Jaypee Institute of Information Technology, Noida
T1 Examination, 2023
B.Tech VI Semester
Course Title: Open Source Software Development

Q1.[CO1] How do copyleft and copyright licenses impact the development and distribution of software? Can you provide examples to support your answer? **[2 Marks]**
Marking scheme: 1 mark for each
Solution:

Copyleft and non-copyleft licenses can have a significant impact on the development and distribution of software.

Copyleft licenses, such as the GPL (General Public License), require that any derivative works or modifications to the original software must also be released under the same license. This ensures that the source code remains freely available and can be modified and distributed by anyone. This can lead to a large and active community of developers contributing to and improving the software. An example of a popular software that uses the GPL license is the Linux operating system.

On the other hand, non-copyleft licenses, such as the BSD (Berkeley Software Distribution) license or the MIT license, do not have the same restrictions as copyleft licenses. This allows developers to use the software in proprietary projects, modify it, and distribute it without having to release the source code. This can make it easier for commercial organizations to adopt and use the software in their products, which can lead to increased development and wider distribution. An example of a popular software that uses a non-copyleft license is the Apache web server software.

Hence, copyleft licenses can encourage a large and active community of developers to contribute to a project, while non-copyleft licenses can make it easier for commercial organizations to adopt and use the software.

Q2. [CO2] Person A and B belong to same IT team in an MNC but are in different time zones. Person A made some changes to the code and committed on to the main system before logging off for the day. Later, person B logged in his time zone, and now he wants to examine the changes along with adding some of his code as well. Write all the steps followed by person A and later by person B on the git for this work flow. **[3 Marks]**
Marking scheme: 1 mark for person A's step and 2 marks for person B's steps.
Solution:
Person A's steps:
1. Make changes to the code in the local working directory.
2. Add the changes to the staging area using the command **git add**.
3. Commit the changes to the local repository using the command **git commit**.
4. Push the changes to the remote repository on the main system using the command **git push**.

Person B's steps:
1. Pull the changes from the remote repository using the command **git pull**.
2. Make changes to the code in the local working directory.
3. Add the changes to the staging area using the command **git add**.
4. Commit the changes to the local repository using the command **git commit**.
5. Push the changes to the remote repository using the command **git push**.

It is important to note that Person B should first pull the changes made by Person A before making any further changes to the code to ensure that there are no conflicts between the two versions of the code. Additionally, if there are conflicts between the changes made by Person A and Person B, Git will prompt Person B to resolve the conflicts before pushing the changes to the remote repository.

Q3. [CO3] [2+1.5+1+1+1.5=7 marks] Create a file names 'student.txt' having following information:

Roll	Name	Batch	Subject1	Subject2	Subject 3
1	John	A1	85	90	80
2	Mary	A1	70	80	90
3	Peter	A2	80	75	70
4	Sarah	A2	90	85	95
5	Tom	A3	60	70	80

Marking scheme: Full marks will be given for a correct answer and partial marks for a partially correct answer**.**

a) Write an awk to print details of all the students having marks more than 75 in any one of the subjects.

```
awk 'NR == 1 || ($4 > 75 || $5 > 75 || $6 > 75)' student.txt
```

or

```
awk '{if ($4 > 75 || $5 > 75 || $6 > 75) {print}}' student.txt
```

b) Write an awk to print the name of the student who stood first in the class.

```
awk 'NR == 1 {next} {total = $4 + $5 + $6; if (total > max) {max = total; name = $2}} END
{print "First in the class:", name}' student.txt
```

or

```
awk 'NR>1{sum=$4+$5+$6;if(sum>max){max=sum;name=$2}} END{print name}' student.txt
```

c) Write an awk to change the default delimiter of the file to pip "|" symbol.

```
awk -F'\t' -v OFS='|' '{print}' student.txt
```

or

```
awk -F"|" '{print}' student.txt
```

d) Write an awk to print average marks in each subject.

```
awk 'NR == 1 {for (i=4; i<=NF; i++) {total[i] = 0; count[i] = 0}} {for (i=4; i<=NF; i++) {total[i]
+= $i; count[i]++}} END {for (i=4; i<=NF; i++) {printf "Average marks in %s: %.2f\n", $i,
total[i]/count[i]}}' student.txt
```

or

awk 'NR>1{sum1+=$4;sum2+=$5;sum3+=$6} END{printf "Average Marks in Subject1: %.2f\nAverage Marks in Subject2: %.2f\nAverage Marks in Subject3: %.2f\n", sum1/(NR-1), sum2/(NR-1), sum3/(NR-1)}' student.txt

e) Write an awk to find the student name who stood 2nd in the class.

awk 'NR == 1 {next} {total = $4 + $5 + $6; if (total > max1) {max2 = max1; name2 = name1; max1 = total; name1 = $2} else if (total > max2) {max2 = total; name2 = $2}} END {print "Second in the class:", name2}' student.txt

or

awk 'NR>1{total=$4+$5+$6; if(total>max1){max2=max1; max1=total; name2=name1; name1=$2} else if(total>max2){max2=total; name2=$2}} END{print "The person who stood 2nd in the class is:", name2}' student.txt

Q4.[CO3] Write linux commands for the following:
Marking scheme: Full marks will be given for a correct answer and partial marks for a partially correct answer**.**

1. Create a directory (experiment) with the text files (file1.text, file2.txt, file3.txt). **[1 Marks]**

```
mkdir experiment

cd experiment

touch file1.txt file2.txt file3.txt
```

2. Fill each text file with sample data containing employee id, name, and city, then use the 'sed' command to replace the names beginning with 'k' with 'Linux'. **[1 Marks]**

```
echo "121 kajal MainSt" >> file1.txt

echo "123 shyam ElmSt" >> file1.txt

echo "134 kate PineAve" >> file2.txt

echo "345 dimple OakAve" >> file2.txt

echo "345 karan Maple" >> file3.txt

sed -i 's/^.* k/Linux/' file*.txt
```

3. Use the grep command to search the word "Linux" in all ".txt" files in the "experiment " directory and display the file names and line numbers where the word is found. **[1.5 Marks]**

```
grep -rnw 'experiment' -e 'Linux' --include "*.txt"

Explanation:
```

- **grep**: This is the command-line utility that is used to search for patterns within files.
- **-rnw**: These are command-line options that specify the behavior of **grep** as follows:
    - **-r**: This option tells **grep** to search recursively in all subdirectories.
    - **-n**: This option tells **grep** to print the line numbers where the matching pattern occurs.
    - **-w**: This option tells **grep** to match whole words only (i.e., not partial matches).
- **'experiment'**: This is the search pattern that **grep** will look for in the files.
- **-e**: This option tells **grep** that the next argument is the search pattern.
- **'Linux'**: This is the search pattern that **grep** will look for in the files.
- **--include "*.txt"**: This option tells **grep** to search only in files that have the **.txt** extension.

4. Suppose the 'experiment' directory contains several subdirectories, each containing music files with different extensions. Write a command that will find all ".mp3" files in the "experiment" directory and move them to a new subdirectory called "mp3". **[1.5 Marks]**

```
find experiment/ -name '*.mp3' -exec mv {} experiment/mp3/ \;
```

Explanation:

- **find**: This command is used to search for files and directories based on specified criteria.
- **experiment/**: This is the directory where **find** will start searching for files.
- **-name '*.mp3'**: This option tells **find** to search for files with the **.mp3** extension.
- **-exec**: This option tells **find** to execute a command on each file that matches the search criteria.
- **mv**: This is the command that will be executed on each file that matches the search criteria. It moves the file to the **experiment/mp3/** directory.
- **{}**: This is a placeholder for the file that **find** finds that matches the search criteria.
- **\;**: This is used to terminate the **-exec** command.

5. Use the top and ps commands to identify processes that are consuming high amounts of resources. **[1 Marks]**

```
$ top # Use the ps command to display information about running
processes, sorted by CPU usage

$ ps -eo pid,user,%cpu,%mem,comm --sort=-%cpu | head
```

The **ps** command in this example uses the **-eo** option to specify the output format, which includes the process ID (pid), user, CPU usage (**%cpu**), memory usage (**%mem**), and command name (comm). The **--sort=-%cpu** option sorts the processes based on the CPU usage, with the processes using the most CPU resources listed first. The **head** command is used to display only the first 10 processes in the list.

Q5.[CO3]Suppose you have a file called "inventory.txt" with the following format: item_name, item_description, item_quantity,item_price. Write a Sed command that will find all the items that have a quantity of zero, and replace the item price with "OUT OF STOCK". The output should be in the same format as the input. **[2 Marks]**

Marking scheme: Full marks will be given for a correct answer and partial marks for a partially correct answer.

Solution:

```
sed -i 's/^\([^,]*,[^,]*,0,\)[^,]*/\1OUT OF STOCK/' inventory.txt
```

- **s/** is the substitution command in **sed**.
- ^ matches the beginning of a line.
- **\([^,]*,[^,]*,0,\)** is a regular expression that matches three comma-separated fields followed by a zero at the beginning of a line. The parentheses capture the matched text as a group so it can be referenced later in the substitution.
- **[^,]*** matches any number of characters that are not commas.
- **/\1OUT OF STOCK/** is the replacement text. **\1** refers to the first captured group in the regular expression, which is the three comma-separated fields and zero at the beginning of the line. This text is replaced with the text "OUT OF STOCK".
- / marks the end of the substitution command, and ' marks the end of the entire **sed** command.