<u>West Bengal State University</u>

Registration No -1071711100060 of 2017

Roll -3201134 No -19162

Sub - Unix Assignment

INDEX

1. Assignment problem:

Create a basic calculator for two no.S

Algorithm:

Input: Two numbers a and b.

Output:The sum of a and b will be stored in c, difference of a and b will be stored in d, product of a and b will be stored in e, quotient of a and b will be stored in f.

```
START
STEP1:print "ENTER TWO NUMBERS :"
STEP2:read a b
STEP3:set c= a+b
STEP4:set d= a-b
STEP5:set e= a*b
STEP6:set f= a/b
STEP7:print "SUM= c "
STEP8:print "DIFFERENCE= d "
STEP9:print "PRODUCT= e "
STEP10:print "QUOTIENT= f "
STOP
```

PROGRAM CODE:

```
echo "ENTER TWO NUMBERS :"
read a b
c=`expr $a + $b`
d=`expr $a - $b`
e=`expr $a \* $b`
f=`expr $a / $b`
echo "SUM= $c "
echo "DIFFERENCE= $d "
echo "PRODUCT= $e "
echo "QUOTIENT= $f "
```

OUTPUT:

```
ENTER TWO NUMBERS :8 2
SUM= 10
DIFFERENCE= 6
PRODUCT= 16
QUOTIENT= 4
```

DISCUSSION:

The program code will execute properly only for integer values,for floating point numbers we have to introduce ‘bc’ instruction.if the quotient is not an integer then the program code will not give the exact result,there are other codes for that problem.

Signature-

2. Assignment problem:

Write a program to solve this equation $\frac{1}{a}+\frac{1}{b}$

Algorithm:

Input: Two numbers a and b

Output: the inverse of a and b will be added and stored in s

```
START
STEP1:print "ENTER TWO NUMBERS :"
STEP2:read a b
STEP3:set c= 1
STEP4:set m= 1/a
STEP5:set n= 1/b
STEP6:set s= m+n
STEP7:print "SUM= s "
STOP
```

Program code:

```
echo "enter two number:"
read a b
c=1
m=`echo "scale=2 ; $c / $a " | bc`
n=`echo "scale=2 ; $c / $b " | bc`
s =`echo $m + $n | bc`
echo "sum= $s"
```

Output:

```
enter two number:
2 4
.75
[computer@localhost ~]$
```

DISCUSSION:

The program code will execute properly as we use here the ‘bc’ instruction. It gives correct answers for integer values and for floating point numbers also.

Signature-

3. Assignment name:
Write a program to check the ODD-EVEN no.s

Algorithm:

Input: one number a.
Output: If the number is even output is “EVEN NUMBER, otherwise output is “ODD NUMBER”.

```
START
print "ENTER A NUMBER"
read a
set r= a%2
if (r=0)
 then
print "EVEN NUMBER"
else
print "ODD NUMBER"
end if
STOP
```

PROGRAM CODE:

```
echo "ENTER A NUMBER :"
read a
r=`expr $a % 2`
if [ $r -eq 0 ]
then
echo "EVEN NUMBER"
else
echo "ODD NUMBER"
fi
```

OUTPUT:

```
ENTER A NUMBER :3
ODD NUMBER
ENTER A NUMBER :8
EVEN NUMBER
```

DISCUSSION:

Like basic C,C++ we can use some comparison operators in UNIX,but they are different from c and c++.In UNIX to check if a number equals another number we use ‘-eq’ operator.We also use –lt,-le,-gt,ge to check less than,less than or quals,greater than,greater than or equals.for if-else condition we always end the loop with ‘fi’.

Signature-

4. Assignment name:
Find the largest no from three no.s

Algorithm:

Input: Three numbers a, b, c

Output: If a is greater than b and c then a is the largest no, or if b is greater than a and c then b is the largest no otherwise c is the largest no.

```
START
Print “enter three no.s”
Read a b c
If(a>b)
Then
        If(a>c)
        Then
        Print “ a is the largest no.”
        Else
        Print ”c is the largest no.”
        End if
Else
        If(b>c)
        Then
        Print” b is the largest no.”
        Else
        Print” c is the largest no.”
        End if
End if
STOP
```

Program code:

```
echo "enter three no."
read a, b, c
if [ $a -gt $b ]
then
          if [ $a -gt $c ]
          then
          echo "$a is the largest no "
          else
          echo "$c is the largest no "
          fi
else
          if [ $b -gt $c ]
          then
          echo "$b is the largest no"
          else
          echo "$c is the largest no"
          fi
fi
```

Output:

```
enter three no.
10 11 12
12 is the largest no
[computer@localhost ~]$
```

DISCUSSION:
1.Like basic C,C++ we can use some comparison operators in UNIX,but they are different from C and C++.
2.In UNIX to check if a number equals another number we use ‘-eq’ operator.
3.We also use –lt,-le,-gt,-ge to check less than,less than or equals,greater than,greater than equals.
4.For if-else condition we always end the loop with ‘fi’.
5.For floating point numbers we have to introduce ‘bc’ instruction.If the quotient is not an integer then the program code will not give the exact result,there are other codes for that problem.

Signature-

5. Assignment name:
Write the program to check the year is leap year or not.

Algorithm:

Input: Enter the year n.

Output: If the year is leap year then print “leap year”, otherwise print “not leap year”.

```
START
Print "ENTER A YEAR"
read n
set i= n%4
set j= n%100
set k= n%400
        if (i=0)
        then
                if (j=0)
                then
                        if (k=0)
                        then
                        print "leap year"
                        else
                        print "not leap year"
                        End if
                else
                print "leap year"
                End if
        else
        print "not leap year"
        End if
STOP
```

Program Code:

```
echo "enter a year"
read n
i=`expr $n % 4`
j=`expr $n % 100`
k=`expr $n % 400`
if [ $i -eq 0 ]
then
        if [ $j -eq 0 ]
        then
                if [ $k -eq 0 ]
                then
                echo " leap year"
                else
                echo " not leap year"
                fi
        else
        echo " leap year"
        fi
else
echo " not leap year"
fi
```

Output:

```
enter a year
2000
 leap year
[computer@localhost ~]$
```

DISCUSSION:

1.Like basic C,C++ we can use some comparison operators in UNIX,but they are different from C and C++.
2.In UNIX to check if a number equals another number we use ‘-eq’ operator.
3.We also use –lt,-le,-gt,-ge to check less than,less than or equals,greater than,greater than equals.
4.For if-else condition we always end the loop with ‘fi’.
5.For floating point numbers we have to introduce ‘bc’ instruction.If the quotient is not an integer then the program code will not give the exact result,there are other codes for that problem.

Signature-

6. Assignment name:
Enter a file name and count the total number of words, lines and characters.

Algorithm:

```
Input: A text file named animesh.txt
Output: Count the total no.of word, line and character.
START
Print “enter the file”
Read f
Use code WC $f
STOP
```

Program code:

```
echo "enter a file name "
read f
wc $f
```

Input:

```
[computer@localhost ~]$ cat>animesh.txt
i am a student
bhairab ganguly college
computer science hons
```

Output:

```
 enter a file name
animesh.txt
 3 10 61 animesh.txt
```

Discussion:

All the spaces are converted into new lines,i.e. all the words are printed line by line using ‘tr’ command. Using ‘grep’ the wanted word is searched and other words get removed.The same word can be in upper case or in lower case in different places of the file.But this difference is completely ignored and only the particular word is counted.

Signature-

7. Assignment name:
Find the current date and time.

Algorithm:

```
START
Print “the current date”
Find the current date using the command
Print “the current time”
Find the current time using the command
STOP
```

Program code:

```
echo "the current date is"
date | cut -d " " -f 1,2,4,7
echo "the current time is"
date | cut -d " " -f 5
```

Output:

```
the current date is
Tue Oct 1 2019
the current time is
14:50:02
[computer@localhost ~]$
```

Discussion:

It display the current date and time.

Signature-

8. Assignment name:
Write a program to check Prime numbers.

Algorithm:

INPUT: Input a number n.

OUTPUT: If n is prime number print “PRIME NUMBER”, otherwise print “NOT PRIME NUMBER”.

```
START
print "ENTER A NUMBER"
read n
set i= 2
set f=1
while(i<n)
          do
          set i=n%i
          if(i=0)
          then
          set f=0
          end if
                  set i=i+1
                        done
                  if(f=0)
                  then
                       print “NOT PRIME NUMBER”
                       else
                       print “PRIME NUMBER”
                    end if
STOP
```

Program code:

```
echo "enter a no."
read n
i=2
f=1
while [ $i -lt $n ]
do
I=`expr $n % $i`
if [ $I -eq 0 ]
then
f=0
fi
i=`expr $i + 1`
done
if [ $f -eq 0 ]
then
echo "not prime"
else
echo "prime no."
Fi
```

Output:

```
enter a no.
```

```
3
prime no.
[computer@localhost ~]$
```

Discussion:

1.Like basic C,C++ we can use some comparison operators in UNIX,but they are different from C and C++.
2.In UNIX to check if a number equals another number we use ‘-eq’ operator.
3.We also use –lt,-le,-gt,-ge to check less than,less than or equals,greater than,greater than equals.
4.For if-else condition we always end the loop with ‘fi’.
5.For floating point numbers we have to introduce ‘bc’ instruction.If the quotient is not an integer then the program code will not give the exact result,there are other codes for that problem.

Signature-

9. Assignment name:
Write a program to print “Good Morning/ Good Afternoon/ Good Evening/ Good Night” as per the current time.

Algorithm:

```
START
Print “the current time”
Find the current time using the command and store in t
Print
        If (t<12)
        Then
        Print “Good Morning”
        Else
                If (t<17)
                Then
                Print”Good Afternoon”
                Else
                        If (t<20)
                        Then
                        Print”Good Evening”
                        Else
                                Print”Good Night”
                        End If
                End If
        End If
STOP
```

Program Code:

```
echo "the current time is"
t=`date | cut -d " " -f 5 | cut -b 1,2`
echo "$t"
if [ $t -le 12 ]
then
        echo "Good Morning"
else
        if [ $t -le 17 ]
        then
                echo "Good afternoon"
        else
                if [ $t -lt 20 ]
                then
                        echo "Good Evening"
                else
                        echo "Good Night"
                fi
        fi
fi
```

Output:

```
the current time is
16
Good afternoon
```

```
[computer@localhost ~]$
```

Discussion:
1.Like basic C,C++ we can use some comparison operators in UNIX,but they are different from C and C++.
2.In UNIX to check if a number equals another number we use ‘-eq’ operator.
3.We also use –lt,-le,-gt,-ge to check less than,less than or equals,greater than,greater than equals.
4.For if-else condition we always end the loop with ‘fi’.
5.For floating point numbers we have to introduce ‘bc’ instruction.If the quotient is not an integer then the program code will not give the exact result,there are other codes for that problem.

Signature-

10. Assignment Name:
Write a program to print the Fibonacci Series.

Algorithm:

INPUT: Input a number n as a range.
OUTPUT: Fibonacci series upto the limit.

```
START
print"ENTER THE LIMIT"
read l
set n1=0
set n2=1
print n1
print n2
set i=2
while (i<=l)
        do
set n3=n1+n2
print n3
set n1=n2
set n2=n3
set i=i+1
        done
STOP
```

Program code:

```
echo "Enter the limit"
read l
n1=0
n2=1
echo "$n1"
echo "$n2"
i=2
while [ $i -lt $l ]
do
n3=`expr $n1 + $n2`
echo "$n3"
n1=$n2
n2=$n3
i=`expr $i + 1`
done
```

Output:

```
Enter the limit
10
0
1
1
2
3
5
8
13
21
```

```
34
[computer@localhost ~]$
```

Discussion:

1.Like basic C,C++ we can use some comparison operators in UNIX,but they are different from C and C++.
2.In UNIX to check if a number equals another number we use ‘-eq’ operator.
3.We also use –lt,-le,-gt,-ge to check less than,less than or equals,greater than,greater than equals.
4.For if-else condition we always end the loop with ‘fi’.
5.For floating point numbers we have to introduce ‘bc’ instruction.If the quotient is not an integer then the program code will not give the exact result,there are other codes for that problem.

Signature-

11. Assignment name:
Find the Factorial of the number in input.

Algorithm:

```
INPUT: Input a number n.
OUTPUT: Factorial of n.
START
print"ENTER A NUMBER"
read n
set f=1
set i=1
while (i<n)
        do
        set f=f*i
        set i=i+1
        done
print f
end
STOP
```

Program code:

```
echo "ENTER A NUMBER: "
read n
f=1
i=1
while [ $i -le $n ]
do
f=`expr $f \* $i`
i=`expr $i + 1`
done
```

Output:

```
echo "Factorial is $f"
ENTER A NUMBER: 10
Factorial is 3628800
```

Discussion:

In UNIX to check if a number equals another number we use ‘-eq’ operator.We also use –lt,-le,-gt,-ge to check less than,less than or equals,greater than,greater than equals.For if-else condition we always end the loop with ‘fi’.For floating point numbers we have to introduce ‘bc’ instruction.If the quotient is not an integer then the program code will not give the exact result,there are other codes for that problem

Signature-

12. Assignment name:
Default reader and writer permission of file ,directory,and then change the read write format default of file.

Algorithm:

```
INPUT: Input a number n.
OUTPUT: change the read write format of the given file.
START
print"enter the file name"
read n
Use code ls -1 $n
Use code chmod 777 $n
Use code ls -1 $
STOP
```

Program code:

```
echo "enter the file name"
read n
ls -l $n
chmod 777 $n
ls -l $
```

Output:

```
[computer@localhost ~]$ sh 2.sh
enter the file name
animesh.txt
-rwxrwxrwx. 1 computer computer 61 Oct  1 14:02 animesh.txt
-rwxrwxrwx. 1 computer computer 61 Oct  1 14:02 animesh.txt
[computer@localhost ~]$ ^C
[computer@localhost ~]$
```

Discussion:

Most file systems have methods to assign permission or access rights to specific users and groups of users.this permissions control the abilityof the users to view,change, navigate,and execute the contents of the file system.

Signature-

13. Assignment name:
Input a file and then insert a blank line after every line.

Algorithm:

```
INPUT: Input the file name
OUTPUT:insert a blank line after every line.
START
print"enter the file name"
read f
print"content of file "
Use code cat $f
While read lines
do
print al.txt
done
print"after inserting blank line"
use code cat al.txt
STOP
```

Program code:

```
echo "enter the file name"
read f
echo "content of file "
cat $f
cat $f | while read line
      do
      echo $line>>a1.txt
      echo >>a1.txt
      done
echo "after inserting blank line"
cat a1.txt
```

Output:

```
[computer@localhost ~]$ sh 2.sh
enter the file name
animesh.txt
content of file
i am a student
bhairab ganguly college
computer science hons
after inserting blank line
i am a student

bhairab ganguly college

computer science hons

[computer@localhost ~]$
```

Discussion:
In this program we discussed how to insert a blank line after every line.the file content are read in a while loop.

Signature-

14. Assignment name:
Sum and average of all arguments.

Algorithm:

INPUT: Input the arguments
OUTPUT:sum and average of all arguments.

```
START
Sum=0
c=$#
for  i in $*
sum=sum+i
done
print Sum
avg=sum/c`
print Avg
STOP
```

Program code:

```
sum=0
c=$#
for i in $*
do
sum=`expr $sum + $i `
done
echo "Sum=$sum"
avg=`expr $sum / $c`
echo "Avg=$avg"
```

Output:

```
[computer@localhost ~]$ sh 2.sh 2 4 6
Sum=12
Avg=4
[computer@localhost ~]$
```

Discussion:

SUM:This function adds all the values of the cells in the argument.
AVG:this function determines the average of the values includedin the argument. It calculate the sum of the cells and then divides that value by the number of cellsin the argument.

Signature-

15. Assignment name:

Input a file from command argument and print the number of line,characters and words.

Algorithm:
INPUT: Input the file from arguments
OUTPUT: and print the number of line,characters and words.

```
START
Use code  l= wc  -l  $l
Use code  w=wc  -w  $l
Use code  c=wc  -c $l
Print  “total line=”
Print  “total word=”
Print “total character=”
STOP
```

Program code:
```
l=`wc -l $1`
w=`wc -w $1`
c=`wc -c $1`
echo "Total line = $l "
echo "Total word = $w "
echo "Total charecter = $c "
```

Output:
```
[computer@localhost ~]$ sh 2.sh Dip.txt
Total line = 599 animesh.txt
Total word = 1673 animesh.txt
Total charecter = 8844 animesh.txt
[computer@localhost ~]$
```

Discussion:
Counting the number of character is important because almost all the text boxes that rely on user input have certain limit on the number of charactersthat can be inserted.

Signature-