
-----   V SEM Info.Science UNIX LAB PROGRAMS

1a) Write a non recursive shell script which accepts any number of arguments and
prints them in the reverse order.


```
echo "no of arguments are: $#"
len=$#

while [ $len -ne 0 ]
 do
   eval echo \$$len
len=`expr $len - 1`
done
---------------
   out put
---------------
$ sh lab1a.sh shiva kumar arun
no of arguments are: 3
arun
kumar
shiva
```


1b) Write a C program that creates a child process to read commands from the
standard input and execute them. You can assume that no arguments will be
passed to the commands to be executed.


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

```
#include<stdlib.h>
main()
{
        char cmd[10];
        int i;
        if(fork() == 0)
        do
        {
                printf("Enter the Command\n");
                scanf("%s",&cmd);
                system(cmd);
                printf("Enter 1 to continue and 0 to stop: ");
                scanf("%d",&i);
        }while(i != 0);
        wait();
}
```

----
  out put

```
$ cc lab1b.c
$ ./a.out
Enter the Command
ls
a.out    lab11b.c lab14a.pl   lab1b.c   lab4a.sh       lab6b.c
lab9a.sh
index    lab12a.awk    lab14b.c    lab2a.sh  lab4b.c lab7a.sh
lab9b.pl
lab10a.awk  lab12b.pl lab15a.sh   lab2b.c   lab5a.sh
        lab7b.awk  typescript
lab10.c     lab13a.pl    lab15b.awk  lab3a.sh  lab5b.c      lab8a.sh
lab11a.htm  lab13b.awk       lab1a.sh    lab3b.c   lab6a.sh
        lab8b.pl
Enter 1 to continue and 0 to stop: 1
Enter the Command
who
shiva    tty4        Dec 15 18:20
Enter 1 to continue and 0 to stop: 0
```


2a) Write a shell script that accepts two file names as command line arguments,
checks if the permissions for these files are identical and outputs common
permissions, and otherwise outputs each filename and otherwise outputs each file
name followed by its permissions.


```
ls -l $1 | cut -d " " -f 1 > file1
ls -l $2 | cut -d " " -f 1 > file2

 if  cmp file1 file2
 then
 echo "Both the permission are same"
 cat file1
 else
 echo "The permission are different"
 echo "The permission for $1 are"
 cat file1
 echo "The permission for $2 are"
 cat file2
fi
```

----
   out put

```
$ cat > file1
Welcome to the word of UNIX
$ cat > file2
Welcome to the word of LINUX
$ sh lab2a.sh file1 file2
Both the permission are same
-rw-rw-r--
$ chmod +x file1
$ sh lab2a.sh file1 file2
```

```
file1 file2 differ: byte 4, line 1
The permission are different
The permission for file1 are
-rwxrwxr-x
The permission for file2 are
-rw-rw-r--
```


2b) Write a C program to create a file with 16 bytes of arbitrary data from the
beginning and other 16 bytes of aribitrary data from an offset of 48. display
the file contents to demonstrate how the hole in the file is handled.


```
#include<stdio.h>
#include<sys/types.h>
#include<unistd.h>
#include<fcntl.h>
main()
{
        char *fname="test.dat",*cmd="od -bc test.dat";
        char
*str1="ABCDEFGHIJKLMNOP",*str2="abcdefghijklmnop";
        int fd;

        if(fd = creat(fname,0755) == -1)
        {
                printf("Cannot Create file %s",fname);
                exit(1);
        }
       fd = open(fname,O_RDWR);
        if(write(fd,str1,16) != 16)
        {
                printf("Error in writing\n");
                exit(1);
        }

        lseek(fd,48,0);

        if(write(fd,str2,16)!=16)
        {
                printf("Error in Writing\n");
                exit(1);
        }
        system(cmd);
}
-------------
  out put
-------------
$ cc lab2b.c
$ ./a.out
0000000 101 102 103 104 105 106 107 110 111 112 113 114
115 116 117 120
         A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P
0000020 000 000 000 000 000 000 000 000 000 000 000 000
000 000 000 000
        \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0  \0
*
0000060 141 142 143 144 145 146 147 150 151 152 153 154
155 156 157 160
         a   b   c   d   e   f   g   h   i   j   k   l   m   n   o   p
0000100
```


3a) Write a shell script that takes a valid directory name as a command line
argument and recursively descends all the sub directories, finds the maximum
length of any file in that hierarchy and writes this maximum value to the
standard output.


```
max=`ls -lR $1|grep '^-'|cut -c 35-42|sort -n|tail -1`
echo "The maximum size of the file in $1 is $max"
```

----
 out put

```
$ sh lab3a.sh /home/shiva
The maximum size of the file in /home/shiva is  3532800
```


3b) Write a C program which accepts valid file name as command line arguments
and for each of the arguments, prints the type of the file.


```
#include<stdio.h>
#include<sys/types.h>
#include<sys/stat.h>
 main(int argc, char *argv[])
{
        int i;
        struct stat buf;
        for(i=1;i<argc;i++)
        {
                printf("%s\t",argv[i]);
                if(stat(argv[i],&buf)==-1)
                {
                        printf("Stat Error");
                        continue;
                }
                if(S_ISREG(buf.st_mode))
                        printf("Regular File\n");
                else if(S_ISDIR(buf.st_mode))
                        printf("Directory File\n");
                else if(S_ISCHR(buf.st_mode))
                        printf("Char Device File\n");
                else if(S_ISBLK(buf.st_mode))
                        printf("Block Device File\n");
                else if(S_ISLNK(buf.st_mode))
                        printf("Symbolic Link File\n");
        }
}
-------------
  out put
------------
$ cc lab3b.c
$ ./a.out /dev/hda /etc/passwd
/dev/hda    Block Device File
/etc/passwd        Regular File
```


4a) Write a shell script that accepts path names and creates all the components
in the path name as directories.


```
a=$#
if [  $a -le 0 -o $a -gt 1 ]
then
 echo "No proper arguments"
else
 mkdir -p $1
fi
-----------
  out put
-----------
$ sh lab4a.sh a/b/c
use ls command to see a tree structure a/b/c.
```


4b) Write a C program which accepts one command line argument, executes the

arguments as a shell command, determines the time taken by it and prints the
timing values. Use the 'times' function and the 'tms' structure. The code need
not include error checking.

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/types.h>
#include<unistd.h>
#include<sys/times.h>
struct tms tms1,tms2;
main(int argc,char *argv[])
{
        int i;
        clock_t start,finish;
        long clkteck = sysconf(_SC_CLK_TCK);
        start = times(&tms1);
        for(i=1;i<argc;i++)
        system(argv[i]);
        finish = times(&tms2);
        printf("Time Taken : %7.2f",(finish - start) /
(double)clktck);
}
```
----
 out put

```
$ cc lab4b.c
$ ./a.out "ls | wc" "who"
    38      38     335
shiva    :0          Dec 15 18:41
shiva    pts/0        Dec 15 18:42
shiva    pts/1        Dec 15 18:42
Time Taken :    1.48
```

5a) Write a shell script which accepts valid names as command line arguments
and prints their corresponding home directories. If no arguments are
specified, print a suitable error message.

```
---------------------------------------------------------------------
--
a="$#"
if [ $a -eq 0 ]
then
echo "error - no argument"
exit 1
fi

while [ "$#" -gt 0 ]
 do
str=" "
str=`cat /etc/passwd | grep "^$1"`
if [ -n "$str" ]
then
 echo -e  "the home dir of user  $1 is \c"
 echo `echo $str | cut -d ":" -f 6`
 echo " "
else
echo "$1 is not a valid loginname"
fi
shift
done
------------
  out put
------------
$ sh lab5a.sh shiva kumar anju
the home dir of user  shiva is /home/shiva
the home dir of user  kumar is /home/kumar
anju is not a valid loginname
---------------------------------------------------------------------
--
```

5b) Write a C program which accepts a valid directory names as arguments and
lists all the files in the given directory as well as the files in the subsequent directories.

--

```
#include<stdio.h>
#include<string.h>
main(int argc, char *argv[])
{
	int i;
	char str[50];
	for(i=1;i<argc;i++)
	{
		strcpy(str,"ls -R ");
		strcat(str,argv[i]);
		system(str);
	}
}
```

----
  out put

```
$ cc lab5b.c
$ ./a.out /home/shiva/lex /home/shiva/yacc
/home/shiva/lex:
a1.l  a3.l   file     file2.l  file5.l      file8.l  lex.yy.c
a2.l  a.out  file1.l  file4.l  file7.l    id       out
/home/shiva/yacc:
a.out  file1.y      file2.y  file3.y  y.tab.c
```


6a) Write a shell script to implement terminal locking. It should prompt the user
for a passwd after accepting the passwd entered by the user it must prompt again
for the matching passwd as configuration. The script must be written to

disregard break, Ctrl-D etc.

```
--------------------------------------------------------------------------
----
echo "Enter Password"
stty -echo
read password
stty echo
echo "Re enter the Password"
stty -echo
read password1
stty echo

if [ "$password" != "$password1" ]
then
echo "password mismatched"
else
 clear
 echo "Terminal Locked"
 echo "To unlock enter the matching password"
password1=" "
stty -echo

until [ "$password" = "$password1" ]
do
trap " " 1 2 15
read password1
done
fi
stty echo
stty sane
------------
  out put
------------
$ sh lab6a.sh
Enter Password
Re enter the Password
Terminal Locked
```

```
To unlock enter the matching password
$
```


6b) Write a C program to prompt the user for the name of an environment
variables and print its values if it is defined and a suitable message otherwise
and to repeat the process if the user wants it.


```
#include<stdio.h>
main()
{
        char *p, *getenv(), name[10];
        int ch = 1;
        while(ch)
        {
                printf("\nEnter the environment variable\n");
                scanf("%s",name);
                p = getenv(name);
                if(p != NULL)
                        printf("Value of %s is %s\n",name,p);
                else
                        printf("%s is not defined\n",name);
                printf("Do you want to continue (0/1)\n");
                scanf("%d",&ch);
        }
}
```
----
 out put

```
$ cc lab6b.c
$ ./a.out
Enter the environment variable
HOME
Value of HOME is /home/shiva
```

```
Do you want to continue (0/1)
1
Enter the environment variable
PWD
Value of PWD is /home/shiva/unixlab
Do you want to continue (0/1)
1
Enter the environment variable
rm
rm is not defined
Do you want to continue (0/1)
0
```

--

7a) Write a shell program that accepts filenames specified as a arguments and
creates a shell scripts that contains these files as well as the code to
recreate these files. Thus if the script generated by your script is executed it
would recreate original file.


```
for i
do
 echo "echo $i > $i"
 echo "$i" > $i
 echo "cat $i"
 echo "echo end of $i"
done
-----------
 out put
-----------
$ sh lab7a.sh file1 file2 > file3
$ cat file1
file1
$ cat file2
```

```
file2
$ sh file3
file1
end of file1
file2
end of file2
```

--

7b) Write a awk script to delete duplicated lines from a textfile. The order of
the original lines must remain unchanged.

--

```
{
        found=0
        for(i=0;i<nlines;i++)
        if(lines[i] == $0)
        {
                found=1
                break
        }
        if(!found)
        {
                lines[nlines++]=$0
                print $0
        }
}
```

----
  out put

```
$ cat > a.txt
aaa
bbb
bbb
ccc
aaa
ddd
```

```
ddd
$ awk -f lab7b.awk a.txt
aaa
bbb
ccc
ddd
```


8a) Write a shell script that finds and displays all the links to a file
specified as the first argument to the script. The second argument which is
optional can be used to specify the directory in which the search is to begin.
If this argument is present the search is to begin in the current working
directory. In either case the starting directory as well as all its
subdirectories at all levels must be searched. The script need not include any
error checking.


```
file="$1"
a=$#
if [ $a -eq 1 ]
then
  direc="."
else
  direc="$2"
fi
lincnt=`ls -l $1 | cut -c 11-15`
inode=`ls -i $1 | cut -c 1-7`
if [ $lincnt -eq 1 ]
then
   echo "No other links to $file"
   exit 0
fi
```

```
echo "The Following files are linked together"
find "$direc" -inum $inode
-------------
  out put
-------------
$ cat > text
This file is unlinked
$ sh lab8a.sh text
No other links to text
$ ln text newtext
$ sh lab8a.sh text
The Following files are linked together
./text
./newtext
```

-

8b) Write a perl script which echoes its command line arguments , one per line
after translating all lower case letters to uppercase.

-

```
foreach $string (@ARGV) {
$string=~tr/a-z/A-Z/;
print("$string\n");
}
---------------
out put
---------------
$ perl lab8b.pl shiva kumar
SHIVA
KUMAR
```


9a) Write a shell script to display the calendar for the current month with
current date replaced by * or ** depending on either the date as 1 or 2 digits.


```
x=`date | cut -c 9-10`
cal > fname
if [ $x -le 9 ]
then
 sed "s/$x/\*/" fname
else
 sed "s/$x/\*\*/" fname
fi
```
----
  out put

```
$ sh lab9a.sh
   December 2002
Su Mo Tu We Th Fr Sa
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
** 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

--
9b) Write a perl script to convert an unsigned binary number to decimal. If an
argument is present, it can be assumed to be a valid binary number and if no
arguments is preaent, the program should display an error message.

--
```
if(@ARGV==0) {
print "Invalid Argument\n"
} else {
 $number=$ARGV[0];
 $number=~ s/(.)/$1 /g;
 @arr=split(/\s+/,$number);
 $multiplier=1;
 foreach $bit (reverse @arr) {
 $tot = $tot + $bit * $multiplier;
 $multiplier *= 2;
}
printf "The Decimal number is $tot\n";
}
```

----
  out put

```
$ perl lab9b.pl 1001
The Decimal number is 9
```


10a) Write a awk script that folds long lines into 40 colums.
Thus any line
that execeeds 40 characters must be broken after 40th character.
A x is to be
appended as the indication of folding and the processing is to be
continued with
the residue. The input is to be supplied through a text file
created by the
user.


```
{
if(length($0) <= 40)
printf "%s\n",$0
else
  {
    str=$0
    while(length(str) > 41)
       {
          printf "%sx\n",substr(str,1,40)
         str=substr(str,41,length(str)-41)
```

```
        }
     printf "%s\n",str
  }
}
```

out put

```
$ cat a.txt
Visit http://engginfo2002.tripod.com for Downloads, Lab
Programs for all
the semester (IS & CS Branch Only).

Syllabus Book is also available for IS, CS, ME & EC Branches
(I SEM to
VIII SEM).

                       Visit Today !!!
$ awk -f lab10a.awk a.txt
Visit http://engginfo2002.tripod.com forx
 Downloads, Lab Programs for all the semx
ester (IS & CS Branch Only

Syllabus Book is also available for IS, x
CS, ME & EC Branches (I SEM to VIII SEM)

                       Visit Today !!!
```


10b) Write a C program to do the following:
        Using fork(), create a child process. The child process prints its own
id and the id of its parent and then exit. the parent process waits for
the child to finish and then prints its own id and then exits.


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

```
#include<sys/types.h>
#include<stdlib.h>
main()
{
        pid_t pid,ppid,mpid;
        pid = fork();
        if(pid == 0)
        {
                ppid = getppid();
                printf("I am child, my parent pid is %d\n",ppid);
                mpid = getpid();
                printf("\nMy pid is %d\n",mpid);
                exit(0);
        }
        wait();
        mpid = getpid();
        printf("\nI am parent, mpid is %d\n",mpid);
        printf("\nMy child pid is %d\n",pid);
}
```

```
------------
  out put
-------------
$ cc lab10b.c
$ ./a.out
I am child, my parent pid is 8333
My pid is 8334
I am parent, mpid is 8333
My child pid is 8334
--------------------------------------------------------------------------
---
```

11b) Write a C program that accepts a file descriptor as the single command
argument and then prints a description of the file flags
for that descriptor. Use the fcntl() function. The program should check for
invalid number of arguments and error return from fcntl()
function.

```
-----------------------------------------------------------------------
---
#include<stdio.h>
#include<fcntl.h>
main(int argc, char *argv[])
{
        int accmode,val;
        if(arcg == 0)
        {
                printf("Invalid number of arguments");
        }
        else
        {
                val=fcntl(atoi(argv[1]),F_GETFL,0);
                accmode=val & O_ACCMODE;
                if(accmode == O_RDONLY)
                        printf("\nRead Only File\n");
                if(accmode == O_WRONLY)
                        printf("\nWrite Only File\n");
                if(accmode == O_RDWR)
                        printf("\nRead & Write only File\n");
                if(val & O_APPEND)
                        printf("\nAppend File\n");
        }
}
--------------
  out put
--------------
$ cc lab11b.c
$ ./a.out 0 < "/dev/tty"

Read Only File
$ ./a.out 1 < "/dev/tty"

Read & Write only File
$ ./a.out 2 < "/dev/tty"
```

Read & Write only File
$ ./a.out 3 < "/dev/tty"

Append File

-
12a) Write a awk script that accepts date argument in the form of mm-dd-yy and
displays it in the form of day , month ,year. The script check the validity of
the argument and in the case of error, display a suitable message.

-
```
BEGIN {
        correct=0
        printf "\nEnter the date (mm-dd-yy)\n"
        getline x < "/dev/tty"
        split(x,fields,"-")
        mon[1]="Jan";mon[2]="Feb";mon[3]="Mar";mon[4]="Ap
r";mon[5]="May"

        mon[6]="June";mon[7]="July";mon[8]="Aug";mon[9]="S
ep";mon[10]="Oct"
        mon[11]="Nov";mon[12]="Dec"

        day[1]=31;day[2]=28;day[3]=31;day[4]=30;day[5]=31;day
[6]=30
        day[7]=31;day[8]=31;day[9]=30;day[10]=31;day[11]=30;
day[12]=31

        if(fields[1] > 0 && fields[1] < 13)
        {
                if(fields[2] <= day[fields[1]])
                correct=1
                else if(fields[1]==2 && fields[3]%4==0 &&
fields[2] == 29)
```

```
            correct=1
            else
            correct=0
        }
            if(correct==1)
            printf
"\n%d,%s,%d",fields[2],mon[fields[1]],2000+fields[3]
            else
            printf "Not a valid date\a"

}
```

```
----------------
   out put
----------------
$ awk -f lab12a.awk
Enter the date (mm-dd-yy)
10-19-02

19,Oct,2002
```

--

12b) Write a perl program reads two lines of words, stores the words from
each line in separate hash, and prints as sorted list of words, with each word
followed by the message "only in line 1" , or "only in line2" or both in line1
and line2" as appropriate.

--

```
printf("Enter words for the first line:\n");
chop($line1=<STDIN>);
printf("Enter words for the second line\n");
chop($line2=<STDIN>);
@arr1=sort(split(/\s+/,$line1));
@arr2=sort(split(/\s+/,$line2));
$len1=@arr1;
```

```
$len2=@arr2;

while($i<$len1 && $j<$len2)
{
 $x=($arr1[$i] cmp $arr2[$j]);
 if($x==0)
  {
    printf("%s is in both lines\n",$arr1[$i]);
    $i++;
    $j++;
  }
 if($x==-1)
  {
   printf("%s is only in line1\n",$arr1[$i]); $i++;
  }
 if($x>=1)
 {
   printf("%s is only in line2\n",$arr2[$j]); $j++;
 }
}
while($i<$len1)
{
  printf("%s is only in line1\n",$arr1[$i]); $i++;
}
while($j<$len2)
{
  printf("%s is only in line2\n",$arr2[$j]); $j++;
}
```

----
   out put

```
$ perl lab12b.pl
Enter words for the first line:
Welcome to the word of UNIX
Enter words for the second line
Welcome to LINUX
```

```
 LINUX is only in line2
 UNIX is only in line1
 Welcome is in both lines
 of is only in line1
 the is only in line1
 to is in both lines
 word is only in line1
```


13a) Write a perl program to recognize palindromes. The program must handle one
letter words and be permissive with white space and punctuation.


```
printf("Enter the string:\t");
chop($line=<STDIN>);
$line=~tr/A-Z/a-z/;
$line=~tr/a-z/ /cd;
$rline=reverse($line);

if($line eq $rline)
{
 printf("\n The string is palindrom\n");
}
else
{
 printf("\nThe string is not palindrom\n");

}
```
----
  out put

```
$ perl lab13a.pl
Enter the string: abcba

The string is palindrom
```

```
$ perl lab13a.pl
Enter the string: UNIX

The string is not palindrom
```

-

13b) Write a awk script to read the /etc/passwd file and list users who have
duplicate user ID numbers.

-

```
BEGIN {FS=":"}
{
 found = 0
 for(i=0;i<nlines;i++)
 if(fields[i] == $3)
  {
   found = 1
   break
  }
 if(!found)
  fields[nlines++] = $3
 else
  printf "%s has duplicate uid\n", $1
}
```

----
  out put

```
$ cat /etc/passwd (Before Editing)
root:x:0:0:root,,,6604878:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
shiva:x:500:500:manjunath:/home/shiva:/bin/bash
manjunath:x:501:501::/home/manjunath:/bin/bash
kumar:x:502:502::/home/kumar:/bin/bash

# login as root and manually edit /etc/passwd such that
# 3rd and 4th field should be indentical (delimeted by :)
# 1st field represents username
# 3rd & 4th fields represents user id and group id respectevely

$cat /etc/passwd (After Editing)
root:x:0:0:root,,,6604878:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
shiva:x:500:500:manjunath:/home/shiva:/bin/bash
manjunath:x:501:501::/home/manjunath:/bin/bash
kumar:x:502:502::/home/kumar:/bin/bash
arun:x:502:502::/home/arun:/bin/bash

# in our example arun is the user name and user id is 502 which
is identical
# to the user kumar (see last two lines above)

awk -f lab13b.awk /etc/passwd
arun has duplicate uid
```


14a) Write a PERL program to determine word frequencies using associative
arrays, in a given paragraph of text. You can assume that the paragraph begins
with a word , and that there or no hyphenated words. Words capitalized
differently are treated as same words.


```
@entire_file=<>;
chop(@entire_file);
foreach $line(@entire_file)
 {
   @arr=split( /\s+/,$line);
   foreach $word(@arr)
    {
     $word=~tr/a-z/A-Z/;
     $count{$word} += 1;
    }
 }
foreach $word(sort(keys(%count)))
 {
  printf("$word:$count{$word}\n");
}
---------------
   out put
---------------
$ cat > text
Welcome to the Word of RedHat Linux 8.0
$ perl lab14a.pl text
8.0:1
LINUX:1
OF:1
REDHAT:1
THE:1
TO:1
WELCOME:1
WORD:1
```

--

14b) Write a C program to print device number for each command line
argument. Additionally if the argument refers to a character file or a block
special file , its st_rdev value also should be printed.

--

```
#include<sys/types.h>
#include<sys/stat.h>
#include<stdio.h>
#include<sys/sysmacros.h>

main(int argc, char *argv[])
{
        int i;
        struct stat buf;

        for(i=1;i<argc;i++)
        {
                printf("%s",argv[i]);
                if(stat(argv[i],&buf)==-1)
                {
                        printf("Stat Error");
                        continue;
                }

        printf("div=%d/%d\n",major(buf.st_dev),minor(buf.st_dev
));
                if(S_ISCHR(buf.st_mode) || S_ISBLK(buf.st_mode))
                {

        printf("(%s)rdev=%d/%d\n",(S_ISCHR(buf.st_mode))?"ch
aracter":"block",
major(buf.st_rdev),minor(buf.st_rdev));
                }
                printf("\n");

        }
                exit(0);


}
-----------------
   out put
-----------------
$ cc lab14b.c
$ ./a.out lab1a.sh lab1b.sh lab2a.sh
lab1a.sh div=3/7

lab1b.c div=3/7

lab2a.sh div=3/7
```


15a) Write a shell script to do the following:
        It accepts two filenames as arguments, sorts both to temporary files,
merges the sorted to the standard output and finally deletes the temporary files.


```
a=$#
if [ $a -ne 2 ]
then
  echo "Arguments not correctly specified"
  exit 1
fi
sort $1 -o temp1
sort $2 -o temp2
echo "The merged output from files $1 and $2"
sort -m temp1 temp2
rm temp[1-2]
```

----
  out put

```
$ cat > file1
arun
kumar
shiva
chandu
$ cat > file2
deepu
daya
$ sh lab15a.sh file1 file2
The merged output from files file1 and file2
arun
chandu
daya
deepu
```

```
kumar
shiva
---------------------------------------------------------------------------
----
```

15b) Write an awk program to print the transpose of a matrix.

```
---------------------------------------------------------------------------
----
BEGIN {
 FS=" "
 noele=1
     }

 {
   split($0,fields," ")
   for(i=1;i<=NF;i++)
   ele[noele++]=fields[i]
 }

 END {
  for(i=1;i<=NF;i++)
   {
     for(j=i;j<noele;j += NF)
     printf("%s\t",ele[j])
     printf "\n"
   }
}
-------------
 out put
-------------
$ cat > matrix
1 2 3 4 5
6 7 8 1 2
$ awk -f lab15b.awk matrix
1       6
2       7
3       8
4       1
```

5   2

Good Luck
