes: 1,2,3 : 3


Weka Workbench
Program
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Experiment
Data mining processes
Simple CLI
Weka Attribute Selection Visualize: AT: iris-weka.filters.unsupervised.attribute.ReplaceMissingValues-weka.filte...
Plot Matrix
-0.75sepalwidth-
0.7 class=Iris-v
0.476petallength-
PlotSize: [100]
PointSize: [1]
Jitter:
Fast scrolling (...
Upd...
Select Attribu...
SubSample ...
Colour: -0.75sepalwidth-0.441sepallength-0.36class=Iris-versicolor+0.325class=Iris-virginica-0.073petall...
Class Colour
-2.04
0.099
2.24

7


Weka Workbench
Program
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Experiment
Data mining processes
Simple CLI
Attribute Evaluator
Choose
PrincipalComponents -R 0.95 -A 5
Search Method
Choose
Ranker -T -1.7976931348623157E308 -N -1
Attribute Selection Mode
Use full training set
Cross-validation
Folds
10
Seed
1
No class
Start
Stop
Result list (right-click for options)
02:19:40 - BestFirst + CfsSubsetEval
08:55:40 - BestFirst + ClassifierSubset
08:58:09 - Ranker + InfoGainAttributeEv
09:03:06 - Ranker + PrincipalComponents
09:08:22 - Ranker + PrincipalComponents
Attribute selection output
=== Run information ===
Evaluator: weka.attributeSelection.PrincipalComponents -R 0.95 -A 5
Search: weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1
Relation: arrhythmia
Instances: 452
Attributes: 280
[list of attributes omitted]
Evaluation mode: evaluate on all training data
=== Attribute Selection on all input data ===
Search Method:
Attribute ranking.
Attribute Evaluator (unsupervised):
Principal Components Attribute Transformer
Correlation matrix
1 -0.06 -0.11 0.38 -0 0.04 0.2 0.03 0.1 -0.27 0.01 -0.04 -0.28 -0.03 -0
-0.06 1 -0.12 -0.25 -0.34 -0.05 0.07 -0.18 -0.08 0.07 -0.14 0.01 0.03 -0.07 0
-0.11 -0.12 1 -0.07 -0.01 0.01 -0.24 -0.04 0.03 0.06 -0 -0.11 0.02 0.01 0
0.38 -0.25 -0.07 1 0.1 0.12 0.12 0.15 0.12 -0.17 -0.04 -0.03 -0.21 0.06 -0
-0 -0.34 -0.01 0.1 1 0.02 0.22 0.4 0.05 -0.15 0.03 0.04 -0.08 0.08 -0
0.04 -0.05 0.01 0.12 0.02 1 0.08 0.07 0.67 -0.01 0.12 -0.03 0.02 0.1 -0
0.2 0.07 -0.24 0.12 0.22 0.08 1 0.17 0.06 -0.03 0.16 -0.1 -0.09 -0.07 -0
0.03 -0.18 -0.04 0.15 0.4 0.07 0.17 1 0.06 -0.1 0.04 -0.11 -0.11 0.08 0
0.1 -0.08 0.03 0.12 0.05 0.67 0.06 0.06 1 -0.06 0.15 0.03 -0.07 0.08 0
-0.27 0.07 0.06 -0.17 -0.15 -0.01 -0.03 -0.1 -0.06 1 -0.05 0.05 0.71 -0.18 0
0.01 -0.14 -0 -0.04 0.03 0.12 0.16 0.04 0.15 -0.05 1 -0 0.18 0.23 -0
-0.04 0.01 -0.11 -0.03 0.04 -0.03 -0.1 -0.11 0.03 0.05 -0 1 0.15 -0.13 0
Weka Workbench
Program
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Experiment
Data mining processes
Simple CLI
Weka Attribute Selection Visualize: AT: arrhythmia-weka.filters.unsupervised.attribute.ReplaceMissingValues-w...
Plot Matrix
0.211class=10+0
-0.183age+0.166
0.186class=4-0.1
PlotSize: [100]
PointSize: [1]
Jitter:
Colour: 0.211class=10+0.192chV4_DD_RRwaveExists=1+0.162chV5_RPwave-0.162chV2_RRwaveExists=1-0.143class=8... (Num)
Class Colour
-2.1
0.25
2.6
Status

Conclusions & Inferences

In this experiment data reduction techniques in Weka are explored.

Post Lab exercise

1. Describe different methods for data reduction

Data reduction is the process of reducing the amount of data in a dataset while retaining important information. This can be important for various reasons, such as to reduce computational load or to remove noise from the data. Here are some methods for data reduction:

1. Sampling: This involves selecting a subset of the data for analysis. Random sampling, stratified sampling, and cluster sampling are some of the common sampling techniques.

2. Feature selection: This involves selecting a subset of the features (or variables) in the data that are most relevant to the analysis. This can be done using techniques such as correlation analysis, principal component analysis (PCA), or mutual information.

3. Feature extraction: This involves transforming the original features into a smaller set of features that capture the most important information. Techniques such as PCA, independent component analysis (ICA), and factor analysis can be used for feature extraction.

4. Data compression: This involves compressing the data using techniques such as wavelet compression or JPEG compression. This can be useful for reducing the storage space required for the data.

5. Clustering: This involves grouping similar data points together into clusters. The number of clusters can be much smaller than the number of data points, resulting in data reduction.

6. Data summarization: This involves summarizing the data using statistics such as mean, median, or mode. This can be useful for reducing the complexity of the data while retaining important information.

7. Data discretization: This involves converting continuous variables into categorical variables. This can be useful for reducing the amount of data required for analysis.

****

5. Basic shell scripting - I

Name of Student	Piyush Ram Kasle	**Roll No.**	9133
Sign here to indicate that you have read all relevant material provided /available on Moodle while performing and writing this experiment		**Sign:**	

Late Submission Details (if any)

Reason(s) of late submission	**Date of practical performance**	**Date of practical submission**

References used

1	Name and author of reference book(s) with page nos.	
2	Name and roll nos. of the peers whose help you have taken (if any)	

Rubrics for assessment of Experiment:

Indicator	Poor	Average	Good
Timeliness Maintains Experiment deadline (3)	Experiment not done (0)	One or More than One week late (1-2)	Maintains deadline (3)
Completeness and neatness Complete all parts of Experiment (3)	N/A	< 80% complete (1-2)	100% complete (3)
Originality Extent of plagiarism (2)	Copied it from someone else (0)	At least try to implement but could not succeed (1)	Implemented (2)
Knowledge In depth knowledge of the Experiment (2)	Unable to answer any questions (0)	Unable to answer few questions (1)	Able to answer all questions (2)

Assessment Marks:

Timeliness	
Completeness and neatness	
Originality	
Knowledge	
Total	

Signature of Teacher with date

Course, Subject & Experiment Details

Course & Branch	T.E. (ECS)	Estimated Time	02 Hours Per Week
Current Semester	Semester VI	Subject Name	Linux Server Administration
Chapter No. & Unit	5.1	Chapter Title	Shell scripting
Experiment Type	Software Performance	Subject Code	ECL 604

Aim & Objective of Experiment

1. To perform shell scripting

Expected Outcome of Experiment

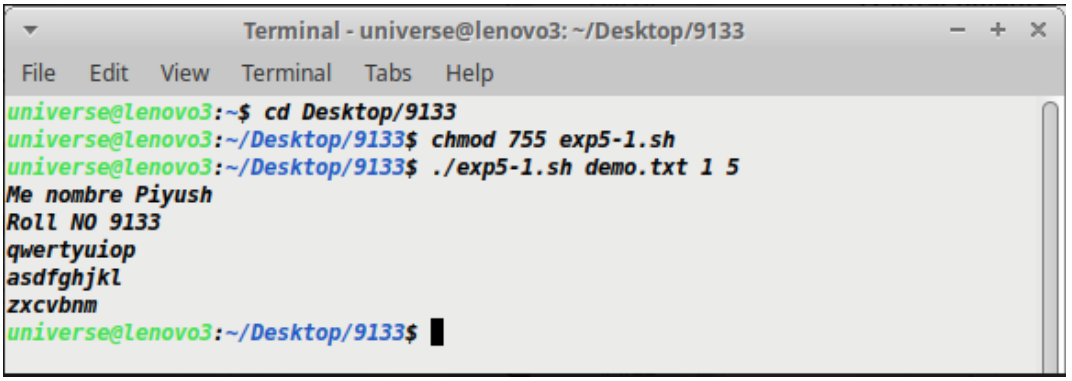
1. To understand shell scripting basics
2. To write codes using shell script

Brief Description of the experiment

1. **Write a Shell Script that accepts a file name, starting and ending line numbers as Arguments and displays all lines between the given line numbers.**
   Code:

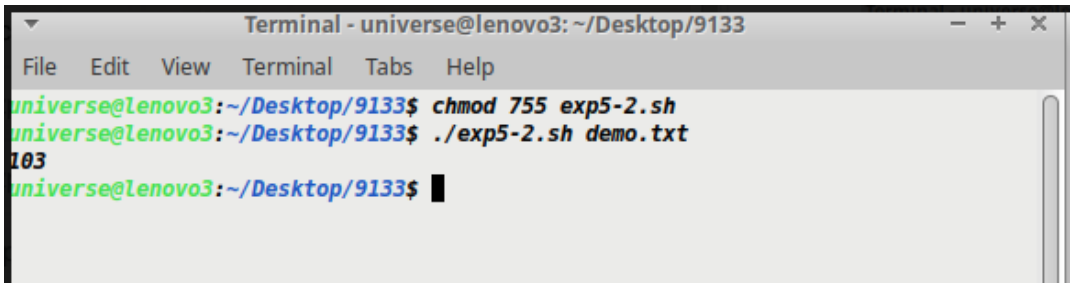
```
#!/bin/bash
if [ $# -lt 3 ]
then
echo "To execute you have to enter
3 arguments in command line in
following order..."
echo " File Name ,starting line
number and ending line number..."
else
sed -n $2,$3p $1
fi
```

Output:



2. **Write a shell script to count no of character in a file , prompt for input-file**
   Code:
   #!/bin/bash
   read file
   echo wc –m $file

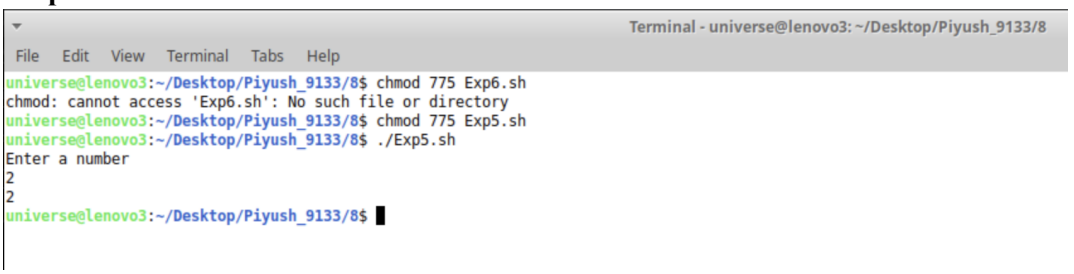
Output:



3. **Write a shell script to find factorial of a number**
   Code:

```
echo "Enter a number"
read num

fact=1

for((i=2;i<=num;i++))
{
  fact=$((fact * i))  #fact = fact * i
}

echo $fact
```

Output:



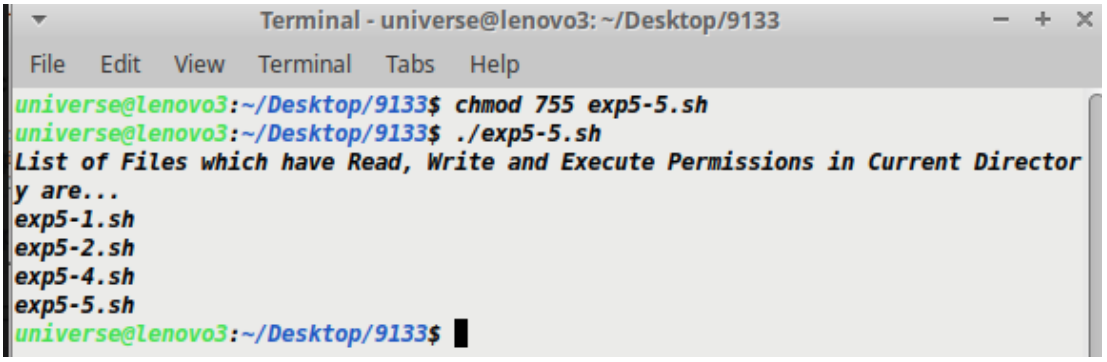
4. **Write a shell script that delete all lines of a file containing a specified word (Hint: use SED command)**
   Code:

Output:

5. **Write a shell script that displays a list of all files in the current directory to which the user has read, write and execute permissions.**
   Code:

```
#!/bin/bash
echo "List of Files which have
Read, Write and Execute
Permissions in Current Directory
are..."
for file in *
do
if [ -r $file -a -w $file -a -x $file ]
then
echo $file
fi
done
```

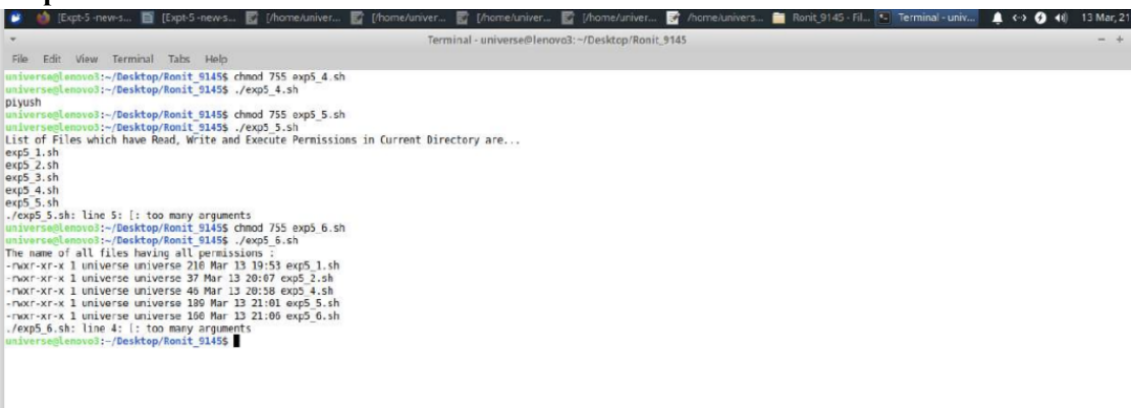
Output:



6. **Write a shell script to count no of file in current directory with full permissions**
   Code:

```
#!/bin/bash
echo "List of files which have all permissions:"
for file in *
do
if [-r $file -a -w $file -a -x $file]
then
i = 0
num = $((i+1))
echo $num
fi
done
```

Output:



Conclusions & Inferences

Understood and performed basic shell scripting.

Post Lab exercise

1. Discuss the usage of SED command to add a word at the end of every line to a file

6. To perform shell scripting (Part - II)

Name of Student		Roll No.	
Sign here to indicate that you have read all relevant material provided /available on Moodle while performing and writing this experiment		Sign:	

Late Submission Details (if any)

Reason(s) of late submission	Date of practical performance	Date of practical submission

References used

1	Name and author of reference book(s) with page nos.	
2	Name and roll nos. of the peers whose help you have taken (if any)	

Rubrics for assessment of Experiment:

Indicator	Poor	Average	Good
Timeliness Maintains Experiment deadline (3)	Experiment not done (0)	One or More than One week late (1-2)	Maintains deadline (3)
Completeness and neatness Complete all parts of Experiment (3)	N/A	< 80% complete (1-2)	100% complete (3)
Originality Extent of plagiarism (2)	Copied it from someone else (0)	At least try to implement but could not succeed (1)	Implemented (2)
Knowledge In depth knowledge of the Experiment (2)	Unable to answer any questions (0)	Unable to answer few questions (1)	Able to answer all questions (2)

Assessment Marks:

Timeliness	
Completeness and neatness	
Originality	
Knowledge	
Total	

Signature of Teacher with date

1. Course, Subject & Experiment Details

FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING
Department of Electronics and Computer Science

Course & Branch	T.E. (ECS)	Estimated Time	02 Hours Per Week
Current Semester	Semester VI	Subject Name	Linux Server Administration
Chapter No. & Unit	5.1	Chapter Title	Bash Shell Scripting
Experiment Type	Software Performance	Subject Code	ECL 604

2. Aim & Objective of Experiment

1. To write programs using shell scripting

3. Expected Outcome of Experiment

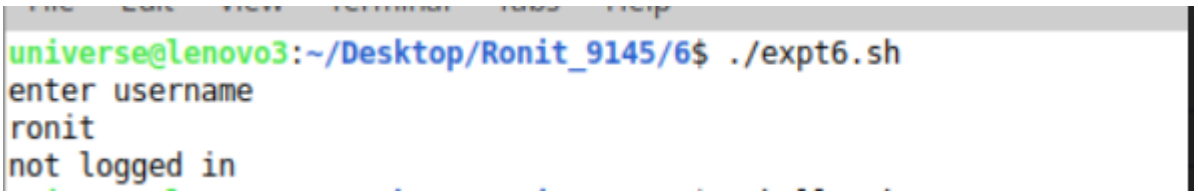
1. To understand shell scripting

4. Brief Description of the experiment

1. Write a shell script program to display list of user currently logged in
   Program:

```
#!/bin/bash

echo enter username
read name
who > test
if grep $name test
then
echo logged in
else
echo not logged in
fi
```

   Screenshot:



2) Write a shell script program to display “HELLO WORLD”.Program:

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

Screenshot:

3 Write a shell script Program to implement calculator Program:

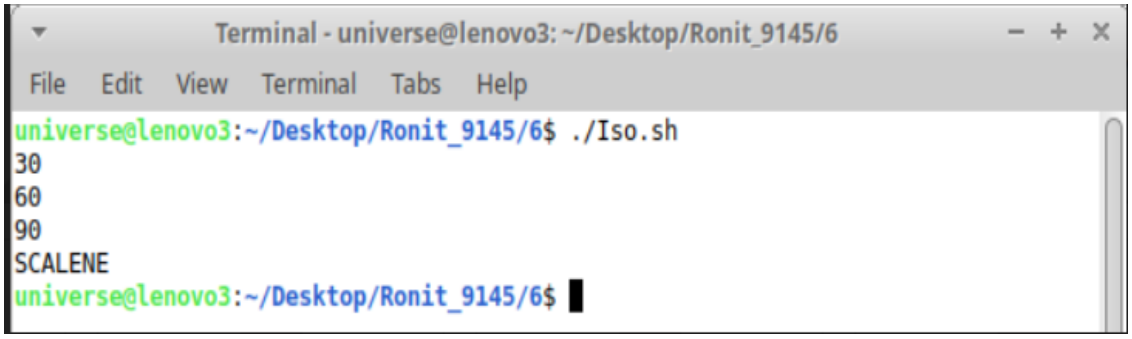
```
# !/bin/bash

# Take user Input
echo "Enter Two numbers : "
read a
read b

# Input type of operation
echo "Enter Choice :"
echo "1. Addition"
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"
read ch

# Switch Case to perform
# calulator operations
case $ch in
  1) res=`echo $a + $b | bc`
  ;;
  2) res=`echo $a - $b | bc`
  ;;
  3) res=`echo $a \* $b | bc`
  ;;
  4) res=`echo "scale=2; $a / $b" | bc`
  ;;
esac
echo "Result : $res"
```

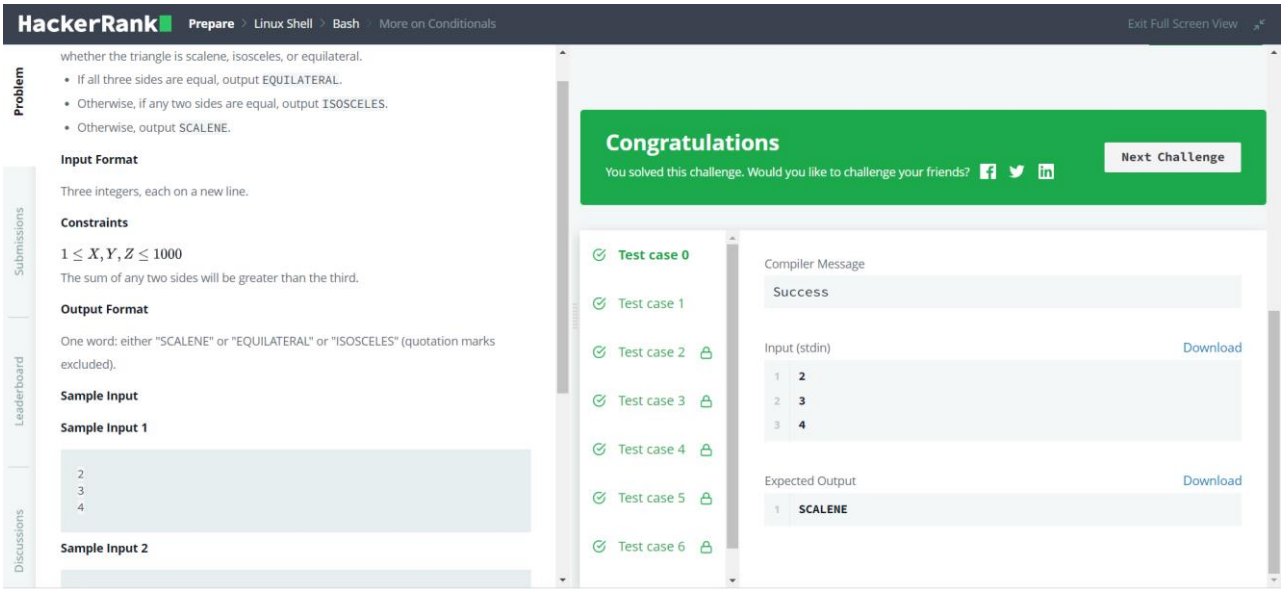
Screenshot:

4 Implement the program from the following link. Submit the solution on hackerrank
https://www.hackerrank.com/challenges/bash-tutorials---more-on-conditionals/problem?isFullScreen=true
Program:

```
read X; read Y; read Z;
if [ $X -eq $Y ] && [ $Y -eq $Z ]; then echo "EQUILATERAL";
elif [ $X -ne $Y ] && [ $X -ne $Z ] && [ $Y -ne $Z ]; then echo "SCALENE";
else echo "ISOSCELES"; fi
```

Screenshot:

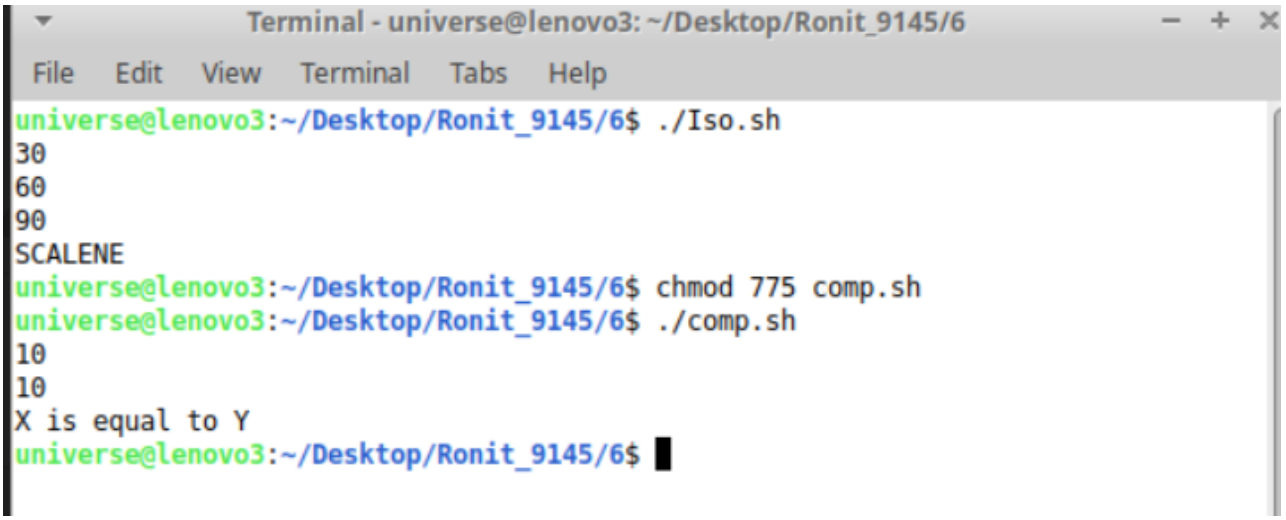


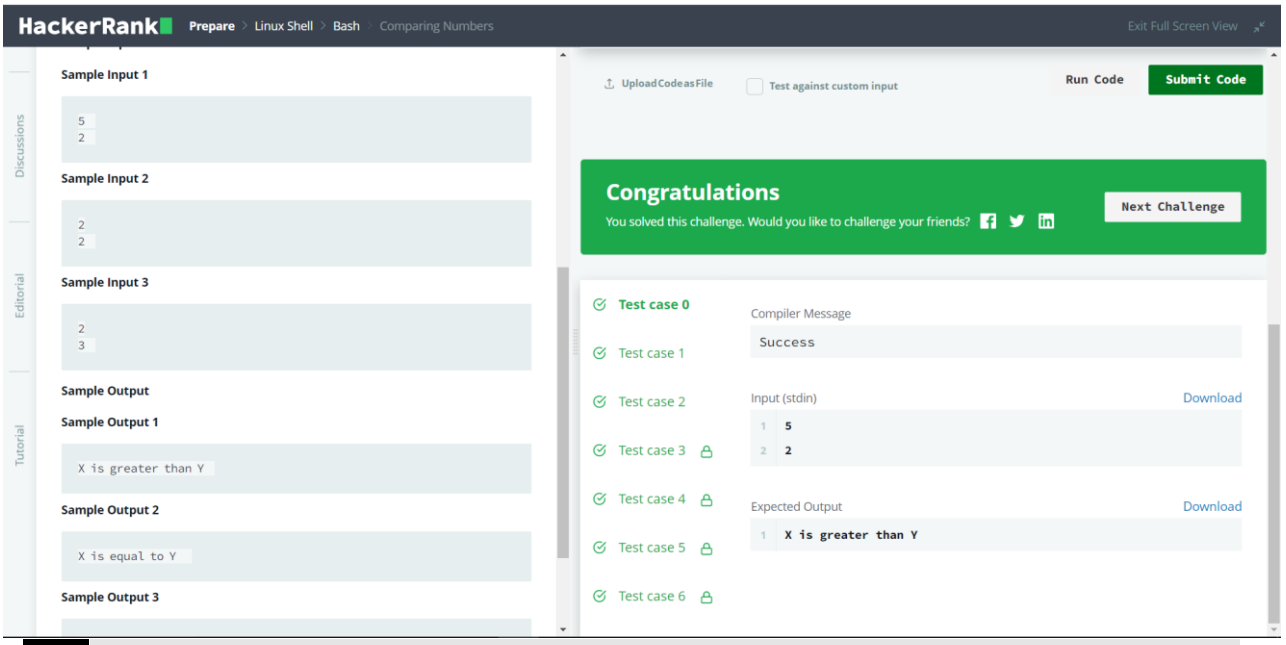


5 Implement the program from the following link. Submit the solution on hackerrank
https://www.hackerrank.com/challenges/bash-tutorials---comparing-numbers/problem?isFullScreen=true
Program:

```
read X
read Y

if (( $X < $Y )); then
echo 'X is less than Y'
elif (( $X > $Y )); then
echo 'X is greater than Y'
else
echo 'X is equal to Y'
fi
```

Screenshot:





5. Conclusions & Inferences

From this experiment, we learned how to write a shell script program and performed 3 programs in terminal and 2 programs in hackerrank.

6. Post Lab exercise

1. Write a note on awk script

Awk is a powerful programming language that is primarily used for text processing and manipulation. It was originally developed at Bell Labs in the 1970s and has since become a popular tool for working with data in Unix-like environments. An awk script is a program written in the awk language that can be executed on the command line or saved as a file.

An awk script typically consists of one or more patterns that specify which lines of input to match, and one or more actions that specify what to do with those lines.

Awk provides a variety of built-in variables and functions that can be used in awk scripts, as well as the ability to define custom variables and functions. For example, the NF variable contains the number of fields in the current line, and the split() function can be used to split a string into an array.

Awk scripts can be very powerful and flexible, and are often used for tasks such as data extraction, data cleaning, and report generation. With its ability to handle large amounts of text data quickly and efficiently, awk is a valuable tool for anyone working with Unix-like systems.

7. To perform advanced shell scripting

Name of Student		**Roll No.**	
Sign here to indicate that you have read all relevant material provided /available on Moodle while performing and writing this experiment		**Sign:**	

Late Submission Details (if any)

Reason(s) of late submission	**Date of practical performance**	**Date of practical submission**

References used

1	Name and author of reference book(s) with page nos.	
2	Name and roll nos. of the peers whose help you have taken (if any)	

Rubrics for assessment of Experiment:

Indicator	Poor	Average	Good
Timeliness Maintains Experiment deadline (3)	Experiment not done (0)	One or More than One week late (1-2)	Maintains deadline (3)
Completeness and neatness Complete all parts of Experiment (3)	N/A	< 80% complete (1-2)	100% complete (3)
Originality Extent of plagiarism (2)	Copied it from someone else (0)	At least try to implement but could not succeed (1)	Implemented (2)
Knowledge In depth knowledge of the Experiment (2)	Unable to answer any questions (0)	Unable to answer few questions (1)	Able to answer all questions (2)

Assessment Marks:

Timeliness	
Completeness and neatness	
Originality	
Knowledge	
Total	

Signature of Teacher with date

1. Course, Subject & Experiment Details

Course & Branch	T.E. (ECS)	Estimated Time	02 Hours Per Week
Current Semester	Semester VI	Subject Name	Linux Server Administration
Chapter No. & Unit	5.1	Chapter Title	Bash Shell Scripting
Experiment Type	Software Performance	Subject Code	ECL 604

2. Aim & Objective of Experiment

1. To write programs using advanced shell scripting

3. Expected Outcome of Experiment

1. To understand advanced shell scripting

4. Brief Description of the experiment

Go to the following URL and write shell script code for the following

URL: https://www.emertxe.com/embedded-systems/linux-systems/ls-assignments/

Exercise No (Batch A): A2, A7, A12, A13, A17, A22, A26, A30
Exercise No (Batch B): A3, A8, A14, A18, A23, A25, A28, A31
Exercise No (Batch C): A5, A10, A11, A15, A20, A24, A27, A35
Exercise No (Batch D): A4, A6, A9, A16, A19, A21, A28, A33

Note: For each program above include following things in the writeup:
Algorithm, shell script code, and screenshot of the output

A3.

Algorithm

1. Take appropriate no. of rows (4) and store it in variable.
2. Take two for loops, one for keeping count of rows and other for displaying numbers.
3. Print the numbers per rows to form the desired pattern.

Code

```
[foss@fedora Ronit_9145]$ cat a3.sh
#!/bin/bash
num=1
row=4
for ((i=1; i<=row; i++))
do
        for ((j=1; j<=i; j++))
        do
                echo -n "$num "
                num=$((num + 1))
        done
echo
done
[foss@fedora Ronit_9145]$
```

Output:

```
[foss@fedora Ronit_9145]$ sh a3.sh
1
2 3
4 5 6
7 8 9 10
[foss@fedora Ronit_9145]$
```

A8.

Algorithm

1. Read the number from the user and store it in variable
2. Extract the last digit from the entered number
3. Add the number to a new variable.
4. Remove the added digit from the entered number.
5. Repeat the steps 2-4 till the number is 0

Output:

```
[foss@fedora Ronit_9145]$ cat a8.sh
echo enter n
read n
num=0
while [ $n -gt 0 ]
do
num=$(expr $num \* 10)
k=$(expr $n % 10)
num=$(expr $num + $k)
n=$(expr $n / 10)
done
echo number is $num
[foss@fedora Ronit_9145]$ sh a8.sh
enter n
2893
number is 3982
[foss@fedora Ronit_9145]$
```

A14.

Algorithm

1. Read the number from the user
2. Prompt the user to enter choice (ascending or descending)
3. Split the number into individual digits using grep command

4. Sort the number in ascending order (sort) or in descending order (sort -r)
5. Print the output

Output:

```
echo "Enter Number"
read n
echo "Enter sort order 1. ASCE 2.DSCE"
read order
case $order in
1)
sorted=$(echo $n | grep -o . | sort | tr  -d '\n')
echo $sorted
;;
2)
sorted=$(echo $n | grep -o . | sort -r | tr  -d '\n')
echo $sorted
;;
*)
echo "Invalid choice"
;;
esac
[foss@fedora Ronit_9145]$ sh a14.sh
Enter Number
686876868
Enter sort order 1. ASCE 2.DSCE
1
666678888
[foss@fedora Ronit_9145]$
```

A18.

Algorithm

1. Take the new directory name from the user
2. Store the current path of the working directory in a variable
3. Rename the directory from old name to new name using mv command
4. Print the new name of the directory

Output:

```
[foss@fedora Ronit_9145]$ cat a18.sh
echo "Enter new directory name"
read new
old=(basename "$PWD")
mv -n "$PWD" "${PWD/*}/$new"
echo "renamed to: $new"
[foss@fedora Ronit_9145]$ sh a18.sh
Enter new directory name
newDir
mv: cannot create directory '/newDir': Permission denied
renamed to: newDir
[foss@fedora Ronit_9145]$
```

A23.

Algorithm

1. Extract the username from each line in etc/passwd
2. Save the usernames in variable
3. Take the username from the list and compare it with all the elements in the list
4. Store the longest/shortest username in a variable
5. Repeat steps 3-4 until no shortest/longest name is found per iteration
6. Print the longest and shortest username

Output:

```
[foss@fedora Ronit_9145]$ cat a23.sh
userlist=$(cut -d: -f1 /etc/passwd)
shortest=
longest=
for username in $userlist; do
if [ -z "$shortest" ] || [ ${#username} -lt ${#shortest} ]; then
shortest=$username
fi
if [ -z "$longest" ] || [ ${#username} -gt ${#longest} ]; then
longest=$username
fi
done
echo "Shortest username: $shortest"
echo "Longest username: $longest"
[foss@fedora Ronit_9145]$ sh a23.sh
Shortest username: lp
Longest username: gnome-initial-setup
[foss@fedora Ronit_9145]$ SS
```

A25.

Algorithm

1. Define all the apha numeric character and store it in a variable

2. Using random function of shell scripting to pickup random character from the stored
string and store it in variable
3. Repeat step 2 eight times to create a 8 character password

Output:

```
[foss@fedora Ronit_9145]$ cat a25.sh
chars="abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789"
length=8
password=""
for i in $(seq 1 $length); do
char=${chars:RANDOM%${#chars}:1}
password="$password$char"
done
echo "Generated password: $password"
[foss@fedora Ronit_9145]$ sh a25.sh
Generated password: g977ZQuB
[foss@fedora Ronit_9145]$
```

A28.

Algorithm

1. Accept the supplied arguments from the user
2. Check if the number of arguments passed are greater than 0.
3. Print the first argument
4. Recursively call the function and start printing 2nd argument onwards
5. Repeat steps 4-5 till arguments are not 0

Code

Output:

```
[foss@fedora Ronit_9145]$ cat a28.sh
function print_args_recursive {
 if [ $# -gt 0 ]; then
        echo $1
        print_args_recursive "${@:2}"
 fi
}
print_args_recursive "apple" "banana" "cherry"
[foss@fedora Ronit_9145]$ sh a28.sh
apple
banana
cherry
[foss@fedora Ronit_9145]$
```

A31

Algorithm

1. Execute the df command and store the output in a variable
2. Read each line of the output
3. Compare each file system and check whether it has 10% of free space
4. Display the name of the file system having 10% less space

Output:

```
[foss@fedora Ronit_9145]$ cat a31.sh
ilesystems=$(df -h)
while read -r line
do
filesystem=$(echo "$line" | awk '{print $1}')
percent_free=$(echo "$line" | awk '{print $5}' | tr -d '%')
if [[ $percent_free -lt 10 ]]; then
echo "$filesystem has less than 10% of free space"
fi
done <<< "$filesystems"
[foss@fedora Ronit_9145]$ sh a31.sh
 has less than 10% of free space
[foss@fedora Ronit_9145]$
```

5. Conclusions & Inferences

Shell Scripts for given questions were written and executed successfully. Usage and syntax of various commands and shell scripts were understood.

6. Post Lab exercise

1. Write a procedure to download the contents of a web page as a formatted text using shell script
2. Explain virtualization and its types. Explain the need of virtualization

8. To study AWK programming

Name of Student		**Roll No.**	
Sign here to indicate that you have read all relevant material provided /available on Moodle while performing and writing this experiment		**Sign:**	

Late Submission Details (if any)

Reason(s) of late submission **Date of practical performance** **Date of practical submission**

References used

1	Name and author of reference book(s) with page nos.	
2	Name and roll nos. of the peers whose help you have taken (if any)	

Rubrics for assessment of Experiment:

Indicator	Poor	Average	Good
Timeliness Maintains Experiment deadline (3)	Experiment not done (0)	One or More than One week late (1-2)	Maintains deadline (3)
Completeness and neatness Complete all parts of Experiment (3)	N/A	< 80% complete (1-2)	100% complete (3)
Originality Extent of plagiarism (2)	Copied it from someone else (0)	At least try to implement but could not succeed (1)	Implemented (2)
Knowledge In depth knowledge of the Experiment (2)	Unable to answer any questions (0)	Unable to answer few questions (1)	Able to answer all questions (2)

Assessment Marks:

Timeliness	
Completeness and neatness	
Originality	
Knowledge	
Total	

Signature of Teacher with date

Course, Subject & Experiment Details

Course & Branch	T.E. (ECS)	Estimated Time	02 Hours Per Week
Current Semester	Semester VI	Subject Name	Linux Server Administration
Chapter No. & Unit	5.1	Chapter Title	Bash Shell Scripting
Experiment Type	Software Performance	Subject Code	ECL 604

Aim & Objective of Experiment

1. To write programs using awk programming

Expected Outcome of Experiment

1. To understand awk programming

Brief Description of the experiment

Refer the website: https://zetcode.com/lang/awk/

1. Print all words included in the **words.txt** file that are longer than eight characters.

```
Terminal - universe@lenovo3: ~/D
File  Edit  View  Terminal  Tabs  Help
universe@lenovo3:~/Desktop/Ronit_9145/8$ awk length($1) < 8 {print $0} word.txt
bash: syntax error near unexpected token `('
universe@lenovo3:~/Desktop/Ronit_9145/8$ awk 'length($1) < 8 {print $0}' word.txt
tree
cup
store
book
cloud
falcon
town
sky
top
war
```

2. Print all words that have three characters from **words.txt**

```
universe@lenovo3:~/Desktop/Piyush_9133/8$ awk 'length($1) == 3' word.txt
ECS
exp
```

3. Print all words in the file **words.txt** along-with the length of each.

```
universe@lenovo3:~/Desktop/Piyush_9133/8$ awk 'length($1) > 0 {print $1, "has",length($1),"chars"}' word.txt
Piyush has 6 chars
ECS has 3 chars
Third has 5 chars
9133 has 4 chars
exp has 3 chars
8 has 1 chars
universe@lenovo3:~/Desktop/Piyush_9133/8$
```

4. Print all students with scores 90+ from **scores.txt**

```
universe@lenovo3:~/Desktop/Piyush_9133/8$  awk '$2 >= 90 { print $0 }' scores.txt
Lucia 95
Joe 92
Sophia 90
universe@lenovo3:~/Desktop/Piyush_9133/8$
```

5. Print average of all scores from **scores.txt**

```
universe@lenovo3:~/Desktop/Piyush_9133/8$  awk '{sum += $2} END { printf("The average score is %.2f\n", sum/NR) }' scores.txt
The average score is 77.56
universe@lenovo3:~/Desktop/Piyush_9133/8$
```

6. Print unique names of cities from **users.txt**

```
universe@lenovo3:~/Desktop/Piyush_9133/8$ awk 'BEGIN {FS=","}  {print $3}' users.txt
 London
 London
 New York
 Portland
 Manchester
 Birmingham
 Los Angeles
 Budapest
universe@lenovo3:~/Desktop/Piyush_9133/8$
```

7. Print names of all females with the number of characters in the name from **users.txt**

```
universe@lenovo3:~/Desktop/Piyush_9133/8$  awk -F, '$4 ~ "F"  {print $1, "has",length($1),"characters"}' users.txt
Jane Doe has 8 characters
Lucy Black has 10 characters
Sofia Harris has 12 characters
universe@lenovo3:~/Desktop/Piyush_9133/8$
```

8. Print total no. of users, total no. of males and females from **users.txt**

```
universe@lenovo3:~/Desktop/Piyush_9133/8$ awk -F, '{ if ($4 ~ "M") {m++} else {f++} } END {printf "users: %d\nmales: %d\nfemales: %d\n", m+f, m, f}' users.txt
users: 8
males: 5
females: 3
universe@lenovo3:~/Desktop/Piyush_9133/8$
```

9. Prints all words that end with e or n from **words.txt**

```
universe@lenovo3:~/Desktop/Piyush_9133/8$ awk '$1 ~ /[e,n]$/ {print $1}' word.txt
tree
store
existence
falcon
town
bookcase
universe@lenovo3:~/Desktop/Piyush_9133/8$
```

10. Remove line numbers from source.c file and save the extracted program (without line numbers) in another file **modified_source.c**

```
universe@lenovo3:~/Desktop/Piyush_9133/8$ awk '{print substr($0, 4) >> "modified_sorce.c"}' source.c
universe@lenovo3:~/Desktop/Piyush_9133/8$
```

11. Print sum of all numbers (line wise) from **values.txt**

```
universe@lenovo3:~/Desktop/Piyush_9133/8$ awk -f calc_sum.awk values.txt
line 1 sum: 73
line 2 sum: 133
line 3 sum: 342
line 4 sum: 287
line 5 sum: 312
line 6 sum: 20
line 7 sum: 293
line 8 sum: 162
line 9 sum: 0
universe@lenovo3:~/Desktop/Piyush_9133/8$
```

12. **Do this for extra credits: Make any one game using awk**

```
universe@lenovo3:~/Desktop/Piyush_9133/8$  touch rock_scissors_paper.awk
universe@lenovo3:~/Desktop/Piyush_9133/8$ awk -f rock_scissors_paper.awk
1 - rock
2 - paper
3 - scissors
9 - end game
2
I have scissors you have paper
Scissors cut paper, you loose
1 - rock
2 - paper
3 - scissors
9 - end game
1
I have rock you have rock
Tie, next throw
1 - rock
2 - paper
3 - scissors
9 - end game
9
```

Conclusions & Inferences

From this experiment we learned about awk programming

Post Lab exercise

1. Write a note on “sed programming” with the help of an example code

SED command in UNIX stands for stream editor and it can perform lots of functions on file like searching, find and replace, insertion or deletion. Though most common use of SED command in UNIX is for substitution or for find and replace. By using SED you can edit files even without opening them, which is much quicker way to find and replace something in file, than first opening that file in VI Editor and then changing it. SED is a powerful text stream editor. Can do insertion, deletion, search and replace(substitution).SED command in unix supports regular expression which allows it perform complex pattern matching.

$sed 's/unix/linux/' ronit.txt.
This command replaces the word “unix” with “linux” in the file

9. To write C program to implement Unix functions

Name of Student		**Roll No.**	
Sign here to indicate that you have read all relevant material provided / available on Moodle while performing and writing this experiment		**Sign:**	

Late Submission Details (if any)

Reason(s) of late submission	Date of practical performance	Date of practical submission

References used

1	Name and author of reference book(s) with page nos.	
2	Name and roll nos. of the peers whose help you have taken (if any)	

Rubrics for assessment of Experiment:

Indicator	Poor	Average	Good
Timeliness			
Maintains Experiment deadline (3)	Experiment not done (0)	One or More than One week late (1-	
Maintains deadline (3)			
Completeness and neatness			
Complete all parts of Experiment (3)	N/A	< 80% complete (1-2)	100% complete (3)
Originality			
Extent of plagiarism (2)	Copied it from someone else (0)	At least try to implement but could not succeed (1)	Implemented (2)
Knowledge			
In depth knowledge of the Experiment (2)	Unable to answer any questions (0)	Unable to answer few questions (1)	Able to answer all questions (2)

Assessment Marks:

Timeliness

Completeness and neatness

Originality

Knowledge

Total

Signature of Teacher with date

1. Course, Subject & Experiment Details

Course & Branch	**T.E. (ECS)**	**Estimated Time**	**02 Hours Per Week**
Current Semester	**Semester VI**	**Subject Name**	**Linux Server Administration**
Chapter No. & Unit	**2.1**	**Chapter Title**	**Unix commands**
Experiment Type	**Software Performance**	**Subject Code**	**ECL 604**

2. Aim & Objective of Experiment

1. **To implement Unix functions using C programming**

3. Expected Outcome of Experiment

1. **To convert Unix commands into C program**

4. Brief Description of the experiment

1. Write a C program to implement the following Unix commands:
   Cat, cp, mv
   (Write algorithm and code for the given question)

Cat command:

code:

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>

#define BUFFER_SIZE 50

int main(int argc, char **argv)
{
  int   file;
  char  buffer[BUFFER_SIZE];
  int   read_size;

  if (argc < 2)
    {
      fprintf(stderr, "Error: usage: ./cat filename\n");
      return (-1);
    }
  file = open(argv[1], O_RDONLY);
```

```
  if (file == -1)
    {
      fprintf(stderr, "Error: %s: file not found\n", argv[1]);
      return (-1);
    }
  while ((read_size = read(file, buffer, BUFFER_SIZE)) > 0)
    write(1, &buffer, read_size);

  close(file);
  return (0);
}
```

output:

```
universe@lenovo3:~/Desktop/Ronit_9145/9$ ./cat catFile.txt
Ronit Patange 9145 ECS

universe@lenovo3:~/Desktop/Ronit_9145/9$
```

cp command:

code:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>

int main(int argc, char ** argv) {
  char buffer[1024];
  int files[2];
  ssize_t count;

  /* Check for insufficient parameters */
  if (argc < 3)
    return -1;
  files[0] = open(argv[1], O_RDONLY);
  if (files[0] == -1) /* Check if file opened */
    return -1;
  files[1] = open(argv[2], O_WRONLY | O_CREAT | S_IRUSR | S_IWUSR);
  if (files[1] == -1) /* Check if file opened (permissions problems ...) */ {
    close(files[0]);
    return -1;
  }

  while ((count = read(files[0], buffer, sizeof(buffer))) != 0)
    write(files[1], buffer, count);

  return 0;
```

}

output:

```
universe@lenovo3:~/Desktop/Ronit_9145/9$ gcc cpCommand.c -o cp
universe@lenovo3:~/Desktop/Ronit_9145/9$ ./cp cp1.txt cp2.txt
universe@lenovo3:~/Desktop/Ronit_9145/9$ cat cp2.txt
File1Contents
universe@lenovo3:~/Desktop/Ronit_9145/9$
```

mv command:
code:

```
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
#include <unistd.h>
#include <stdlib.h>
#include <string.h>

#define SBUF 256
#define DBUF 256

int main(int ac, char * argv[]) {
  DIR * dir_ptr; // the directory
  struct dirent * direntp;

  if (ac == 1) {
    printf("Usage:  %s MOVE\n", argv[0]);
    exit(0);
  }

  if (ac > 1 && ac < 3) {
    printf("Error! few arguments provided ");
    exit(0);
  }

  char src_folder[SBUF];
  char dest_folder[DBUF];
  strcpy(src_folder, argv[1]);
  strcpy(dest_folder, argv[2]);

  dir_ptr = opendir("."); //open directory
  if (dir_ptr == NULL) {
    perror(".");
    exit(1);
  }

  while ((direntp = readdir(dir_ptr)) != NULL) {
```

```
  if (strcmp(direntp -> d_name, dest_folder) != 0) //search file or directory
  {
    printf("found the file %s", dest_folder);

    break;
  } else
    printf("not found");
  break;
 }
 rename(src_folder, dest_folder);
 closedir(dir_ptr);

 return 0;
}
```

output:

```
universe@lenovo3:~/Desktop/Ronit_9145/9$ ./mv mvFile.txt mvFileRenamedBy9145.txt
universe@lenovo3:~/Desktop/Ronit_9145/9$ ./mv mvFile.txt mvFileRenamed.txt
found the file mvFileRenamed.txtuniverse@lenovo3:~/Desktop/Ronit_9145/9$
```

2. Write a C program to redirect output of Unix command “ls -l” in some other file

code:

```
#include <stdio.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>

int main() {
 int pid; //process id
 pid = fork(); //create another process
 if (pid < 0) { //fail
  printf("\nFork failed\n");
  exit(-1);
 } else if (pid == 0) { //child
  execlp("/bin/ls", "ls", "-l", NULL); //execute ls
 } else { //parent
  wait(NULL); //wait for child
  printf("\nchild complete\n");
  exit(0);
 }
}
```

output:

```
universe@lenovo3:~/Desktop/Ronit_9145/9$ gcc lsCommand.c -o ls
universe@lenovo3:~/Desktop/Ronit_9145/9$ ./ls > cp1.txt
universe@lenovo3:~/Desktop/Ronit_9145/9$ cat cp1.txt
total 180
-rw-rw-r-- 1 universe universe 11450 Apr 11 20:08 1.png
-rw-rw-r-- 1 universe universe 14662 Apr 11 20:10 2.png
-rw-rw-r-- 1 universe universe  8185 Apr 11 20:13 3.png
-rwxrwxr-x 1 universe universe 16992 Apr 11 20:06 cat
-rw-rw-r-- 1 universe universe   576 Apr 10 15:27 catCommand.c
-rw-rw-r-- 1 universe universe    24 Apr 11 20:07 catFile.txt
-rwxrwxr-x 1 universe universe 16880 Apr 11 20:09 cp
-rw-rw-r-- 1 universe universe     0 Apr 11 20:21 cp1.txt
-rw-rw-r-- 1 universe universe    14 Apr 10 15:27 cp2.txt
-rw-rw-r-- 1 universe universe   649 Apr 10 15:27 cpCommand.c
-rw-rw-r-- 1 universe universe 13457 Apr 11 20:18 kaushik2.png
-rw-rw-r-- 1 universe universe  8464 Apr 11 20:19 kaushik3.png
-rw-rw-r-- 1 universe universe 11538 Apr 11 20:17 kaushik.png
-rwxrwxr-x 1 universe universe 16872 Apr 11 20:20 ls
-rw-rw-r-- 1 universe universe   445 Apr 10 15:27 lsCommand.c
-rwxrwxr-x 1 universe universe 17112 Apr 11 20:11 mv
-rw-rw-r-- 1 universe universe   982 Apr 10 15:27 mvCommand.c
-rw-rw-r-- 1 universe universe     0 Apr 10 15:27 mvFileRenamed.txt

child complete
universe@lenovo3:~/Desktop/Ronit_9145/9$
```

5. Conclusions & Inferences

All questions were solved successfully and the outputs were observed.

6. Post Lab exercise

1. Write any TWO of the above programs using Java OR Python

```
#!/usr/bin/python
import sys, os
for _arg in sys.argv[1:]:
  try:
        with open(_arg, 'r') as _file:
              for line in iter((_file.readline),""):
              print(line),
   except IOError as _err:
              sys.stderr.write(os.path.basename(sys.argv[0]) + ": " + _arg +
              ": " + _err.strerror + "\n")
```

```
universe@lenovo3:~/Desktop/Ronit_9145/9$ python3 postlab1.py catFile.txt
Ronit Patange 9145 ECS


universe@lenovo3:~/Desktop/Ronit_9145/9$
```

```
import shutil
src=r"/home/Desktop/Ronit_9145/9/post.txt"
dst_path = r"/home /Desktop/Ronit_9145/9/post2.txt" shutil.move(src, dst_path)
print("File moved to:",dst_path)
```

```
universe@lenovo3:~/Desktop/Ronit_9145/9$ python3 post2.py
File moved to: /home/universe/Desktop/Ronit_9145/9/post2.txt
universe@lenovo3:~/Desktop/Ronit_9145/9$
```

10.To select and apply appropriate license for the mini-project

Name of Student		Roll No.	
Sign here to indicate that you have read all relevant material provided / available on Moodle while performing and writing this experiment		Sign:	

Late Submission Details (if any)

Reason(s) of late submission	Date of practical performance	Date of practical submission

References used

1	Name and author of reference book(s) with page nos.	
2	Name and roll nos. of the peers whose help you have taken (if any)	

Rubrics for assessment of Experiment:

Indicator	Poor	Average	Good
Timeliness			
Maintains Experiment deadline (3)	Experiment not done (0)	One or More than One week late (1-2)	Maintains deadline (3)
Completeness and neatness			
Complete all parts of Experiment (3)	N/A	< 80% complete (1-2)	100% complete (3)
Originality			
Extent of plagiarism (2)	Copied it from someone else (0)	At least try to implement but could not succeed (1)	Implemented (2)
Knowledge			
In depth knowledge of the Experiment (2)	Unable to answer any questions (0)	Unable to answer few questions (1)	Able to answer all questions (2)

Assessment Marks:
Timeliness
Completeness and neatness
Originality
Knowledge
Total

Signature of Teacher with date

1. Course, Subject & Experiment Details

Course & Branch	**T.E. (ECS)**	**Estimated Time**	**02 Hours Per Week**
Current Semester	**Semester VI**	**Subject Name**	**Linux Server Administration**
Chapter No. & Unit	**1**	**Chapter Title**	**Open source license**
Experiment Type	**Software Performance**	**Subject Code**	**ECL 604**

2. Aim & Objective of Experiment

1. To select and apply appropriate license to the mini-project

3. Expected Outcome of Experiment

1. To study different open source licenses
2. To understand the procedure of “applying for the open source license”

4. Brief Description of the experiment

1. Write your mini-project problem statement in brief

Hand Sign Language Interpreter**:** To create a portable device using ML/DLModel for understanding language of hand sign for effective communication for Deaf and mute people.

2. Mention the type of open source license for your mini-project and justify your choice

We have used BSD 2-Clause "Simplified" License, Because our project is of social relivance we have used this license so that others can use it and modifify the code for better products to be available later on.

3. Describe the contents of your chosen license

Use, copy and distribute modified source or binary forms of the licensed program, provided that all distributed copies are accompanied by the license

4. Upload the entire source code of your mini-project on github repository with appropriate license. Copy the link to the github code here:

   https://github.com/Hardik7843/Sem6_with_licencse/tree/master

5. Fill the following Google form (Required data: Roll no and name of the students (all group members), mini-project title, github link of the code
   (Note: This need to done by only one student per group)
   https://docs.google.com/forms/d/e/1FAIpQLSdPjM6PupnOBa7TbNTi3NjSKH7lLfn9rpf7Ycv2Rpxycy9Tag/viewform?usp=sf_link

5. Conclusions & Inferences

Selected and applied appropriate license to mini-project of semester 6.

5. References

1. https://www.mend.io/rc-content/wp/the-complete-guide-for-open-source-licenses-2021.pdf
2. https://www.gnu.org/licenses/gpl-howto.en.html
3. https://www.freecodecamp.org/news/how-open-source-licenses-work-and-how-to-add-them-to-your-projects-34310c3cf94/

EXPERIMENT NO.1

AIM: Use of Crimping Tool for RJ45.

THEORY: Crimping an RJ45 Connector Correctly Proper Wiring for Ethernet Cat5/Cat5e/Cat 6Cables

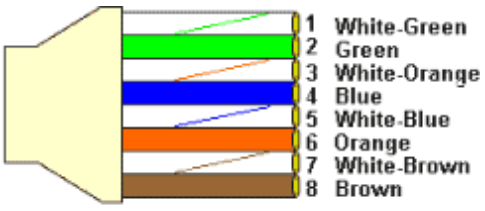
Cables can transmit information along their length. To actually get that information where it needs to go, you need to make the right connections to an RJ45 connector.

Your cable run needs to terminate into a connector, and that connector needs a jack to plug into.

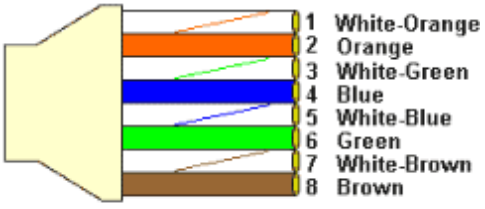
Registered Jack 45 (RJ45) is a standard type of physical connector for network cables. RJ45 connectors are commonly seen with Ethernet cables and networks.

Modern Ethernet cables feature a small plastic plug on each end of the cable. That plug is inserted into RJ45 jacks of Ethernet devices. The term “plug” refers to the cable or “male” end of the connection while the term “jack” refers to the port or “female” end.

T568A or T568B Wiring Standard:



568A CABLE END



568B CABLE END

T568A and T568B are the two colour codes used for wiring eight-position modular plugs. Both are allowed under the ANSI/TIA/EIA wiring standards. The only difference between the two color codes is that the orange and green pairs are interchanged.

There is no transmission differences between T568A and T568B cabling schemes. North America's preference is for T568B. Both ends must use the same standard. It makes no difference to the transmission characteristics of data.

T568B wiring pattern is recognized as the preferred wiring pattern.

STEP 1:

Using a Crimping Tool, trim the end of the cable you're terminating, to ensure that the ends of the conducting wires are even.

STEP 2:

Being careful not to damage the inner conducting wires, strip off approximately 1 inch of the cable's jacket, using a modular crimping toolor a UTP cable stripper.

STEP 3:

Separate the 4 twisted wire pairs from each other, and then unwind each pair, so that you end up with 8 individual wires. Flatten the wires out as much as possible, since they'll need to be very straight for proper insertion into the connector.

STEP 4:

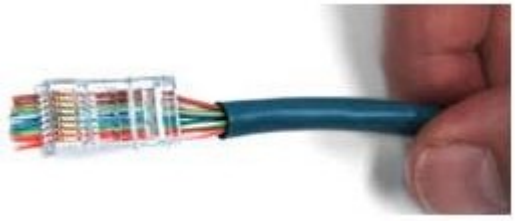
Holding the cable with the wire ends facing away from you. Moving from left to right, arrange the wires in a flat, side-by-side ribbon formation, placing them in the following order: white/orange, solid orange, white/green, solid blue, white/blue, solid green, white/brown, solid brown.

STEP 5:

Holding the RJ45 connector so that its pins are facing away from you and the plug-clip side is facing down, carefully insert the flattened, arranged wires into the connector, pushing through until the wire ends emerge from the pins. For strength of connection, also push as much of the cable jacket as possible into the connector.

STEP 6:

Check to make sure that the wire ends coming out of the connector's pin side are in the correct order; if not, remove them from the connector, rearrange into proper formation, and re-insert. Remember, once the connector is crimped onto the cable, it's permanent. If you realize that a mistake has been made in wire order after termination, you'll have to cut the connector off and start all over again!



STEP 7:

Insert the prepared connector/cable assembly into the RJ45 slot in your crimping tool. Firmly squeeze the crimper's handles together until you can't go any further. Release the handles and repeat this step to ensure a proper crimp.

STEP 8:

If your crimper doesn't automatically trim the wire ends upon termination, carefully cut wire ends to make them as flush with the connector's surface as possible. The closer the wire ends are trimmed, the better your final plug-in connection will be.

STEP 9:

After the first termination is complete, repeat process on the opposite end of your cable

CONCLUSION: Thus, we have studied the use of crimping tool for RJ-45.

Post Lab Assignments:

1. **What are the different ways in which you can classify Networks?**
2. **Architecture of the Internet with diagrams**

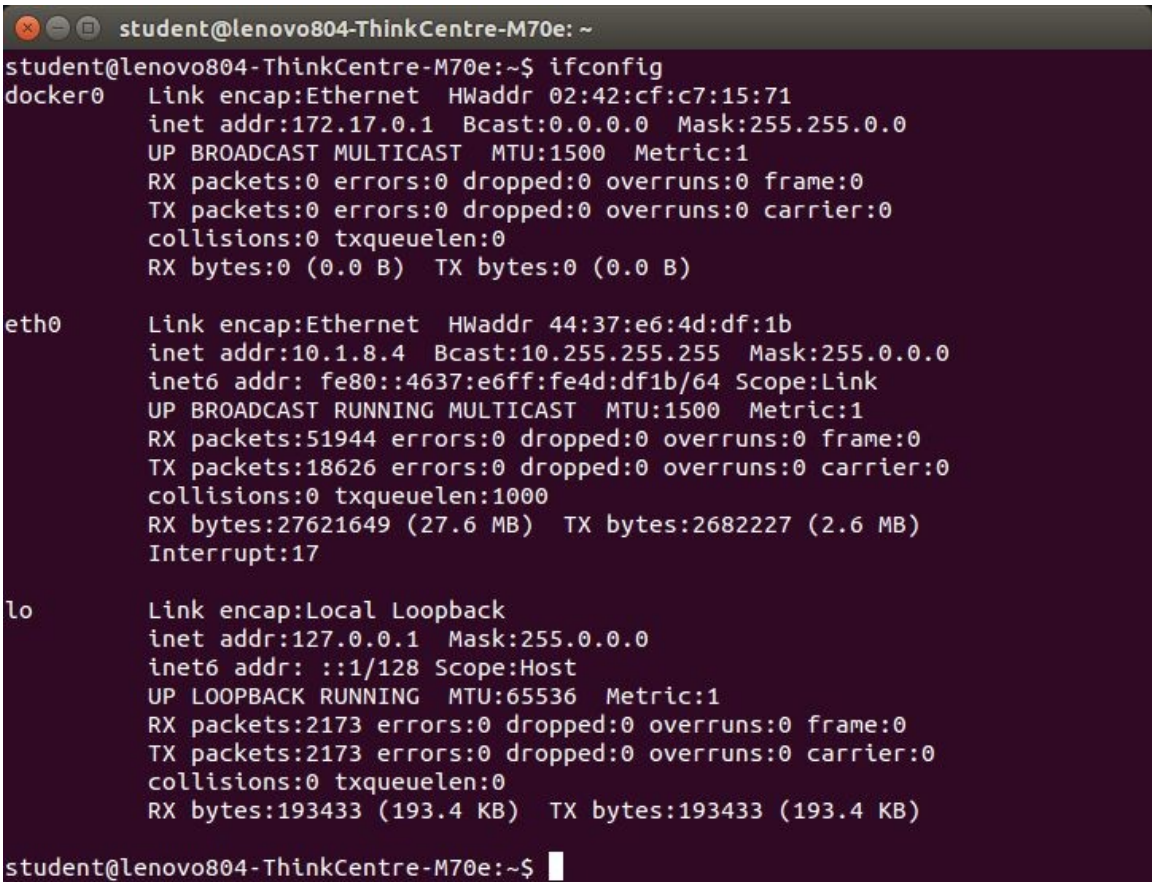
EXPERIMENT NO.2

AIM: Usebasic networkingcommands inLinux(ping,tracert,nslookup, netstat, ARP,RARP, ip, ifconfig,dig,route)

THEORY:

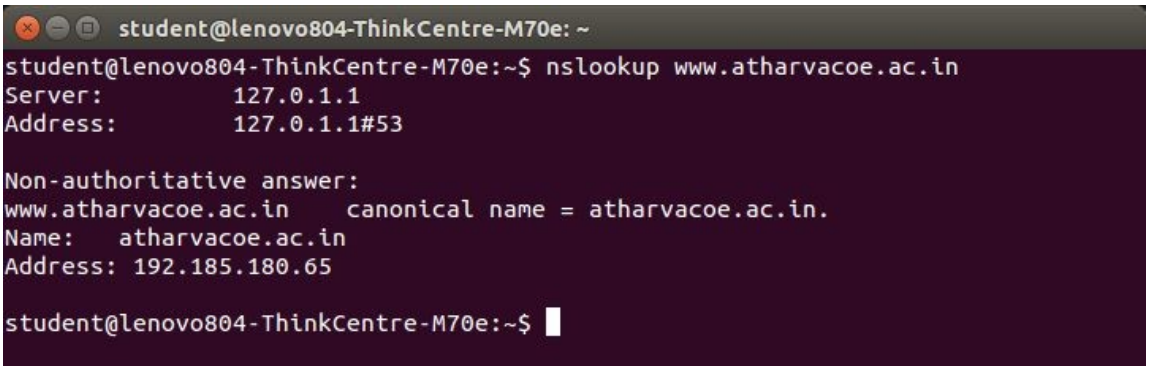
1. **ifconfig**

ifconfig(interface configuration) command is used to configure the kernel-resident network interfaces. It is used at the boot time to set up the interfaces as necessary. After that, it is usually used when needed during debugging or when you need system tuning. Also, this command is used to assign the IP address and netmask to an interface or to enable or disable a given interface.



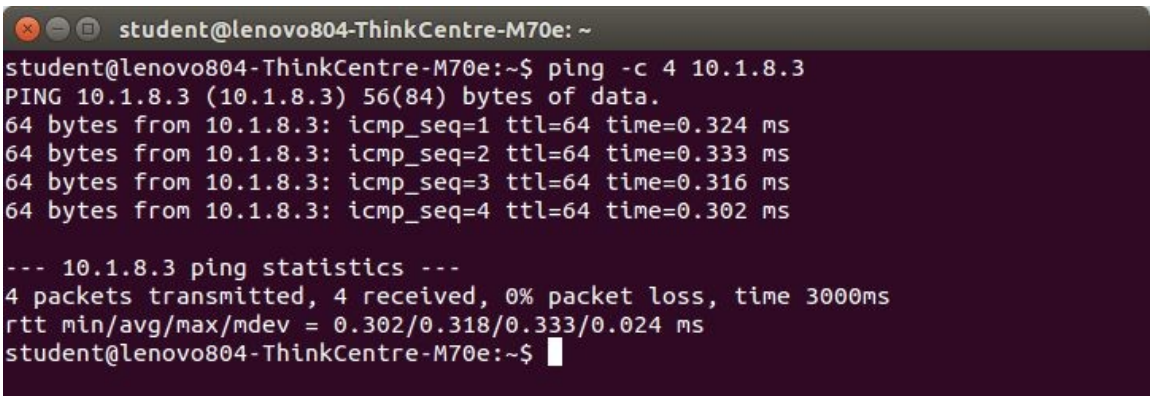
2. NSLOOKUP

Nslookup (stands for "Name Server Lookup") is a useful command for getting information from DNS server. It is a network administration tool for querying the Domain Name System (DNS) to obtain domain name or IP address mapping or any other specific DNS record. It is also used to troubleshoot DNS related problems.



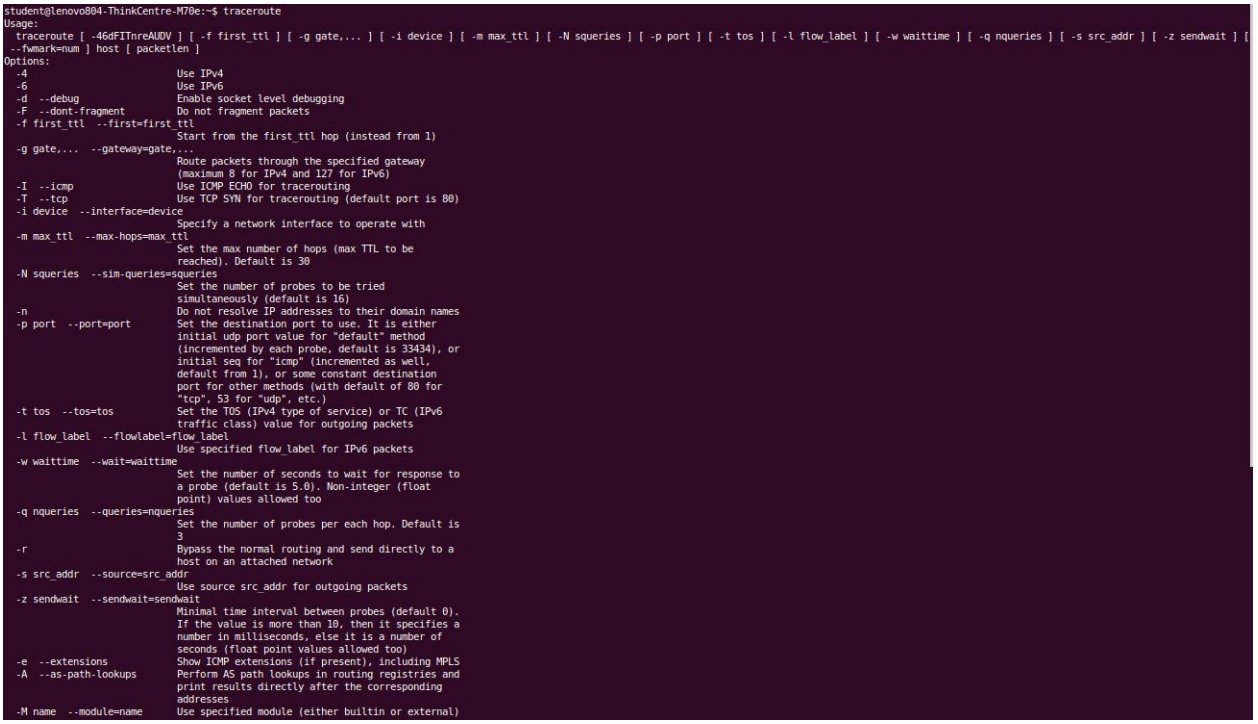
3. Ping

PING (Packet Internet Groper) command is used to check the network connectivity between host and server/host. This command takes as input the IP address or the URL and sends a data packet to the specified address with the message "PING" and get a response from the server/host this time is recorded which is called latency. Fast ping low latency means faster connection. Ping uses **ICMP(Internet Control Message Protocol)** to send an **ICMP echo message** to the specified host if that host is available then it sends **ICMP reply message**. Ping is generally measured in millisecond every modern operating system has this ping pre-installed.



4. TRACEROUTE

traceroute command in Linux prints the route that a packet takes to reach the host. This command is useful when you want to know about the route and about all the hops that a packet takes. Below image depicts how traceroute command is used to reach the Google(172.217.26.206) host from the local machine and it also prints detail about all the hops that it visits in between.



5. Netstat

Netstat command displays various network related information such as network connections, routing tables, interface statistics, masquerade connections, multicast memberships etc.,

```
student@lenovo804-ThinkCentre-M70e: ~
student@lenovo804-ThinkCentre-M70e:~$ netstat -a
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 lenovo804-ThinkC:domain *:*                     LISTEN
tcp        0      0 localhost:ipp           *:*                     LISTEN
tcp        0      0 10.1.8.4:40190          bom05s11-in-f2.1e:https TIME_WAIT
tcp        0      0 10.1.8.4:52797          151.101.2.114:https     TIME_WAIT
tcp        0      0 10.1.8.4:38575          bom05s15-in-f14.1:https ESTABLISHED
tcp        0      0 10.1.8.4:38576          bom05s15-in-f14.1:https ESTABLISHED
tcp        0      0 10.1.8.4:52065          bom05s15-in-f4.1e:https TIME_WAIT
tcp        0      0 10.1.8.4:52796          151.101.2.114:https     TIME_WAIT
tcp        0      0 10.1.8.4:40191          bom05s11-in-f2.1e:https TIME_WAIT
tcp        0      0 10.1.8.4:38634          bom05s15-in-f14.1:https ESTABLISHED
tcp        0      0 10.1.8.4:38637          bom05s15-in-f14.1:https TIME_WAIT
tcp        0      0 10.1.8.4:38573          bom05s15-in-f14.1:https ESTABLISHED
tcp        0      0 10.1.8.4:37409          server-52-222-135:https TIME_WAIT
tcp        0      0 10.1.8.4:41299          a184-30-54-102.de:https TIME_WAIT
```

6. ARP

arp command manipulates the System's ARP cache. It also allows a complete dump of the AR ocol is l 2(I

```
student@lenovo804-ThinkCentre-M70e: ~
student@lenovo804-ThinkCentre-M70e:~$ arp -v
Address                  HWtype  HWaddress           Flags Mask            Iface
10.8.1.3                         (incomplete)                              eth0
10.0.0.3                 ether   08:35:71:f0:35:c0   C                     eth0
10.1.8.3                 ether   44:37:e6:4d:e0:f7   C                     eth0
Entries: 3      Skipped: 0      Found: 3
student@lenovo804-ThinkCentre-M70e:~$
```

7. IP

ip command in Linux is present in the net-tools which is used for performing several network administration tasks. IP stands for Internet Protocol. This command is used to show or manipulate routing, devices, and tunnels. It is similar to *ifconfig* command but it is much more powerful with more functions and facilities attached to it. *ifconfig* is one of the deprecated commands in the net-tools of Linux that has not been maintained for many years. ip command is used to perform several tasks like assigning an address to a network interface or configuring network interface parameters.
It can perform several other tasks like configuring and modifying the default and static routing, setting up tunnel over IP, listing IP addresses and property information, modifying the status of the interface, assigning, deleting and setting up IP addresses and routes.

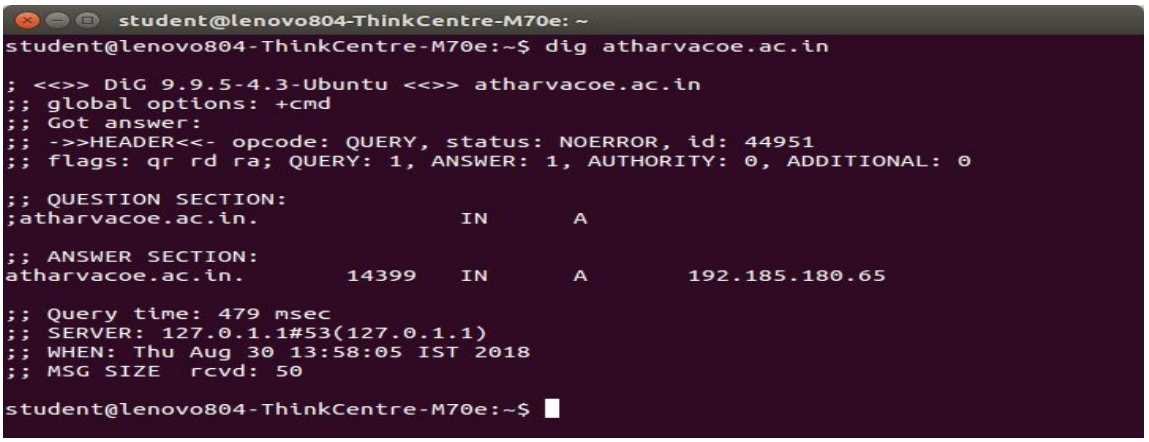
8. Dig

dig command stands for ***Domain Information Groper***. It is used for retrieving information

```
student@lenovo804-ThinkCentre-M70e: ~
student@lenovo804-ThinkCentre-M70e:~$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 44:37:e6:4d:df:1b brd ff:ff:ff:ff:ff:ff
    inet 10.1.8.4/8 brd 10.255.255.255 scope global eth0
       valid_lft forever preferred_lft forever
    inet6 fe80::4637:e6ff:fe4d:df1b/64 scope link
       valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
    link/ether 02:42:cf:c7:15:71 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 scope global docker0
       valid_lft forever preferred_lft forever
student@lenovo804-ThinkCentre-M70e:~$
```

about DNS name servers. It is basically used by network administrators. It is used for

verifying and troubleshooting DNS problems and to perform DNS lookups. Dig command replaces older tools such as nslookup and the host.



CONCLUSION: Hence, in this experiment, we have successfully studied some important networking command and also implemented them in Linux

9. Task-related functions using FreeRTOs

Course, Subject & Experiment Details

Academic year	2022 – 2023	Estimated Time	02 Hours
Course	T.E. (ECS)	Subject Name	Embedded systems and RTOS
Semester	VI	Chapter Title	FreeRTOS
Experiment Type	Coding	Subject Code	ECC 601

Aim & Objective of Experiment

To demonstrate the usage of vTaskSuspend, vTaskResume and vTaskDelete functions of FreeRTOS

Theory:

vTaskSuspend(): This function is used to Suspend a task, the suspended remains in the same state util it is resumed. For this, we need to pass the handle of the tasks that needs to be suspended. Passing NULL will suspend own task.

vTaskResume(): This function is used to resume a suspended task. If the Resumed task has higher priority than the running task then it will preempt the running task or else stays in ready state

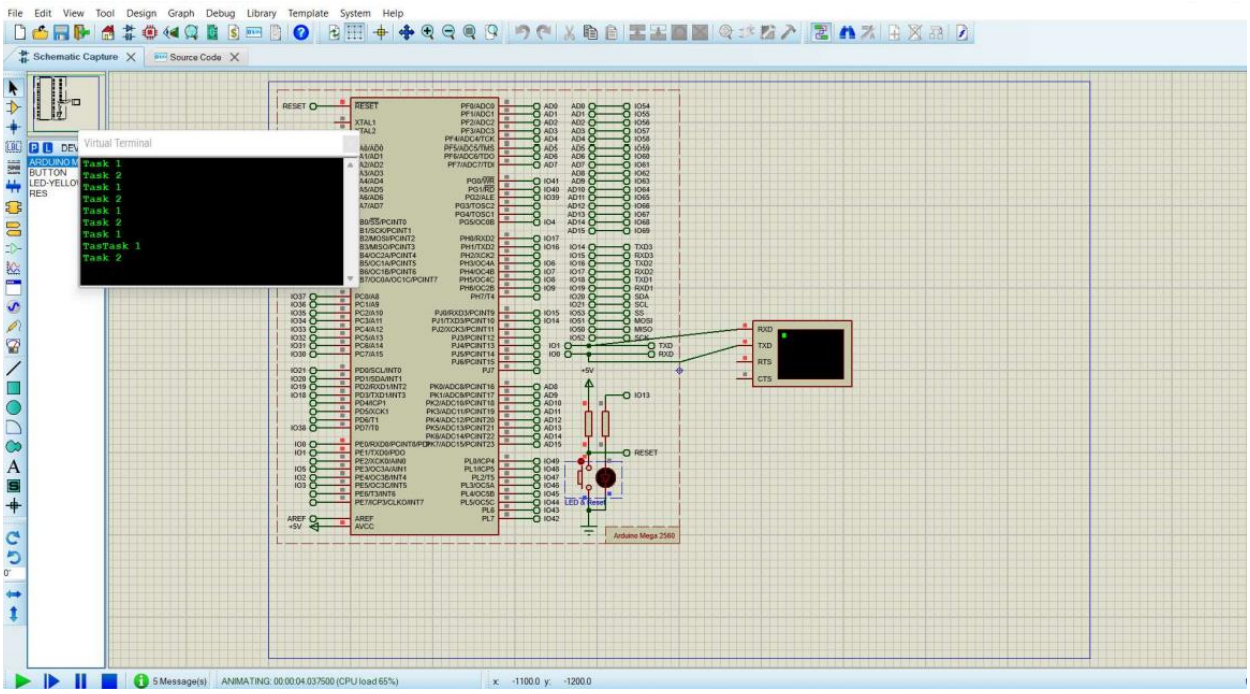
vTaskDelete(): This function is used to delete as task. We need to pass the taskHandle of the task to be deleted.
To delete the own task we should pass NULL as the parameter.

Code:

```
#include <Arduino_FreeRTOS.h>

void Task_Print1(void *param);

void Task_Print2(void *param);

TaskHandle_t Task_Handle1;

TaskHandle_t Task_Handle2;
```

```
int counter = 0;
void setup() {
 // put your setup code here, to run once:
 Serial.begin(9600);
 xTaskCreate(Task_Print1,"Task1",100,NULL,1,&Task_Handle1);
 xTaskCreate(Task_Print2,"Task2",100,NULL,1,&Task_Handle2);
}
void loop() {
 // put your main code here, to run repeatedly:
}
void Task_Print1(void *param){
 (void) param;
 TickType_t getTick;
 getTick = xTaskGetTickCount(); //the getTick will get time from systick of OS
 while(1){
 Serial.println("Task 1");
 counter++;
 if(counter == 15){
 vTaskResume(Task_Handle2);
 }
// vTaskDelayUntil(&getTick,1000 / portTICK_PERIOD_MS);
 vTaskDelay(1000/portTICK_PERIOD_MS);
 }
}
void Task_Print2(void *param){
 (void) param;
```

```
while(1){
counter++;
Serial.println("Task 2");
if(counter == 10){
vTaskDelete(Task_Handle2);
vTaskSuspend(Task_Handle2);
}
vTaskDelay(1000/portTICK_PERIOD_MS);
}
}
```



Postlab questions

1. Explain the instances when tasks may need to be temporarily suspended and resumed later.
2. What is the possible disadvantage of deleting tasks?

10. Inter-process communication using FreeRTOS

Course, Subject & Experiment Details

Academic year	2022 – 2023	Estimated Time	02 Hours
Course	T.E. (ECS)	Subject Name	Embedded systems and RTOS
Semester	VI	Chapter Title	FreeRTOS
Experiment Type	Coding	Subject Code	ECC 601

Aim & Objective of Experiment

To demonstrate the usage of a Mutex using FreeRTOS

Theory:

Inter Process Communication (IPC) refers to a mechanism, where the operating systems allow various processes to communicate with each other. This involves synchronizing their actions and managing shared data.

Mutexes are binary semaphores that include a priority inheritance mechanism. Whereas binary semaphores are the better choice for implementing synchronisation (between tasks or between tasks and an interrupt), mutexes are the better choice for implementing simple mutual exclusion (hence 'MUT'ual 'EX'clusion).

When used for mutual exclusion the mutex acts like a token that is used to guard a resource. When a task wishes to access the resource it must first obtain ('take') the token. When it has finished with the resource it must 'give' the token back - allowing other tasks the opportunity to access the same resource.

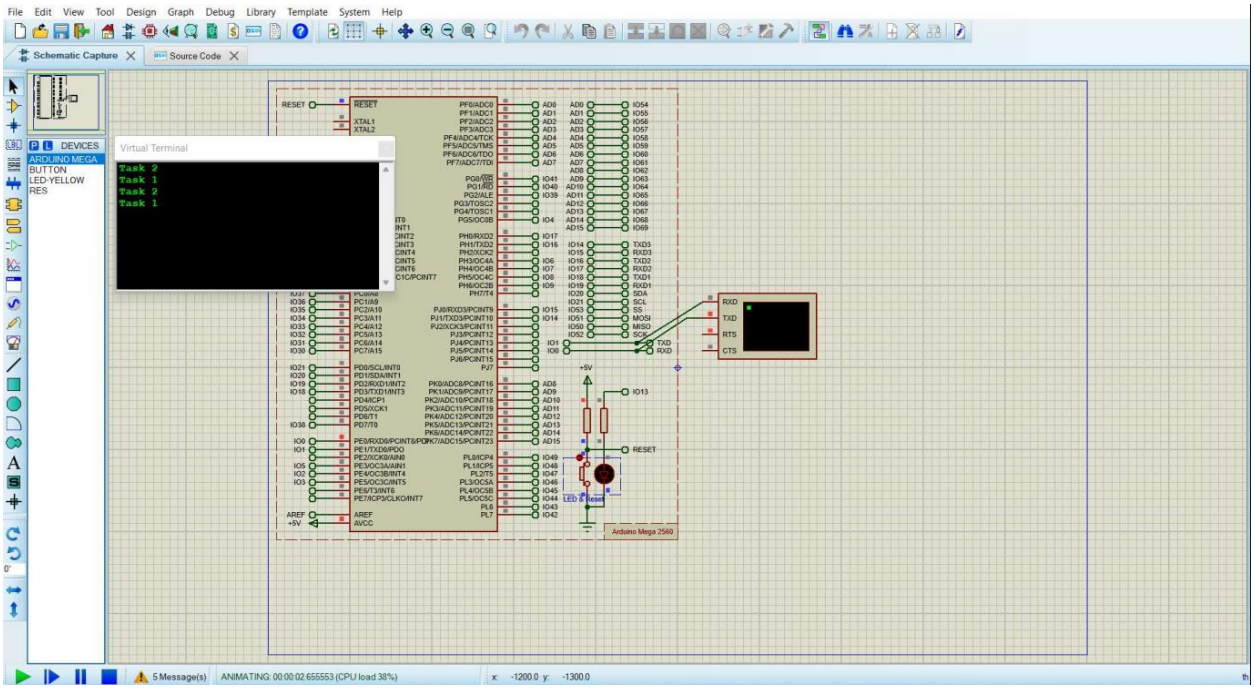
Note that Binary semaphores and mutexes are very similar but have some subtle differences: Mutexes include a priority inheritance mechanism, binary semaphores do not. This makes binary semaphores the better choice for implementing synchronisation (between tasks or between tasks and an interrupt), and mutexes the better choice for implementing simple mutual exclusion.

A semaphore can be used in order to control the access to a particular resource that consists of a finite number of instances

Code:

```
#include <Arduino_FreeRTOS.h>
void Task_Print1(void *param);
void Task_Print2(void *param);
TaskHandle_t Task_Handle1;
TaskHandle_t Task_Handle2;
volatile boolean myMutex = false;
void setup() {
 Serial.begin(9600);
 xTaskCreate(Task_Print1, "Task 1", 100, NULL, 1, &Task_Handle1);
 xTaskCreate(Task_Print2, "Task 2", 100, NULL, 1, &Task_Handle2);
 // put your setup code here, to run once:
}
void loop() {
 // put your main code here, to run repeatedly:
}
void Task_Print1(void *param)
{
(void) param;
 while(1)
 {
 while(myMutex == true);
 for(int i = 0; i < 5; i++)
 {
 Serial.println(i);
 vTaskDelay(1000/portTICK_PERIOD_MS);
```

```
 }
 myMutex = true;
 }
}
void Task_Print2(void *param)
{
 (void) param;
 while(1)
 {
 while(myMutex == false);
 for(int i = 0; i < 5; i++)
 {
 Serial.println(i);
 vTaskDelay(1000/portTICK_PERIOD_MS);
 }
 myMutex = false;
 }
```



Postlab questions:

1. Explain the various IPC techniques supported by FreeRTOS
2. Clearly describe the usage of a queue IPC

Rubrics for TPS activity

Sr.	Contributed to	Novice (1)	Practitioner (3)	Proficient (4)	Mastery (5)
1	**the partnership by following directions, staying on task and listening to partner**	Little evidence of cooperation.	Needed prompting to stay on task.	Stayed on task and cooperated with partner.	Stayed on task, cooperated with partner and demonstrated leadership by assisting others with cooperation and staying on task.
2	**completion of worksheets**	No work Completed	Little effort and minimal answers and facts. Responses on worksheet are difficult to understand. Missed the key points.	Answers reflect basic information requested.	Responses are complete. Answers reflect in-depth thought and thinking outside the box. Detailed and inferences made.
3	**presentation of assigned role**	Did not present to the class.	Minimal presentation with little information shared.	Presented his or her portion to the class.	Presented his or her portion to the class using eye contact and good volume.

4	Timeline	
	Submitted before timeline (5)	Submitted after timeline (0)

Rubric	Marks
1	
2	
3	
4	
Total Out of 20	

Roll No	Name of student	Teacher In-charge	Sign with date
		Vaibhav V. Godbole /Dipali Koshti	