

## OUTPUT

CASE 1 :- ENTER THE YEAR  
2000  
2000 is a leap year

CASE 2 : ENTER THE YEAR  
1900  
1900 is not a leap year.

CASE 3 : ENTER THE YEAR  
2012  
2012 is a leap year.

CASE 4 : ENTER THE YEAR  
2013  
2013 is not a leap year. ✓

# Shell script to find if a given year is leap year or not.

#!/bin/sh

echo "Enter the year"

read y

if [ `expr \$y % 400` -eq 0 ]

then

echo " \$y is a leap year "

elif

then

[ `expr \$y % 100` -eq 0 ]

echo " \$y is not a leap year "

elif

then

[ `expr \$y % 4` -eq 0 ]

echo " \$y is a leap year "

else

echo " \$y is not a leap year "

fi

OUTPUT

ENTER THE RADIUS

12

AREA IS 452.16

# Shell script to find the area of a circle.

#!/bin/sh

echo "Enter the radius"

read r

pi = 3.14

area =  $\text{echo } \$pi \backslash * \$r \backslash * \$r \backslash bc$

echo "Area is \$area"

## OUTPUT

CASE 1 :- ENTER THE NUMBER

1

1 is positive

CASE 2 : ENTER THE NUMBER

-1

-1 is negative

CASE 3 : ENTER THE NUMBER

0

NUMBER IS 0.

# Shell script to check if a number is positive, negative or zero.

#!/bin/sh

echo "Enter the number"

read n

if [ \$n -lt 0 ]

then

echo " \$n is negative "

elif [ \$n -gt 0 ]

then

echo " \$n is positive "

else

echo " \$n is zero "

fi

OUTPUT :- ENTER NUMBER 1

25

ENTER NUMBER 2

49

ENTER NUMBER 3

15

PRINTING LARGEST

49

# Shell script to find the largest of three numbers.

#!/bin/sh

echo "Enter number 1"

read n1

echo "Enter number 2"

read n2

echo "Enter number 3"

read n3

echo "Finding largest"

if [ \$n1 -gt \$n2 -a \$n1 -gt \$n3 ]  
then

echo \$n1

elif [ \$n2 -gt \$n1 -a \$n2 -gt \$n3 ]  
then

echo \$n2

else

echo \$n3

fi



OUTPUT

ENTER THE NUMBER

5

120

# Shell script to find the factorial of a number

#!/bin/sh

echo "Enter the number"

read n

fact = 1

while [ \$n -gt 1 ]; do

fact = \$((\$fact \* n))

n = \$((\$n - 1))

done

echo \$fact.

OUTPUT :-

ENTER THE BASIC SALARY

2,000

GROSS SALARY IS 2500

# Shell script to find the gross salary of a person

#!/bin/sh

echo "Enter the basic salary"

read bs

hra = \$ `echo "scale=2; \$bs \* 20 / 100" | bc`


da = `echo "scale=2; \$bs \* 10 / 100" | bc`

gross = `echo "\$hra + \$da + \$bs" | bc`

echo "Gross salary is \$gross"

OUTPUT :- ENTER degree fahrenheit temperature  
100

100 degrees fahrenheit is equal to  
 $37.7778^{\circ}$  degrees celsius.



# Shell script to convert Fahrenheit to Celsius.  
#11bin/sh

echo "Enter degree fahrenheit temperature"  
read f

c = `echo "scale=4; (\$f-32)/1.8" | bc`

echo

echo "\$f degree fahrenheit is equal to \$c  
degree celsius"

OUTPUT

1. Add      2. Sub      3. Mul      4. Div

1.

Enter the number.

10

20

30

OUTPUT

1. Add      2. Sub      3. Mul      4. Div

2

Enter the number.

10

20

-10

OUTPUT

1. Add      2. Sub      3. Mul      4. Div

3

Enter the number.

10

20

200

OUTPUT

1. Add      2. Sub      3. Mul      4. Div

4

Enter the number.

10

20

0.5000

# Shell script to perform arithmetic operations on given two numbers.

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

```
echo "1. Add \n 2. Sub \n 3. Mul \n 4. Div"
```

```
read i
```

```
echo "Enter the two numbers"
```

```
read a
```

```
read b
```

```
case $i in
```

```
1) ans = $(($a+$b)) ;;
```

```
2) ans = $(($a-$b)) ;;
```

```
3) ans = $(($a*$b)) ;;
```

```
4) ans = `echo "scale=2; $a/$b" | bc`;
```

```
* ) echo "Enter valid input"
```

```
    exit 1
```

```
esac
```

```
echo $ans
```



OUTPUT.

ENTER AN NUMBER:-

10

Sum = 30



# Shell script to find sum of even numbers upto n

#!/bin/sh

echo "Enter <sup>an</sup> the number of elements"

read n

sum=0; i=2

~~echo~~

while [ \$i -le \$n ]

do

sum=\$((sum+i))

i=\$((i+2))

done

echo "Sum = \$sum"

# OUTPUT

1 1 1  
1 1 2  
1 1 3  
1 2 1  
1 2 2  
1 2 3  
1 3 1  
1 3 2  
1 3 3  
2 1 1  
2 1 2  
2 1 3  
2 2 1  
2 2 2  
2 2 3  
2 3 1  
2 3 2  
2 3 3  
3 1 1  
3 1 2  
3 1 3  
3 2 1  
3 2 2  
3 2 3

3 3 1  
3 3 2  
3 3 3

# Shell script to print all combinations of number 1 2 3

#!/bin/sh

for i in 1 2 3  
do

for j in 1 2 3  
do

for k in 1 2 3  
do

echo \$i \$j \$k

done

done

done

### OUTPUT

Enter the number

3

Enter the power

3

3 to the Power 3 is 27

# Shell script to find the power of a number

#!/bin/sh

echo "Enter the number"

read n

echo "Enter the power"

read p

pow = \$n

mul = 1

while [ \$p -gt 0 ]

do

mul = \$(mul \* n)

p = \$(p - 1)

done

echo " \$n to the power \$p is \$mul "

### OUTPUT

Enter the number of elements  
6

Enter the 6 numbers

30

26

84

93

29

11

Sum of the numbers is 273



# Shell script to find the sum of n natural numbers

#!/bin/sh

echo "Enter the number of elements"

read n

sum=0

echo "Enter the \$n numbers"

while [ \$n -gt 0 ]

do

read a

sum=\$((sum + a))

n=\$((n - 1))

done

echo "Sum of the numbers is \$sum"



OUTPUT :-

Enter the internal marks and see of subject 1

35

50

Grade of subject 1 is D

Enter the internal marks and see subject 2

45

70

Grade of subject 2 is B

Enter the internal marks and SEE subject 3

25

45

Grade of subject 3 is E

Enter the internal marks and SEE subject 4

49

90

Grade of subject 4 is S

Enter the internal marks and SEE subject 5

33

12

Student has failed in this subject

Enter the internal marks and see of subject 6

29

81

Grade of subject 6 is C

Pass = 5

Fail = 1

2

\* Shell script to display pass class of a student

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

```
n=1
```

```
fail=0
```

```
pass=0
```

```
while [ $n -le 6 ]
```

```
do
```

```
    echo "Enter the internal marks and see of subject $n"
```

```
    read internal
```

```
    read external
```

```
    if [ $internal -lt 20 -o $external -lt 40 ]
```

```
    then
```

```
        echo "Student has failed in this subject"
```

```
        fail=$((fail+1))
```

```
        n=$((n+1))
```

```
    continue
```

```
fi
```

```
if [ $marks -gt 90 ]
```

```
then
```

```
    echo "Grade of subject $n is S"
```

```
    pass=$((pass+1))
```

```
elif [ $marks -gt 80 ]
```

```
    echo "Grade of subject is A"
```

```
    pass=$((pass+1))
```

```
elif [ $marks -gt 70 ] ; then
```

```
    echo "Grade of subject is B"
```

```
pass = $(1 pass+1))  
elif [ $marks -gt 60 ]  
    echo "Grade of subject $n is C"  
    pass = $(1 pass+1))  
else  
    echo "Grade of subject $n is P"  
    fail = $(1 fail+1))  
fi  
n = $(1 n+1))  
done  
echo " Pass = $pass"  
echo " Fail = $fail"
```

OUTPUT

Enter the number.

5

1 1 2 3 5

\* Shell script to find the fibonacci series up to n.

```
echo "Enter the number"
```

```
read n
```

```
i=0
```

```
las=1
```

```
ser=0
```

```
while [ $i -lt $n ]
```

```
do
```

```
    echo "$las"
```

```
    temp=$ser
```

```
    ser=$las
```

```
    las=$((temp+las))
```

```
    i=$((i+1))
```

```
done
```

OUTPUT.

Enter the string

abcdefghijklmnopqrstuvwxyz

Vowels = 5

\* Shell script to count the number of vowels of a string.

```
echo "enter the string"
```

```
read str
```

```
len = `expr $str | wc -c`
```

```
leng = `expr $len - 1`
```

```
count = 0
```

```
while [ $len -gt 0 ]
```

```
do
```

```
ch = `expr $str | cut -c $len`
```

```
case $ch in
```

```
[aeiou,AEIOU]) count = `expr $count + 1`;;
```

```
esac
```

```
len = `expr $len - 1`
```

```
done
```

```
echo "vowel = $count"
```

## OUTPUT

Enter the filename

abcd.txt

lines = 1      abcd.txt

Words = 1      abcd.txt

Character = 27      abcd.txt



\* Shell script to check number of lines, words and characters in a file.

```
echo "enter the file name"
```

```
read f
```

```
l=`wc -l $f`
```

```
c=`wc -c $f`
```

```
w=`wc -w $f`
```

```
echo "Lines = $l \n Words = $w \n Characters = $c"
```

## OUTPUT

SHELL = /bin/bash  
SESSION-MANAGER = local/dashline . . . . .  
QT\_ACCESSIBILITY = 1  
COLORTERM = true color.  
:

\* Write a C/C++ program to find contents of its environment list.

```
#include <stdio.h>
```

```
int main( int argc, char * argv[] )
```

```
{
```

```
    int i;
```

```
    char ** ptr;
```

```
    extern char ** environ;
```

```
    for ( ptr = environ ; *ptr != 0 ; ptr++)
```

```
        printf( "%s", *ptr );
```

```
    return 0;
```

```
}
```

### OUTPUT 1

. /a.out test1.txt test1copy.txt  
Hard link created.

### OUTPUT 2

. /a.out -s test1.txt test1soft.txt  
Soft Symbolic link created.

\* 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] <sy-fn> <newlink> ");
        return 1;
    }

    else if ( argc == 4 ) {
        if ( symlink (argv[2], argv[3]) == -1 )
            printf (" Symbolic cannot be created ");
        else
            printf (" Sytem Symbolic link created ");
    } else {
        if ( link (argv[1], argv[2]) == -1 )
            printf (" Cannot make Hard link ");
        else
            printf (" Hard link created ");
    }

    return 0;
}
```

## OUTPUT.

System supports job control  
System supports SAVED SET-UID AND SE-610  
System supports CHOWN RESTRICTED 0  
Path name true option is 0  
Dirable character is 0

- \* Write C/C++ POSIX compliant program that prints POSIX defined configuration options supported on any given systems using feature test macros.

```
#define -POSIX_SOURCE
#define -POSIX -C-SOURCE 199309L
#include <stdio.h>
#include <unistd.h>

int main() {
    #ifdef -POSIX -JOB-CONTROL
        printf("System supports job control");
    #else
        printf("System does not support job control");
    #endif

    #ifdef -POSIX -SAVED_IDS
        printf("System supports Saved SetID and SE_UID in");
    #else
        printf("System does not support SAVED setuid
        and SE-UID ");
    #endif

    #ifdef -POSIX -CHOWN-RESTRICTED
        printf("System supports CHOWN RESTRICTED in
        %d\n", -POSIX -CHOWN-RESTRICTED);
    #else
        printf("System does not support CHOWN OPTIONS in");
    #endif
}
```

```
#if def -POSIX-NO-TRUNC
```

```
printf("Path name trunc option is %d\n",  
-POSIX-VDISABLE);
```

```
#else
```

```
printf("Path name trunc option is not supported");
```

```
#endif
```

```
#if def -POSIX-VDISABLE
```

```
printf("Disable character is %d\n", -POSIX-VDISABLE);
```

```
#else
```

```
printf("Disable character is not supported");
```

```
#endif
```



### OUTPUT 1.

./a.out    abcd.txt

ab c d e f g h i j k l m n a p q r s t u v w x y z

### OUTPUT 2.

./a.out    abcd.txt    a b c d e f u

./a.out    abcd.txt

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

```
#include <sys/types.h>
```

```
#include <unistd.h>
```

```
#include <fcntl.h>
```

```
#include <sys/stat.h>
```

```
#include <string.h>
```

```
#include <errno.h>
```

```
#include <stdio.h>
```

```
int main (int argc, char * argv[]) {
```

```
    int fd;
```

```
    char buf[256];
```

```
    if (argc != 2 || argv[1] != "w") {
```

```
        printf("USAGE: ./a.out <file> [w|r]\n",
```

```
        argv[0]);
```

```
        return 1;
```

```
    }
```

```
    mkfifo (argv[1], S_IFIFO | S_IRWXU | S_IRWXG
```

```
    | S_IRWXO);
```

```
    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 | O\_CREAT);  
write(fd, argv[2], strlen(argv[2]));

}

close(fd);

}