

## Particulars of the Experiments Performed

### CONTENTS

Expt No.	Date	Experiment	Marks Obtained	Page No.
1	25/10/2021	Shell script to check whether a year is leap year or not		1
2	25/10/2021	Shell script to find the Area of a circle		3
3	25/10/2021	Shell script to check if a No. is Zero/positive/Negative		5
4	25/10/2021	Shell script to find largest of three numbers		7
.				
5	8/11/2021	Shell script to Compute factorial of a number		9
6	8/11/2021	Shell script to Compute gross Salary		11
7	8/11/2021	Shell script to Convert temper from farenheit to centigrade		13
8	8/11/2021	Shell script to perform Arithmetic on two numbers		15
9	15/11/2021	Shell script to find sum of even numbers upto n		17
10	15/11/2021	Print all Combinations of 1 2 3		19
11	15/11/2021	Power of a number		21
12	15/11/2021	Find Sum of natural numbers		23

# Particulars of the Experiments Performed

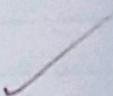
## CONTENTS

Expt No.	Date	Experiment	Marks Obtained	Page No.
13	29/11/2021	Shell script to display full class of a student		25
14	29/11/2021	To find fibonacci series upto n		31
15	29/11/2021	Count the number of vowels of a string		33
16	29/11/2021	To check number of lines, words, characters in a file		35
17	03/11/2022	Program to output the contents of its environment list		38
18	3/11/2022	To emulate Unix ln command		39
19	3/11/2022	POSIX compliant program that prints the posix defined configuration option supported on any given system using the feature test macros		40
20	3/11/2022	Program which demonstrates IPC b/n a reader process and a writer process. Use mknod, open, read, write and close apis in your program.		42

O/P

Enter year : 700

Not Leap Year



1) Shell script to check whether a year is a leap year or not

```
#!/bin/bash
echo -n "Enter Year: "
read y
if [ $(y \% 4) -eq 0 ]
then
    if [$(y \% 400) -ne 0 -a $(y \% 100) -eq 0]
    then
        echo 'Not Leap Year'
    else
        echo 'Leap Year'
    fi

```

O/P

Enter radius

7

The area is 147

2) Shell Script to find Area of circle

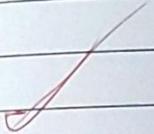
```
# ! /bin/sh
```

```
echo 'Enter radius'
```

```
read r
```

```
area= `echo 22/7 \* $r \* \$r/bc`
```

```
echo 'The area is : $area'
```



o/p

Enter a number

0

Zero



3 shell script to check if a number is zero / positive / negative

```
# ! bin/sh
```

```
echo 'Enter a number'  
read n
```

```
if [ $n -gt 0 ]
```

```
then
```

```
echo 'positive'
```

```
elif [ $n -lt 0 ]
```

```
then
```

```
echo 'Negative'
```

```
else
```

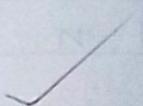
```
echo 'Zero'
```

fi

o/p

• / big03      28      12      25

28 is the largest



4) shell script to find largest of 3 nos.

```
# ! /bin/sh
```

```
if [ $# -ne 3 ]
```

```
then
```

```
echo 'Insufficient parameters'  
exit 128
```

```
fi
```

```
if [ $1 -gt $2 -a $1 -gt $3 ]
```

```
then
```

```
echo "$1 is the largest"
```

```
elif [ $2 -gt $1 -a $2 -gt $3 ]
```

```
then
```

```
echo "$2 is the largest"
```

```
else
```

```
echo "$3 is the largest"
```

O/P:

• /fac.sh 5

$$5! = 120$$

Lab 2

5 shell script to compute factorial of a number

```
#!/bin/bash
```

```
if [ $# -ne 1 ]
```

```
then
```

```
echo 'Insufficient params'
```

```
exit 128
```

```
fi
```

```
ans = 1
```

```
n = 1
```

```
while [ $n -le $1 ]
```

```
do
```

```
ans = `expr $ans \* $n`
```

```
n = `expr $n + 1`
```

```
done
```

```
echo "$1! = $ans"
```

O/P :

$$\cdot / gross - sal \cdot sh \quad 1000$$

$$\text{Total Salary} = 1300$$



6 Compute gross-salary

#!/bin/sh

if [ \$# -ne 1 ]

then

echo 'Enter basic salary'

exit 128

fi

hra = `expr \$1 \\* 20 / 100 | bc`

da = `expr \$1 \\* 10 / 100 | bc`

sal = `expr \$1 + \$hra + \$da | bc`

echo "Total salary = \$sal"

O/P

• / temp-sh 41

41 degrees in celsius = 5



7 Script to convert temperature from  
fahrenheit to celsius

#!/bin/sh

if [ \$# -ne 1 ]

then

echo 'Enter temp in F'

fi

x= `expr \$1 -32 / 9`

c= `expr \$x \\* 5 / 9`

echo "\$1 degrees in celsius = \$c"

O/P

• / arithmetic, sh      20    8

sum = 28

Difference = 12

product = 160

Quotient = 2

Remainder = 4

8 Script to perform arithmetic on two numbers

```
#!/bin/sh
```

```
if [ $# -ne 2 ]
```

then

```
echo 'Enter 2 numbers'
```

```
exit 128
```

fi

```
s= `expr $1 + $2 | bc`
```

```
d= `expr $1 - $2 | bc`
```

```
m= `expr $1 * $2 | bc`
```

```
q= `expr $1 / $2 | bc`
```

```
r= `expr $1 % $2 | bc`
```

```
echo "Sum = $s"
```

```
echo "Difference = $d"
```

```
echo "Product = $m"
```

```
echo "Quotient = $q"
```

```
echo "Remainder = $r"
```

O/P :-

bash sum\_of\_Even.sh

6

Sum = 12



## LAB 3

9. Shell Script to find sum of even numbers upto n

```
#!/bin/bash
```

```
if [ $# -ne 1 ]
```

```
then
```

```
echo "Enter range"
```

```
exit 128
```

```
fi
```

```
sum=0
```

```
for ((i=0; i<=$1; i++))
```

```
do
```

```
if ((i % 2 == 0))
```

```
then
```

```
sum=$((i + sum))
```

```
fi
```

```
done
```

```
echo "Sum = $sum"
```

O/P :-

bash combor.sh

1 2 3

1 3 2

2 1 3

2 3 1

3 1 2

3 2 1

10) Print all combinations of 1 2 3

#!/bin/bash

for i in 1 2 3

do

for j in 1 2 3

do

for k in 1 2 3

do

if [ \$i -ne \$j -a \$j -ne \$k  
-a \$i -ne \$k ]

then

echo "\$i - ne \$k"]

fi

done

done

✓

O/P:-

bash power.sh 2 4

$$2^4 = 16$$

## 11 Power of a number

tit: /bin/bash

if [ \$# -ne 2 ]

then

echo 'Enter 2 numbers'

exit

fi

res=`expr \$1 \(\*\ \${\\$1}^{\*\*}\ \$2)\) bc`

echo "\$1 ^ \$2 = \$res"

O/p:

bash naturalnosum.sh 3

Sum = 6

12 shell script to find sum of natural numbers

```
#!/bin/bash
```

```
if [ $# -ne 1 ]
```

```
then
```

```
echo 'Enter range'
```

```
exit 128
```

```
fi
```

```
Sum=0
```

```
for (( i=0 ; i <= $1 ; i++ ))
```

```
do
```

```
Sum=$((i+Sum))
```

```
done
```

```
echo "Sum = $Sum"
```

25  
O/P

Enter CIE marks in all subs

50

48

49

47

Enter SEE in all subs

100

97

95

98

Grade for subject 1 is S grade

Grade for subject 2 is S grade

Grade for subject 3 is S grade

Grade for subject 4 is S grade

## WEEK - 4

13 Shell script to display the pass class of a student.

```
#!/bin/bash
```

```
# CIE marks
```

```
echo "Enter CIE marks in all subs"
cie=()
```

```
for i in {0..3}
do
read cie[$i]
done
```

```
#SEE marks
```

```
echo "Enter SEE in all subs"
see=()
```

```
for i in {0..3}
do
read see[$i]
done
```

```
# calculate marks
```

Analysing

Answers

Ques.

Level of significance

Test statistic

Value = 8.218

Decision rule

Reject H<sub>0</sub> if

|T| > 2.776

|T| > 2.776

|T| > 2.776

Conclusion

totalMarks = ()

for i in {0..3}

do

totalMarks[\$i] = expr \${cie[i]} + \${sec[i]} / 2  
| bc

done

for i in {0..3}

do

echo "Grade for subject \$i is = "

if [ \${totalMarks[\$i]} -ge 90 ]

then

echo "S Grade"

elif [ \${totalMarks[\$i]} -ge 80 ]

then

echo "A Grade"

elif [ \${totalMarks[\$i]} -ge 70 ]

then

echo "B Grade"

elif [ \${totalMarks[\$i]} -ge 60 ]

then

echo "C Grade"

elif echo "E Grade"

elif [ \${totalMarks[\$i]} -ge 50 ]

then

echo "D Grade"

E1

[REDACTED]

18, 13 mi S [REDACTED]

ab

[REDACTED] [REDACTED] [REDACTED]  
sd 1

[REDACTED]

18, 13 mi S [REDACTED]

ab

[REDACTED] [REDACTED] [REDACTED]

[REDACTED] [REDACTED] [REDACTED] [REDACTED]

ab

18, 13 ab

[REDACTED] [REDACTED] [REDACTED] [REDACTED]

ab

[REDACTED] R ab

[REDACTED] [REDACTED] [REDACTED] [REDACTED]

ab

[REDACTED] ab

[REDACTED] [REDACTED] [REDACTED] [REDACTED]

ab

[REDACTED] ab

[REDACTED] [REDACTED] [REDACTED] [REDACTED]

ab

[REDACTED] ab

```
elif [ ${totalMarks[$i]} -ge 40 ]
then
    echo "E grade"
else
    echo "Fail"
fi
done
```

✓

Si

C) + [d] / - [d]

[d] - [d] ni 3 rot  
ob[d] - [d] ni 3 rot  
ob

2006

[d] - [d] ni 3 rot  
ob[d] - [d] ni 3 rot  
ob[d] - [d] ni 3 rot  
ob

sharp 3" edge

[d] - [d] ni 3 rot  
ob

2006

[d] - [d] ni 3 rot  
ob[d] - [d] ni 3 rot  
ob

2006

"sharp 3" edge

[d] - [d] ni 3 rot  
ob

2006

"sharp 3" edge

[d] - [d] ni 3 rot  
ob

2006

"sharp 3" edge

[d] - [d] ni 3 rot  
ob

2006

```
elif [ ${totalMarks[$c]} -ge 40 ]
```

```
then
```

```
echo "E grade"
```

```
else
```

```
echo "Fail"
```

```
fi
```

```
done
```

O/P:

Enter n: 8

0 1 1 2 3 5 8 13

14) Shell script to find the Fibonacci Series up to n

```
#!/bin/bash
```

```
echo -e "Enter n: \c"
```

```
read n
```

```
first = 0
```

```
second = 1
```

```
if ((n > 1))
```

```
then
```

```
    printf "%d" $first
```

```
fi
```

```
if ((n >= 2))
```

```
then
```

```
    printf "%d" $second
```

```
fi
```

```
((n = n - 2))
```

```
while ((n > 0))
```

```
do
```

```
((third = first + second))
```

```
printf "%d" $third
```

```
((first = second))
```

```
((second = third))
```

```
((n--))
```

```
done
```

```
echo "# add a newline"
```

O/P

Enter string : Hey there

No. of vowels : 3

15 Shell script to count the number of vowels of a string.

```
#!/bin/bash
```

```
echo -e "Enter a string: \c"
read string
```

```
numVowels=`grep -o "[AEIOUaeiou]" <<
$string | wc -l`
```

```
echo "No. of vowels: $numVowels"
```

O/P :

test.txt

1. Hey there! I'm Vineeth, a CS Undergrad.
2. I build websites and software
3. I'm a web developer, software developer and an ethical hacker

fileCounter.sh

bash fileCounter.sh test.txt

Line count = 2

Word count = 22

Character count = 133



16 Shell Script to check number of lines, words, characters in a file.

```
#!/bin/bash
```

```
if [ $# -ne 1 ]
```

```
then
```

```
echo 'Enter filename'
```

```
exit 128
```

```
fi
```

```
echo "Line Count = `wc -l $1`"
```

```
echo "Word Count = `wc -w $1`"
```

```
echo "Character Count = `wc -m $1`"
```

AM  
6/12/2021

Write a C/C++ program that outputs the contents of its Environment list.

```
#include<stdio.h>

int main(int argc, char *argv[])
{
    int i;
    char **ptr;
    extern char **environ;

    printf("\n\n Start success\n\n");
    for(ptr = environ; *ptr != 0; ptr++)
        printf("%s\n", *ptr);
    printf("\n\n End success\n\n");

    return 0;
}
```

Write a C/C++ program to emulate the unix ln command.

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

int main(int argc, char *argv[])
{
    if (argc < 3 || argc > 4 || argc == 4 &&
        strcmp(argv[1], "-s")) {
        printf("Usage: ./a.out [-s]<org-file><new-link>\n");
        return 1;
    }

    if (argc == 4) {
        if ((symlink(argv[2], argv[3])) == -1)
            printf("Cannot create symbolic link\n");
        else
            printf("Symbolic link created\n");
    }
    else {
        if ((link(argv[1], argv[2])) == -1)
            printf("cannot create hard link\n");
        else
            printf("Hard link created\n");
    }
    return 0;
}
```

write a C/C++ POSIX compliant program that prints the POSIX defined configuration options supported on any given system using feature test macros.

```
#define _POSIX_SOURCE
#define _POSIX_C_SOURCE 199309L
#include <stdio.h>

int main()
{
    #ifdef _POSIX_JOB_CONTROL
        printf ("System supports job control\n");
    #else
        printf ("System does not support job
control\n");
    #endif

    #ifdef _POSIX_SAVED_IDS
        printf ("System supports saved set-UID
and saved set-GID\n");
    #else
        printf ("System does not support saved
set-UID and saved set-GID\n");
    #endif
}
```

```

#define _POSIX_CHOWN_RESTRICTED
printf("chown-restricted option is %d\n",
->POSIX_CHOWN_RESTRICTED);
#else
printf("System does not support chown-
restricted option\n");
#endif

#define _POSIX_NO_TRUNC
printf("pathname trunc option is %d
\n", -POSIX_NO_TRUNC);
#else
printf("System does not support
system-wide pathname trunc option
\n");
#endif

#define _POSIX_VDISABLE
printf("Disable character for terminal
files is %d\n", -POSIX_VDISABLE);
#else
printf("System does not support
->POSIX_VDISABLE\n");
#endif

return 0;
}

```

Write C/C++ program which demonstrates Interprocess communication between a reader process and a writer process. Use mkfifo, open, read, write and close APIs in your program

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <errno.h>

int main(int argc, char *argv[])
{
    int fd;
    char buf[256];

    if (argc != 2 && argc != 3) {
        printf("USAGE %s <file> [<arg>]\n", argv[0]);
    }

    if (argc == 2) {
        fd = open(argv[1], O_RDONLY | O_NONBLOCK);
    }

    while (read(fd, buf, sizeof(buf)) > 0)
        printf("%s", buf);
}
```

```
else {  
    fd = open(argv[1], O_WRONLY);  
    write(fd, argv[2], strlen(argv[2]));  
}  
  
close(fd);  
}
```