System Software

Lab File For 4th Semester

Submitted To

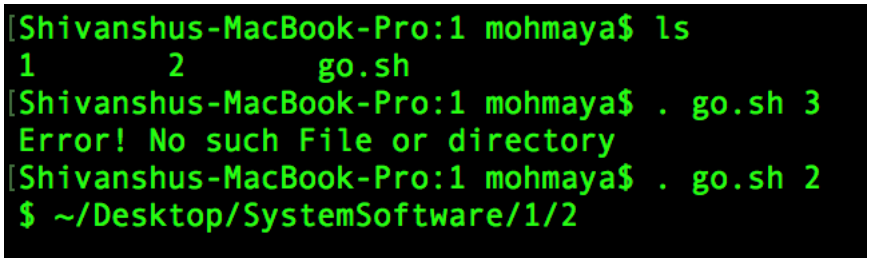
Submitted By:
Shivanshu Chaudhary
15BCS0075
B.Tech. Computer Engg.
Jamia Millia Islamia
New Delhi

INDEX

1 . Write a function go() which would change the $ prompt to the current directory name in which you are working. Thus, if you are working in directory /usr/aa10, the prompt should look like
/usr/aa10
If you execute the go() function as shown below: $ /usr/aa10>go abc
The prompt should look like
/usr/aa10/abc

```
#rename the command prompt as per the command line argument

if ! test $# -eq 1
then
        echo "Invalid format"
        echo "Right format is . go.sh [renaming part]"
        exit
fi
if ! test -d $1
then
        echo "Error! No such File or directory"
else
        cd $1
        export PS1="$ \w "
fi
```



2. Suppose a user has rename some files in current directory using command ‘mv file file.$$’. Write a shell script to search all files and rename them such that they do not contain the shell PID.

```
echo "Initially the files are:"
ls -1

for i in *
do
  mv $i $i.$$
done

echo "Files after renaming"
ls -1

echo "Now removing the extensions"

for i in *.$$
do
  mv $i ${i%.$$}
done

echo "Now the files are"
ls -1
```

```
Shivanshus-MacBook-Pro:2 mohmaya$ ./renamepid.sh
Initially the files are:
1
10
11
12
13
14
15
2
3
4
5
6
7
8
9
Files after renaming
1.4264
10.4264
11.4264
12.4264
13.4264
14.4264
15.4264
2.4264
3.4264
4.4264
5.4264
6.4264
7.4264
8.4264
9.4264
Now removing the extensions
Now the files are
1
10
11
12
13
14
15
2
3
4
5
6
7
8
9
Shivanshus-MacBook-Pro:2 mohmaya$
```

3. Write a shell script containing a function mycd() using which you should be able to shuttle between directories. The function should work in the following manner:
mycd dir1 : change the current directory to dir1
mycd : change the directory to previous directory

```
#Execute as . mycd.sh <directory if any>
if test $# -eq 0
then
  cd ..
  echo "In the directory $(pwd)"
elif test $# -eq 1
then
  if test -d $1
  then
    cd $1
    echo "In the directory $(pwd)"
  else
    echo "No such Directory exists"
  fi
else
  echo "Invalid input"
  echo "The correct format is ./mycd.sh [directory name]"
fi
```

```
[Shivanshus-MacBook-Pro:3 mohmaya$ ls
 mycd.sh testdir
[Shivanshus-MacBook-Pro:3 mohmaya$ . mycd.sh 1
 No such Directory exists
[Shivanshus-MacBook-Pro:3 mohmaya$ . mycd.sh testdir
 In the directory /Users/mohmaya/Desktop/SystemSoftware/3/testdir
[Shivanshus-MacBook-Pro:testdir mohmaya$ cd ..
[Shivanshus-MacBook-Pro:3 mohmaya$ . mycd.sh
 In the directory /Users/mohmaya/Desktop/SystemSoftware
 Shivanshus-MacBook-Pro:SystemSoftware mohmaya$
```

4. Write a function mkcd(), which would create all the directories present in the path supplied to it as argument and change over to the last directory in the path. For example,

$mkcd d1/d2/d3/d4/d5

```
if test $# -eq 0
then
  echo "Invalid Input"
  echo "mkcd [directory list]"
  exit
fi

IFS='/'

for i in $1
do
  if ! test -d $i
  then
    mkdir $i
  fi
  cd $i
done

echo "Currently in directory $(pwd)"
```

```
[Shivanshus-MacBook-Pro:4 mohmaya$ ls
 mkcd.sh
[Shivanshus-MacBook-Pro:4 mohmaya$ . mkcd.sh d1/d2/d3/d4/d5
 Currently in directory /Users/mohmaya/Desktop/SystemSoftware/4/d1/d2/d3/d4/d5
[Shivanshus-MacBook-Pro:d5 mohmaya$ cd ..
[Shivanshus-MacBook-Pro:d4 mohmaya$ cd ..
[Shivanshus-MacBook-Pro:d3 mohmaya$ cd ..
[Shivanshus-MacBook-Pro:d2 mohmaya$ cd ..
[Shivanshus-MacBook-Pro:d1 mohmaya$ cd ..
[Shivanshus-MacBook-Pro:4 mohmaya$ ls
 d1      mkcd.sh
 Shivanshus-MacBook-Pro:4 mohmaya$
```

5. Write a shell script which will receive any number of filenames as arguments. The shell script check whether such files already exists. If they do, it should be reported. If these files do not exist, then check if a subdirectory mydir exist in the current directory. If it does not exists then it should be created. If mydir already exists then it should be reported along with the number of files that are currently present in mydir.

```
if test $# -eq 0
then
  echo "Invalid Input"
  echo "./fileordir.sh [file list]"
  exit
fi

myarr=("$@")
count=0
for i in ${myarr[@]}
do
  if test -f $i
  then
    echo "$i is a file"
    count=`expr $count + 1`
  else
    echo "$i is not a file"
  fi
done
```

```
if test $count -eq 0
then
  echo "None of the files exist"
  if ! test -d mydir
  then
    echo "mydir doesn't exist. Being Created ..."
    mkdir mydir
  else
    echo "mydir exists"
    cd mydir
    nf=$(ls -l | wc -l)
    echo "No. of files in mydir : $nf"
  fi
fi
```

```
[Shivanshus-MacBook-Pro:5 mohmaya$ ls
 1                2                fileordir.sh
[Shivanshus-MacBook-Pro:5 mohmaya$ ./fileordir.sh 1 2 3
 1 is a file
 2 is a file
 3 is not a file
[Shivanshus-MacBook-Pro:5 mohmaya$ ./fileordir.sh 3 4 5
 3 is not a file
 4 is not a file
 5 is not a file
 None of the files exist
 mydir doesn't exist. Being Created ...
[Shivanshus-MacBook-Pro:5 mohmaya$ ./fileordir.sh 3 4 5
 3 is not a file
 4 is not a file
 5 is not a file
 None of the files exist
 mydir exists
 No. of files in mydir :        0
 Shivanshus-MacBook-Pro:5 mohmaya$
```

6. Write a shell script which reports names and sizes of all the files in a directory whose size is exceeding 512 bytes. The filenames should be printed in descending order of their sizes. The total number of such files should also be reported.

```
cd test
ls -1S>fillist

exec<fillist

while read line
do
  if test -f $line
  then
    size=`ls -l|grep -w $line|tr -s ' '|cut -f5 -d' '`
    if test $size -gt 1000
    then
      echo "$line  :  $size bytes"
    fi
  fi
done
```

```
[Shivanshus-MacBook-Pro:test mohmaya$ ls -l
total 152
-rw-r--r--  1 mohmaya  staff  2000 May 24 17:34 1
-rw-r--r--  1 mohmaya  staff  2000 May 24 17:34 10
-rw-r--r--  1 mohmaya  staff  1028 May 24 17:34 11
-rw-r--r--  1 mohmaya  staff  1028 May 24 17:34 12
-rw-r--r--  1 mohmaya  staff  1028 May 24 17:34 13
-rw-r--r--  1 mohmaya  staff  1028 May 24 17:34 14
-rw-r--r--  1 mohmaya  staff   102 May 24 17:34 15
-rw-r--r--  1 mohmaya  staff   102 May 24 17:34 16
-rw-r--r--  1 mohmaya  staff   102 May 24 17:34 17
-rw-r--r--  1 mohmaya  staff   102 May 24 17:34 18
-rw-r--r--  1 mohmaya  staff  2000 May 24 17:34 2
-rw-r--r--  1 mohmaya  staff  2000 May 24 17:34 3
-rw-r--r--  1 mohmaya  staff  2000 May 24 17:34 4
-rw-r--r--  1 mohmaya  staff  2000 May 24 17:34 5
-rw-r--r--  1 mohmaya  staff  2000 May 24 17:34 6
-rw-r--r--  1 mohmaya  staff  2000 May 24 17:34 7
-rw-r--r--  1 mohmaya  staff  2000 May 24 17:34 8
-rw-r--r--  1 mohmaya  staff  2000 May 24 17:34 9
-rw-r--r--  1 mohmaya  staff    53 May 24 17:35 fillist
[Shivanshus-MacBook-Pro:test mohmaya$ cd ..
[Shivanshus-MacBook-Pro:6 mohmaya$ ./sizrep.sh
./sizrep.sh: line 11: test: too many arguments
10  :  2000 bytes
2  :  2000 bytes
3  :  2000 bytes
4  :  2000 bytes
5  :  2000 bytes
6  :  2000 bytes
7  :  2000 bytes
8  :  2000 bytes
9  :  2000 bytes
11  :  1028 bytes
12  :  1028 bytes
13  :  1028 bytes
14  :  1028 bytes
```

7. Write a shell script for renaming each file in the directory such that it will have current shell PID as extension. The shell script should ensure that the directories do not get renamed.

```
cd files
for i in *
do
  if test -f $i
  then
    mv $i $i.$$
  fi
done
```

```
[Shivanshus-MacBook-Pro:7 mohmaya$ ls
 PIDrename.sh    files
[Shivanshus-MacBook-Pro:7 mohmaya$ cd files
[Shivanshus-MacBook-Pro:files mohmaya$ ls
 1        2        3        4        5        6
[Shivanshus-MacBook-Pro:files mohmaya$ cd ..
[Shivanshus-MacBook-Pro:7 mohmaya$ ./PIDrename.sh
[Shivanshus-MacBook-Pro:7 mohmaya$ cd files/
[Shivanshus-MacBook-Pro:files mohmaya$ ls
 1.4502  2.4502  3.4502  4.4502  5.4502  6.4502
 Shivanshus-MacBook-Pro:files mohmaya$
```

8. The word Unix is present in only some files supplied as arguments to the shell script. Your shell script should search each of these files in turn and stop at the first file that it encounters containing the word Unix. This filename should be displayed on the screen.

```
if test $# -eq 0
then
  echo "No file specified"
  echo "Quitting...."
  exit
fi

myarr=("$@")

for i in ${myarr[@]}
do
  if ! test -f $i
  then
    echo "$i is not a file"
  else
    x=$(grep -w Unix $i|wc -l)
    if ! test $x -eq 0
    then
      echo "First occurance of Unix found in $i"
      exit
    fi
  fi
done

echo "Unix not found”
```

```
[Shivanshus-MacBook-Pro:8 mohmaya$ ls
1                2                3                UNIXFinder.sh
[Shivanshus-MacBook-Pro:8 mohmaya$ cat 1
Hello
How are you
UNIXlover
[Shivanshus-MacBook-Pro:8 mohmaya$ cat 2
Hello
How are You
I love Unix
[Shivanshus-MacBook-Pro:8 mohmaya$ cat 3
hey!
How you doin?
[Shivanshus-MacBook-Pro:8 mohmaya$ ./UNIXFinder.sh
No file specified
Quitting....
[Shivanshus-MacBook-Pro:8 mohmaya$ ./UNIXFinder.sh 1 3
Unix not found
[Shivanshus-MacBook-Pro:8 mohmaya$ ./UNIXFinder.sh 1 2 3
First occurance of Unix found in 2
Shivanshus-MacBook-Pro:8 mohmaya$
```

9. Write a shell script using function to calculate the LCM and HCF of two numbers.

```
HCF()
{
  if test $2 -eq 0
  then
    retval=$1
  elif test $2 -gt $1
  then
    HCF $2 $1
    retval=$?
  else
    val=`expr $1 % $2`
    HCF $2 $val
    retval=$?
  fi
  return "$retval"
}
if ! test $# -eq 2
then
  echo "Wrong Input... Correct format is ./LCMHCF.sh <number 1> <number 2>"
  exit
fi
HCF $1 $2
hcf=$?
prod=`expr $1 \* $2`
lcm=`expr $prod / $hcf`
echo "HCF: $hcf"
echo "LCM: $lcm"
```

```
[Shivanshus-MacBook-Pro:9 mohmaya$ ./LCMHCF.sh
 Wrong Input... Correct format is ./LCMHCF.sh <number 1> <number 2>
[Shivanshus-MacBook-Pro:9 mohmaya$ ./LCMHCF.sh 2
 Wrong Input... Correct format is ./LCMHCF.sh <number 1> <number 2>
[Shivanshus-MacBook-Pro:9 mohmaya$ ./LCMHCF.sh 11 19
 HCF: 1
 LCM: 209
[Shivanshus-MacBook-Pro:9 mohmaya$ ./LCMHCF.sh 36 48
 HCF: 12
 LCM: 144
 Shivanshus-MacBook-Pro:9 mohmaya$
```

10. Write a shell script which will receive any number of filenames as arguments. The shell script should check whether every argument supplied is a file or a directory. If it is a directory it should be appropriately reported. If it is a filename then name of the file as well as the number of lines present in it should be reported.

```
myarr=("$@")
for i in ${myarr[@]}
do
  if test -d $i
  then
    echo "$i is a directory"
  elif test -f $i
  then
    cn=`wc -l $i|tr -s ' '|cut -f2 -d' '`
    echo "$i is a File with $cn lines."
  fi
done
```

```
[Shivanshus-MacBook-Pro:10 mohmaya$ ls -l
 total 24
 drwxr-xr-x  2 mohmaya  staff   68 May 24 16:45 1
 -rw-r--r--  1 mohmaya  staff    6 May 24 16:46 2
 -rw-r--r--  1 mohmaya  staff   27 May 24 16:46 3
 -rwxrwxrwx  1 mohmaya  staff  206 May 24 17:12 fildir.sh
[Shivanshus-MacBook-Pro:10 mohmaya$ cat 2
 hello
[Shivanshus-MacBook-Pro:10 mohmaya$ cat 3
 WALALALALA
 asdadasdasd
 asc
[Shivanshus-MacBook-Pro:10 mohmaya$ ./fildir.sh 1 2 3
 1 is a directory
 2 is a File with 1 lines.
 3 is a File with 3 lines.
 Shivanshus-MacBook-Pro:10 mohmaya$
```

11. Write a shell script of cut n lines starting from position m from a file without using head or tail command

```
if ! test $# -eq 3
then
  echo "Invalid Input format"
  echo "Format is: ./cutn.sh <m> <n>"
fi
fname=$1
m=$2
n=$3
n=`expr $m + $n`
if test -f $fname
then
        cn=1
        exec<$fname
        while read line
        do
                if test $cn -ge $m
                then
                        if test $cn -lt $n
                        then
                                echo $line
                        fi
                fi
                cn=`expr $cn + 1`
        done
else
        echo "No such file"
        exit 1
fi
```

```
[Shivanshus-MacBook-Pro:11 mohmaya$ ./cutn.sh 3 0 2
 asdadasdasd
[Shivanshus-MacBook-Pro:11 mohmaya$ ./cutn.sh 3 0 3
 asdadasdasd
 asc
[Shivanshus-MacBook-Pro:11 mohmaya$ ./cutn.sh 3 0 1
[Shivanshus-MacBook-Pro:11 mohmaya$ ./cutn.sh 3 1 1
 asdadasdasd
 Shivanshus-MacBook-Pro:11 mohmaya$
```

12. Write a shell script which reports names and sizes of all files in a directory whose size is exceeding 1000 bytes. The file names should be printed in descending order of their sizes. The total number of such files should also be reported.

```
cd test
ls -1S>fillist

exec<fillist
count=0
while read line
do
  if test -f $line
  then
    size=`ls -l|grep -w $line|tr -s ' '|cut -f5 -d' '`
    if test $size -gt 1000
    then
      echo "$line  :  $size bytes"
      count=`expr $count + 1`
    fi
  fi
done
echo "Total number of Such Files : $count"
```

```
[Shivanshus-MacBook-Pro:12 mohmaya$ cd test
[Shivanshus-MacBook-Pro:test mohmaya$ ls -l
 total 152
 -rw-r--r--  1 mohmaya  staff  2000 May 24 17:30 1
 -rw-r--r--  1 mohmaya  staff  2000 May 24 17:30 10
 -rw-r--r--  1 mohmaya  staff  1028 May 24 17:30 11
 -rw-r--r--  1 mohmaya  staff  1028 May 24 17:30 12
 -rw-r--r--  1 mohmaya  staff  1028 May 24 17:30 13
 -rw-r--r--  1 mohmaya  staff  1028 May 24 17:30 14
 -rw-r--r--  1 mohmaya  staff   102 May 24 17:31 15
 -rw-r--r--  1 mohmaya  staff   102 May 24 17:31 16
 -rw-r--r--  1 mohmaya  staff   102 May 24 17:31 17
 -rw-r--r--  1 mohmaya  staff   102 May 24 17:31 18
 -rw-r--r--  1 mohmaya  staff  2000 May 24 17:30 2
 -rw-r--r--  1 mohmaya  staff  2000 May 24 17:30 3
 -rw-r--r--  1 mohmaya  staff  2000 May 24 17:30 4
 -rw-r--r--  1 mohmaya  staff  2000 May 24 17:30 5
 -rw-r--r--  1 mohmaya  staff  2000 May 24 17:30 6
 -rw-r--r--  1 mohmaya  staff  2000 May 24 17:30 7
 -rw-r--r--  1 mohmaya  staff  2000 May 24 17:30 8
 -rw-r--r--  1 mohmaya  staff  2000 May 24 17:30 9
 -rw-r--r--  1 mohmaya  staff    53 May 24 17:33 fillist
[Shivanshus-MacBook-Pro:test mohmaya$ cd ..
[Shivanshus-MacBook-Pro:12 mohmaya$ ls
 sizrep.sh       test
[Shivanshus-MacBook-Pro:12 mohmaya$ ./sizrep.sh
 ./sizrep.sh: line 11: test: too many arguments
 10  :  2000 bytes
 2  :  2000 bytes
 3  :  2000 bytes
 4  :  2000 bytes
 5  :  2000 bytes
 6  :  2000 bytes
 7  :  2000 bytes
 8  :  2000 bytes
 9  :  2000 bytes
 11  :  1028 bytes
 12  :  1028 bytes
 13  :  1028 bytes
 14  :  1028 bytes
 ./sizrep.sh: line 11: test: too many arguments
 Total number of Such Files : 13
```

13. Write a shell script to generate a matrix of order n * m.

```
echo "Matrix 1"
echo "Enter N(rows) and M(Cols)"
read n m



r=`expr $n - 1`
c=`expr $m - 1`

for i in $(seq 0 $r)
do
  for j in $(seq 0 $c)
  do
    echo "Enter Value at Row: $i Column: $j"
    read x
    y=`expr $i \* $m`
    y=`expr $y + $j`
    arr[$y]=$x
  done
done

echo "THE MATRIX IS: "
for i in $(seq 0 $r)
do
  for j in $(seq 0 $c)
  do
    y=`expr $i \* $m`
    y=`expr $y + $j`
    echo -n "${arr[$y]}  "
  done
  echo ""
done
```

```
[Shivanshus-MacBook-Pro:13 mohmaya$ ./Matrixnm.sh
Matrix 1
Enter N(rows) and M(Cols)
3 2
Enter Value at Row: 0 Column: 0
0
Enter Value at Row: 0 Column: 1
1
Enter Value at Row: 1 Column: 0
2
Enter Value at Row: 1 Column: 1
3
Enter Value at Row: 2 Column: 0
4
Enter Value at Row: 2 Column: 1
5
THE MATRIX IS:
0  1
2  3
4  5
[Shivanshus-MacBook-Pro:13 mohmaya$ ./Matrixnm.sh
Matrix 1
Enter N(rows) and M(Cols)
2 3
Enter Value at Row: 0 Column: 0
0
Enter Value at Row: 0 Column: 1
1
Enter Value at Row: 0 Column: 2
2
Enter Value at Row: 1 Column: 0
3
Enter Value at Row: 1 Column: 1
4
Enter Value at Row: 1 Column: 2
5
THE MATRIX IS:
0  1  2
3  4  5
```

14. Write a shell script to implement bubble sort algorithm on n numbers.

```
echo "enter number of elements in array"
read n
# taking input from user
echo "enter Numbers in array:"
for (( i = 0; i < $n; i++ ))
do
  echo "Enter $i th element of the array"
  read nos[$i]
done
#printing the number before sorting
echo "Numbers in an array are:"
for (( i = 0; i < $n; i++ ))
do
  echo -n "${nos[$i]}   "
done
# Now do the Sorting of numbers
for (( i = 0; i < $n ; i++ ))
do
  for (( j = $i; j < $n; j++ ))
  do
    if [ ${nos[$i]} -gt ${nos[$j]}  ]
    then
      t=${nos[$i]}
      nos[$i]=${nos[$j]}
      nos[$j]=$t
    fi
  done
done
echo ""
# Printing the sorted number
echo "Sorted Numbers "
```

```
for (( i=0; i < $n; i++ ))
do
  echo -n "${nos[$i]}   "
done
echo ""
```

```
Shivanshus-MacBook-Pro:14 mohmaya$ ./bubblesort.sh
enter number of elements in array
6
enter Numbers in array:
Enter 0 th element of the array
23
Enter 1 th element of the array
34
Enter 2 th element of the array
12
Enter 3 th element of the array
6
Enter 4 th element of the array
67
Enter 5 th element of the array
13
Numbers in an array are:
23   34   12   6   67   13
Sorted Numbers
6   12   13   23   34   67
Shivanshus-MacBook-Pro:14 mohmaya$
```

15. Write a shell script to print the multiplication of two matrix of order n * m.

```
echo " For Matrix 1"
echo "Enter N(rows) and M(Cols)"
read n m


echo "For Matrix 2"
echo "Enter O(Rows) and P(Cols)"
read o p



if ! test $m -eq $o
then
  echo "Incompatible Matrix Orders"
  exit
fi


# Accepting Matrix 1
echo " For Matrix 1"
r1=`expr $n - 1`
c1=`expr $m - 1`

for i in $(seq 0 $r1)
do
  for j in $(seq 0 $c1)
  do
    echo "Enter Value at Row: $i Column: $j"
    read x
    y=`expr $i \* $m`
    y=`expr $y + $j`
```

```
    arr1[$y]=$x
  done
done


#Accepting Matrix 2
echo "For Matrix 2"
r2=`expr $o - 1`
c2=`expr $p - 1`

for i in $(seq 0 $r2)
do
  for j in $(seq 0 $c2)
  do
    echo "Enter Value at Row: $i Column: $j"
    read x
    y=`expr $i \* $p`
    y=`expr $y + $j`
    arr2[$y]=$x
  done
done



echo "THE MATRIX 1 IS: "
for i in $(seq 0 $r1)
do
  for j in $(seq 0 $c1)
  do
    y=`expr $i \* $m`
    y=`expr $y + $j`
    echo -n "${arr1[$y]}  "
  done
```

```
  echo ""
done



echo "THE MATRIX 2 IS: "
for i in $(seq 0 $r2)
do
  for j in $(seq 0 $c2)
  do
    y=`expr $i \* $p`
    y=`expr $y + $j`
    echo -n "${arr2[$y]}  "
  done
  echo ""
done


#Multiplication

for i in $(seq 0 $r1)
do
  for j in $(seq 0 $c2)
  do
    sum=0
    for k in $(seq 0 $c1)
    do
      y1=`expr $i \* $m`
      y1=`expr $y1 + $k`
      y2=`expr $k \* $p`
      y2=`expr $y2 + $j`
      val1=${arr1[$y1]}
      val2=${arr2[$y2]}
```

```
      prod=`expr $val1 \* $val2`
      sum=`expr $sum + $prod`
    done
    y=`expr $i \* $p`
    y=`expr $y + $j`
    arr3[$y]=$sum;
  done
done

echo "Final Matrix is"
for i in $(seq 0 $r1)
do
  for j in $(seq 0 $c2)
  do
    y=`expr $i \* $p`
    y=`expr $y + $j`
    echo -n "${arr3[$y]}    "
  done
  echo ""
done
```

```
[Shivanshus-MacBook-Pro:15 mohmaya$ ./MatrixMult.sh
 For Matrix 1
Enter N(rows) and M(Cols)
2 3
For Matrix 2
Enter O(Rows) and P(Cols)
2 3
Incompatible Matrix Orders
[Shivanshus-MacBook-Pro:15 mohmaya$ ./MatrixMult.sh
 For Matrix 1
Enter N(rows) and M(Cols)
2 3
For Matrix 2
Enter O(Rows) and P(Cols)
3 1
 For Matrix 1
Enter Value at Row: 0 Column: 0
1
Enter Value at Row: 0 Column: 1
2
Enter Value at Row: 0 Column: 2
3
Enter Value at Row: 1 Column: 0
4
Enter Value at Row: 1 Column: 1
5
Enter Value at Row: 1 Column: 2
6
For Matrix 2
Enter Value at Row: 0 Column: 0
1
Enter Value at Row: 1 Column: 0
9
Enter Value at Row: 2 Column: 0
8
THE MATRIX 1 IS:
1  2  3
4  5  6
THE MATRIX 2 IS:
1
9
8
Final Matrix is
43
97
Shivanshus-MacBook-Pro:15 mohmaya$
```

16. The word ‘linux’ is present in only some of the files supplied as arguments to the shell script. Your shell script should search each of the file and stops at the first file that it encounters containing the word ‘linux’. This filename should be displayed on the screen.

```
if test $# -eq 0
then
  echo "No file specified"
  echo "Quitting...."
  exit
fi

myarr=("$@")

for i in ${myarr[@]}
do
  if ! test -f $i
  then
    echo "$i is not a file"
  else
    x=$(grep -w linux $i|wc -l)
    if ! test $x -eq 0
    then
      echo "First occurance of linux found in $i"
      exit
    fi
  fi
done

echo "linux not found"
```

```
[Shivanshus-MacBook-Pro:16 mohmaya$ cat 1
 HEY!!!
 HOW YOU DOIN ;)
[Shivanshus-MacBook-Pro:16 mohmaya$ cat 2
 U kno i like
 linux
[Shivanshus-MacBook-Pro:16 mohmaya$ cat 3
 linux is great!!
 but not as great as MACos
[Shivanshus-MacBook-Pro:16 mohmaya$ ./srclinux.sh 1
 linux not found
[Shivanshus-MacBook-Pro:16 mohmaya$ ./srclinux.sh 1 2 3
 First occurance of linux found in 2
[Shivanshus-MacBook-Pro:16 mohmaya$ ./srclinux.sh 1 3 2
 First occurance of linux found in 3
 Shivanshus-MacBook-Pro:16 mohmaya$
```

17. A shell script receives even number of filenames. Suppose four filenames are supplied then the first file should get copied into second file, the third file should get copied into fourth file, and so on. If odd number of filenames are supplied then no copying should take place and an error message should be displayed.

```
cd files
nfiles=$#
rem=`expr $nfiles % 2`
cpy()
{
  cp $1 $2
}
if test $# -eq 0
then
  echo "No files entered."
  exit
fi
if test $rem -eq 1
then
  echo "Odd files entered. Quitting..."
  exit
fi
myarr=("$@")
for i in ${myarr[@]}
do
  if ! test -f $i
  then
    echo "$i is not a file. Quitting..."
    exit
  fi
done
a=0
while test $a -lt $#
do
```

```
  b=`expr $a + 1`
  f1=${myarr[$a]}
  f2=${myarr[$b]}
  cpy $f1 $f2
  a=`expr $a + 2`
done
```

```
[Shivanshus-MacBook-Pro:17 mohmaya$ ls ./files
 1        2        3        4        5        6
[Shivanshus-MacBook-Pro:17 mohmaya$ cat ./files/1
 asdfgh
[Shivanshus-MacBook-Pro:17 mohmaya$ cat ./files/2
 qwerty
[Shivanshus-MacBook-Pro:17 mohmaya$ cat ./files/3
 asdfgh
[Shivanshus-MacBook-Pro:17 mohmaya$ cat ./files/4
 qwerty
[Shivanshus-MacBook-Pro:17 mohmaya$ cat ./files/5
 asdfgh
[Shivanshus-MacBook-Pro:17 mohmaya$ cat ./files/6
 qwerty
[Shivanshus-MacBook-Pro:17 mohmaya$ ./evod.sh 1 2 3
 Odd files entered. Quitting...
[Shivanshus-MacBook-Pro:17 mohmaya$ ./evod.sh 1 2 7
 Odd files entered. Quitting...
[Shivanshus-MacBook-Pro:17 mohmaya$ ./evod.sh 1 2 7 8
 7 is not a file. Quitting...
[Shivanshus-MacBook-Pro:17 mohmaya$ ./evod.sh 1 2 3 4
[Shivanshus-MacBook-Pro:17 mohmaya$ cat ./files/1
 asdfgh
[Shivanshus-MacBook-Pro:17 mohmaya$ cat ./files/2
 asdfgh
[Shivanshus-MacBook-Pro:17 mohmaya$ cat ./files/3
 asdfgh
[Shivanshus-MacBook-Pro:17 mohmaya$ cat ./files/4
 asdfgh
[Shivanshus-MacBook-Pro:17 mohmaya$ cat ./files/5
 asdfgh
[Shivanshus-MacBook-Pro:17 mohmaya$ cat ./files/6
 qwerty
 Shivanshus-MacBook-Pro:17 mohmaya$
```

18. Write a program in C to implement: Usual Sender-Receiver (One Texts, the other Read).

shm_com.h

```
#define TEXT_SZ 2048

struct shared_use_st{
  int written_by_you;
  char some_text[TEXT_SZ];
};
```

1.cpp

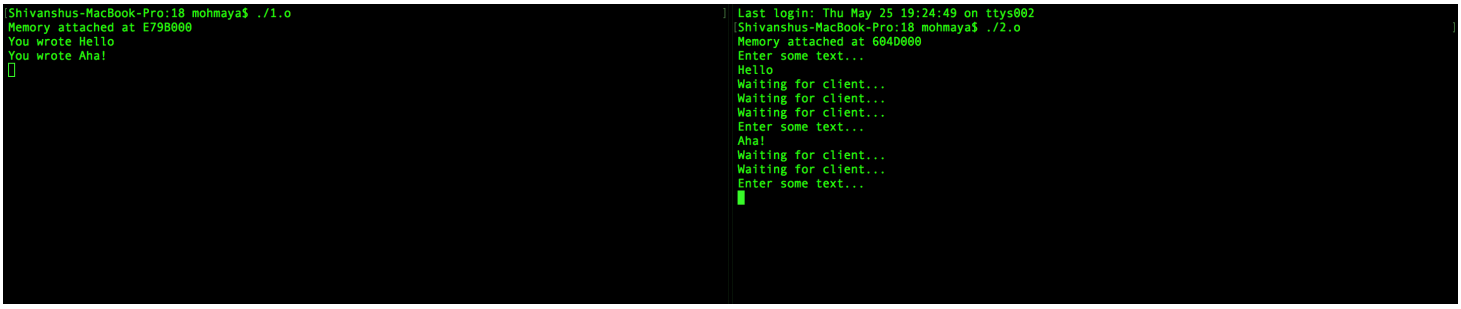
```
//Receiver
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/shm.h>
#include "shm_com.h"
int main() {
  int running = 1;
  void *shared_memory = (void *)0;
  struct shared_use_st *shared_stuff;
  int shmid;
  srand((unsigned int)getpid());
  shmid = shmget((key_t)1234, sizeof(struct shared_use_st),0666|IPC_CREAT);
  if(shmid == -1)
  {
     fprintf(stderr,"shmget failed\n");
     exit(EXIT_FAILURE);
  }
  shared_memory = shmat(shmid,(void *)0,0);
  if(shared_memory == (void *)-1)
```

```
  {
    fprintf(stderr,"shmat failed\n");
    exit(EXIT_FAILURE);
  }
  printf("Memory attached at %X\n",shared_memory);
  shared_stuff = (struct shared_use_st *)shared_memory;
  shared_stuff -> written_by_you = 0;
  while(running)
  {
   if(shared_stuff -> written_by_you)
   {
    printf("You wrote %s",shared_stuff->some_text);
    sleep(rand() % 4);
    shared_stuff -> written_by_you = 0;
    if(strcmp(shared_stuff->some_text,"end") == 0)
    {
     running = 0;
    }
   }
  }
  if(shmdt(shared_memory) == -1)
  {
   fprintf(stderr,"shmdt failed\n");
   exit(EXIT_FAILURE);
  }
  if(shmctl(shmid, IPC_RMID, 0) == -1)
  {
   fprintf(stderr,"shmctl(IPC_RMID) Failed\n");
   exit(EXIT_FAILURE);
  }
  exit(EXIT_SUCCESS);
}
```

2.cpp

```
//Sender
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/shm.h>
#include "shm_com.h"
int main()
{
  int running = 1;
  void *shared_memory = (void *)0;
  struct shared_use_st *shared_stuff;
  char buffer[BUFSIZ];
  int shmid;
  srand((unsigned int)getpid());
  shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666|IPC_CREAT);
  if(shmid == -1)
  {
     fprintf(stderr,"shmget failed\n");
     exit(EXIT_FAILURE);
  }
  shared_memory = shmat(shmid,(void *)0,0);
  if(shared_memory == (void *)-1)
  {
     fprintf(stderr,"shmat failed\n");
     exit(EXIT_FAILURE);
  }
  printf("Memory attached at %X\n",shared_memory);
  shared_stuff = (struct shared_use_st *)shared_memory;
  while(running)
  {
     while(shared_stuff -> written_by_you == 1)
```

```
    {
      sleep(1);
      printf("Waiting for client...\n");
    }
    printf("Enter some text... \n");
    fgets(buffer,BUFSIZ,stdin);
    strncpy(shared_stuff -> some_text, buffer, TEXT_SZ);
    shared_stuff -> written_by_you = 1;
    if(strcmp(shared_stuff -> some_text, "end") == 0)
    {
    running = 0;
    }
  }
  if(shmdt(shared_memory) == -1)
  {
    fprintf(stderr,"shmdt failed\n");
    exit(EXIT_FAILURE);
  }
  exit(EXIT_SUCCESS);
}
```



19. Write a program in C to: Producer generates two numbers, receiver adds them and displays the result.

shm_com.h

```
struct shared_use_st{
  int written_by_you;
  int aa,bb;
};
```

1.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/shm.h>
#include "shm_com.h"
int main() {
  int running = 1;
  void *shared_memory = (void *)0;
  struct shared_use_st *shared_stuff;
  int shmid;
  srand((unsigned int)getpid());
  shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666|IPC_CREAT);
  if(shmid == -1)
  {
     fprintf(stderr,"shmget failed\n");
     exit(EXIT_FAILURE);
  }
  shared_memory = shmat(shmid,(void *)0,0);
  if(shared_memory == (void *)-1)
  {
     fprintf(stderr,"shmat failed\n");
     exit(EXIT_FAILURE);
```

```
  }
  printf("Memory attached at %X\n",shared_memory);
  shared_stuff = (struct shared_use_st *)shared_memory;
  shared_stuff -> written_by_you = 0;
  while(running)
  {
     if(shared_stuff -> written_by_you)
     {
       printf("1st Number= %d\n",shared_stuff -> aa);
       printf("2nd Number= %d\n",shared_stuff -> bb);
       if(shared_stuff -> aa == 0 && shared_stuff -> bb == 0)
       {
         running = 0;
       }
       printf("The Sum of two numbers is %d\n\n",(shared_stuff -> aa)+(shared_stuff -> bb));
       shared_stuff -> written_by_you = 0;
     }
  }
  if(shmdt(shared_memory) == -1)
  {
     fprintf(stderr,"shmdt failed\n");
     exit(EXIT_FAILURE);
  }
  if(shmctl(shmid, IPC_RMID, 0) == -1)
  {
     fprintf(stderr,"shmctl(IPC_RMID) Failed\n");
     exit(EXIT_FAILURE);
  }
  exit(EXIT_SUCCESS);
}
```

2.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/shm.h>
#include "shm_com.h"
int main() {
    int running = 1;
    int i,j;
    void *shared_memory = (void *)0;
    struct shared_use_st *shared_stuff;
    int shmid;
    srand((unsigned int)getpid());
    shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666|IPC_CREAT);
    if(shmid == -1)
    {
        fprintf(stderr,"shmget failed\n");
        exit(EXIT_FAILURE);
    }
    shared_memory = shmat(shmid,(void *)0,0);
    if(shared_memory == (void *)-1)
    {
        fprintf(stderr,"shmat failed\n");
        exit(EXIT_FAILURE);
    }
    printf("Memory attached at %X\n",shared_memory);
    shared_stuff = (struct shared_use_st *)shared_memory;
    while(running)
    {
      while(shared_stuff -> written_by_you == 1)
      {
```

```
      sleep(1);
    }
    printf("Enter two numbers: \n");
    scanf("%d%d",&i,&j);
    shared_stuff -> aa = i;
    shared_stuff -> bb = j;
    shared_stuff -> written_by_you = 1;
    if(shared_stuff -> aa == 0 && shared_stuff -> bb == 0)
    {
      running = 0;
    }
  }
  if(shmdt(shared_memory) == -1)
  {
     fprintf(stderr,"shmdt failed\n"); exit(EXIT_FAILURE);
  }
  exit(EXIT_SUCCESS);
}
```

```
Shivanshus-MacBook-Pro:19 mohmaya$ ./1.0
-bash: ./1.0: No such file or directory
Shivanshus-MacBook-Pro:19 mohmaya$ ./1.o
Memory attached at CEBD000
1st Number= 2
2nd Number= 3
The product of two numbers is 6

1st Number= 4
2nd Number= 5
The product of two numbers is 20
```

```
Shivanshus-MacBook-Pro:19 mohmaya$ ./2.o
Memory attached at DDF3000
Enter two numbers:
2
3
Enter two numbers:
4 5
Enter two numbers:
```

20. Write a program in C to implement Two-way chat.

shm_com.h

```
#define TEXT_SZ 2048

struct shared_use_st{
  int written_by_you;
  char some_text[TEXT_SZ];
};
```

1.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/shm.h>
#include "shm_com.h"
int main()
{
    int running = 1;
    void *shared_memory = (void *)0;
    char buffer[BUFSIZ];
    struct shared_use_st *shared_stuff;
    int shmid;
    srand((unsigned int)getpid());
    shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666|IPC_CREAT);
    if(shmid == -1)
    {
        fprintf(stderr,"shmget failed\n");
        exit(EXIT_FAILURE);
    }
    shared_memory = shmat(shmid,(void *)0,0);
    if(shared_memory == (void *)-1)
```

```
    {
        fprintf(stderr,"shmat failed\n");
        exit(EXIT_FAILURE);
    }
    printf("Memory attached at %X\n",shared_memory);
    shared_stuff = (struct shared_use_st *)shared_memory;
    shared_stuff -> written_by_you = 0;
    while(running)
    {
        printf("You: ");
        fgets(buffer,BUFSIZ,stdin);
        strncpy(shared_stuff -> some_text, buffer, TEXT_SZ);
        shared_stuff -> written_by_you = 1;
        while(shared_stuff -> written_by_you == 1)
        {}
        printf("Friend: %s",shared_stuff->some_text);
    }
    if(shmdt(shared_memory) == -1)
    {
        fprintf(stderr,"shmdt failed\n");
        exit(EXIT_FAILURE);
    }
    if(shmctl(shmid, IPC_RMID, 0) == -1)
    {
        fprintf(stderr,"shmctl(IPC_RMID) Failed\n");
        exit(EXIT_FAILURE);
    }
    exit(EXIT_SUCCESS);
}
```

2.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/shm.h>
#include "shm_com.h"
int main() {
    int running = 1;
    void *shared_memory = (void *)0;
    struct shared_use_st *shared_stuff;
    char buffer[BUFSIZ];
    int shmid;
    srand((unsigned int)getpid());
    shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666|IPC_CREAT);
    if(shmid == -1)
    {
        fprintf(stderr,"shmget failed\n");
        exit(EXIT_FAILURE);
    }
    shared_memory = shmat(shmid,(void *)0,0);
    if(shared_memory == (void *)-1)
    {
        fprintf(stderr,"shmat failed\n");
        exit(EXIT_FAILURE);
    }
    printf("Memory attached at %X\n",shared_memory);
    shared_stuff = (struct shared_use_st *)shared_memory;
    while(running)
    {
        while(shared_stuff -> written_by_you==0){}
        printf("Friend: %s",shared_stuff->some_text); printf("You: ");
```

```
        fgets(buffer,BUFSIZ,stdin);
        strncpy(shared_stuff -> some_text, buffer, TEXT_SZ);
        shared_stuff -> written_by_you = 0;
    }
    if(shmdt(shared_memory) == -1)
    {
        fprintf(stderr,"shmdt failed\n"); exit(EXIT_FAILURE);
    }
    exit(EXIT_SUCCESS);
}
```

```
Shivanshus-MacBook-Pro:20 mohmaya$ ./1.o
Memory attached at 8F73000
You: Hello
Friend: Hi
You: how you Doin?
Friend: Blush
You:
```

```
Shivanshus-MacBook-Pro:20 mohmaya$ ./2.o
Memory attached at E157000
Friend: Hello
You: Hi
Friend: how you Doin?
You: Blush
```

21. Write a program in C to implement: ONE generates array, NEXT sorts the array, LAST searches for a particular element (binary search)

shm_com.h

```
#define ARR_SZ 20

struct shared_use_st{
  int nn;
  int written_by_you;
  int array[100];
};
```

1.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/shm.h>
#include "shm_com.h"
int main() {
    int running = 1;
    int n,i,ar[50];
    void *shared_memory = (void *)0;
    struct shared_use_st *shared_stuff;
    int shmid;
    srand((unsigned int)getpid());
    shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666|IPC_CREAT);
    if(shmid == -1)
    {
        fprintf(stderr,"shmget failed\n");
        exit(EXIT_FAILURE);
    }
```

```
    shared_memory = shmat(shmid,(void *)0,0);
    if(shared_memory == (void *)-1)
    {
        fprintf(stderr,"shmat failed\n");
        exit(EXIT_FAILURE);
    }
    printf("Memory attached at %X\n",shared_memory);
    shared_stuff = (struct shared_use_st *)shared_memory;
    shared_stuff -> written_by_you = 0;
    while(running)
    {
        printf("Enter the Number of Elements: \n");
        scanf("%d",&n);
        shared_stuff -> nn = n;
        printf("Enter the Elements: \n");
        for(i=0;i<n;i++)
        {
            scanf("%d",&ar[i]);
            shared_stuff -> array[i] = ar[i];
        }
        shared_stuff -> written_by_you = 1;
        while(shared_stuff -> written_by_you != 3){}
        break;
    }
    if(shmdt(shared_memory) == -1)
    {
        fprintf(stderr,"shmdt failed\n");
        exit(EXIT_FAILURE);
    }
    exit(EXIT_SUCCESS);
}
```

2.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/shm.h>
#include "shm_com.h"
int main() {
    int running = 1;
    int n,i,j,temp;
    void *shared_memory = (void *)0;
    struct shared_use_st *shared_stuff;
    int shmid;
    srand((unsigned int)getpid());
    shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666|IPC_CREAT);
    if(shmid == -1)
    {
      fprintf(stderr,"shmget failed\n");
      exit(EXIT_FAILURE);
    }
    shared_memory = shmat(shmid,(void *)0,0);
    if(shared_memory == (void *)-1)
    {
        fprintf(stderr,"shmat failed\n");
        exit(EXIT_FAILURE);
    }
    printf("Memory attached at %X\n",shared_memory);
    shared_stuff = (struct shared_use_st *)shared_memory;
    while(running)
    {
        while(shared_stuff -> written_by_you != 1){}
        n = shared_stuff -> nn;
        printf("Initial Array:\n");
```

```
    for(i=0;i<n;i++)
    {
        printf("%d ",shared_stuff -> array[i]);
    }
    for(i=0;i<n;i++)
    {
        for(j=i+1;j<n;j++)
        {
            if((shared_stuff -> array[i]) > (shared_stuff ->array[j]))
            {
                temp = shared_stuff -> array[i];
                shared_stuff -> array[i] = shared_stuff ->array[j];
                shared_stuff -> array[j] = temp;
            }
        }
    }
    printf("\n\nSorted Array:\n");
    for(i=0;i<n;i++)
    {
        printf("%d ",shared_stuff -> array[i]);
    }
    printf("\n\n");
    shared_stuff -> written_by_you = 2;
    break;
}
if(shmdt(shared_memory) == -1)
{
    fprintf(stderr,"shmdt failed\n");
    exit(EXIT_FAILURE);
}
if(shmctl(shmid, IPC_RMID, 0) == -1)
{
  fprintf(stderr,"shmctl(IPC_RMID) Failed\n");
```

```
      exit(EXIT_FAILURE);
    }
    exit(EXIT_SUCCESS);
}
```

3.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/shm.h>
#include "shm_com.h"
int bs(int arr[],int n,int x)
{
      int l=0,r=n,m;
      while(l<=r)
      {
        m=(l+r)/2;
        if(arr[m]==x)
            return 0;
        else if(x< arr[m])
          r=m-1;
        else
          l=m+1;
      }
      return 1;
}

int main()
{
    int running = 1;
```

```
int n,m,i,j,temp,find;
void *shared_memory = (void *)0;
struct shared_use_st *shared_stuff;
int shmid;
srand((unsigned int)getpid());
shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666|IPC_CREAT);
if(shmid == -1)
{
  fprintf(stderr,"shmget failed\n");
  exit(EXIT_FAILURE);
}
shared_memory = shmat(shmid,(void *)0,0);
if(shared_memory == (void *)-1)
{
  fprintf(stderr,"shmat failed\n"); exit(EXIT_FAILURE);
}
printf("Memory attached at %X\n",shared_memory);
shared_stuff = (struct shared_use_st *)shared_memory;
while(running)
{
    while(shared_stuff -> written_by_you != 2){}
    n = shared_stuff -> nn;
    printf("Given array:\n");
    for(i=0;i<n;i++)
      printf("%d ",shared_stuff -> array[i]);
    while(1)
    {
        printf("\nEnter the Element to be searched: ");
        scanf("%d",&m);
        if(m==-1)
          break;
        find = bs(shared_stuff -> array,n-1,m);
        if(find == 0)
```

```
            printf("Element %d is in the array",m);
          else
          printf("Element %d is NOT in the array",m);
        }
        shared_stuff -> written_by_you = 3; break;
    }
    if(shmdt(shared_memory) == -1)
    {
        fprintf(stderr,"shmdt failed\n");
        exit(EXIT_FAILURE);
    }
    if(shmctl(shmid, IPC_RMID, 0) == -1)
    {
        fprintf(stderr,"shmctl(IPC_RMID) Failed\n");
        exit(EXIT_FAILURE);
    }
    exit(EXIT_SUCCESS);
}
```

```
Shivanshus-MacBook-Pro:21 mohmaya$ ./1.o
Memory attached at CEDB000
Enter the Number of Elements:
5
Enter the Elements:
1 3 2 6 5
```

```
Shivanshus-MacBook-Pro:21 mohmaya$ ./2.o
Memory attached at 2A9E000
Initial Array:
1 3 2 6 5

Sorted Array:
1 2 3 5 6
```

23. Write a program in C to implement: Two separate producers produces individually a matrix, the consumer adds and multiply those and returns result back to each of them.

shm_com.h

```
struct shared_use_st{
  int written_by_you;
  int n;
  int arr1[40][40];
  int arr2[40][40];
  int add[40][40];
  int mul[40][40];
};
```

1.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/shm.h>
#include "shm_com.h"
int main() {
    int running = 1;
    int i,j,m;
    void *shared_memory = (void *)0;
    struct shared_use_st *shared_stuff;
    int shmid;
    srand((unsigned int)getpid());
    shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666|IPC_CREAT);
    if(shmid == -1)
    {
        fprintf(stderr,"shmget failed\n");
        exit(EXIT_FAILURE);
```

```
}
shared_memory = shmat(shmid,(void *)0,0);
if(shared_memory == (void *)-1)
{
    fprintf(stderr,"shmat failed\n");
    exit(EXIT_FAILURE);
}
printf("Memory attached at %X\n",shared_memory);
shared_stuff = (struct shared_use_st *)shared_memory;
shared_stuff -> written_by_you = 0;
while(running)
{
    printf("\nEnter the order of matrix: ");
    scanf("%d",&m);
    shared_stuff -> n = m;
    printf("\nEnter the elements in the matrix: \n");
    for(i=0;i<m;i++)
    {
      for(j=0;j<m;j++)
      {
        scanf("%d",&(shared_stuff -> arr1[i][j]));
      }
    }
    shared_stuff -> written_by_you = 1;
    while(shared_stuff -> written_by_you != 2){}
    printf("\nMatrix from Producer 2:\n");
    for(i=0;i<m;i++)
    {
      for(j=0;j<m;j++)
      {
        printf("%d",shared_stuff -> arr2[i][j]);
      }
    }
```

```
        while(shared_stuff -> written_by_you != 3){}
        printf("\nAdded Matrix:\n");
        for(i=0;i<m;i++)
        {
                for(j=0;j<m;j++)
                {
                  printf("%d\t",shared_stuff -> add[i][j]);
                  printf("\n");
                }
                printf("\n");
        }
        printf("\nMultiplied Matrix:\n");
        for(i=0;i<m;i++)
        {
                for(j=0;j<m;j++)
                {
                  printf("%d\t",shared_stuff -> mul[i][j]);
                  printf("\n");
                }
                printf("\n");
        }
    }
    if(shmdt(shared_memory) == -1)
    {
      fprintf(stderr,"shmdt failed\n");
      exit(EXIT_FAILURE);
    }
    exit(EXIT_SUCCESS);
}
```

2.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/shm.h>
#include "shm_com.h"
int main() {
    int running = 1;
    int i,j,m;
    void *shared_memory = (void *)0;
    struct shared_use_st *shared_stuff;
    int shmid;
    srand((unsigned int)getpid());
    shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666|IPC_CREAT);
    if(shmid == -1)
    {
        fprintf(stderr,"shmget failed\n");
        exit(EXIT_FAILURE);
    }
    shared_memory = shmat(shmid,(void *)0,0);
    if(shared_memory == (void *)-1)
    {
        fprintf(stderr,"shmat failed\n");
        exit(EXIT_FAILURE);
    }
    printf("Memory attached at %X\n",shared_memory);
    shared_stuff = (struct shared_use_st *)shared_memory;
    shared_stuff -> written_by_you = 0;
    while(running)
    {
        while(shared_stuff -> written_by_you != 1){}
```

```
m = shared_stuff -> n;
printf("\nMatrix from Producer 1:\n");
for(i=0;i<m;i++)
{
  for(j=0;j<m;j++)
  {
    printf("%d ",shared_stuff -> arr1[i][j]);
  }
  printf("\n");
}

printf("\nEnter the elements in the matrix: (Order =%d)\n",m);
for(i=0;i<m;i++)
{
  for(j=0;j<m;j++)
  {
    scanf("%d",&(shared_stuff -> arr2[i][j]));
  }
}
shared_stuff -> written_by_you = 2;
while(shared_stuff -> written_by_you != 3){}
printf("\nAdded Matrix:\n");
for(i=0;i<m;i++)
{
    for(j=0;j<m;j++)
    {
        printf("%d\t",shared_stuff -> add[i][j]);
    }
    printf("\n");
}
printf("\nMultiplied Matrix:\n");
for(i=0;i<m;i++)
{
```

```
        for(j=0;j<m;j++)
        {
          printf("%d\t",shared_stuff -> mul[i][j]);
        }
        printf("\n");
      }
      if(shmdt(shared_memory) == -1)
      {
          fprintf(stderr,"shmdt failed\n");
          exit(EXIT_FAILURE);
      }
      exit(EXIT_SUCCESS);
    }
}
```

3.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/shm.h>
#include "shm_com.h"
int main() {
    int running = 1 ,m,sum;
    int i,j,k;
    void *shared_memory = (void *)0;
    struct shared_use_st *shared_stuff;
    int shmid;
    srand((unsigned int)getpid());
    shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666 | IPC_CREAT);
    if(shmid == -1)
    {
        fprintf(stderr,"shmget failed\n");
        exit(EXIT_FAILURE);
    }
    shared_memory = shmat(shmid,(void *)0,0);
    if(shared_memory == (void *)-1)
    {
        fprintf(stderr,"shmat failed\n");
        exit(EXIT_FAILURE);
    }
    printf("Memory attached at %X\n",shared_memory);
    shared_stuff = (struct shared_use_st *)shared_memory;
    while(running)
    {
        while(shared_stuff -> written_by_you != 1){}
        m = shared_stuff -> n;
```

```
printf("\nMatrix from Producer 1:\n");
for(i=0;i<m;i++)
{
  for(j=0;j<m;j++)
  {
    printf("%d ",shared_stuff -> arr1[i][j]);
  }
  printf("\n");
}
while(shared_stuff -> written_by_you != 2){}
printf("\nMatrix from Producer 2:\n");
for(i=0;i<m;i++)
{
        for(j=0;j<m;j++)
        {
            printf("%d ",shared_stuff -> arr2[i][j]);
        }
        printf("\n");
}
for(i=0;i<m;i++)
{
         for(j=0;j<m;j++)
         {
            shared_stuff -> add[i][j]= shared_stuff -> arr1[i][j] + shared_stuff -> arr2[i]
[j];
            sum=0;
            for(k=0;k<m;k++)
            {
                sum+= (shared_stuff -> arr1[i][k])*(shared_stuff -> arr2[k][j]);
            }
            shared_stuff -> mul[i][j] = sum;
        }
        printf("\n");
}
```

```
        shared_stuff -> written_by_you = 3; break;
    }
    if(shmdt(shared_memory) == -1)
    {
        fprintf(stderr,"shmdt failed\n");
        exit(EXIT_FAILURE);
    }
    if(shmctl(shmid, IPC_RMID, 0) == -1)
    {
        fprintf(stderr,"shmctl(IPC_RMID) Failed\n");
        exit(EXIT_FAILURE);
    }
    exit(EXIT_SUCCESS);
}
```

```
Shivanshus-MacBook-Pro:22 mohmaya$ ./1.o
Memory attached at D22A000

Enter the order of matrix: 3

Enter the elements in the matrix:
1 2 3
4 5 6
7 8 9

Matrix from Producer 2:
987654321
Added Matrix:
10
10
10

Multiplied Matrix:
30
24
18
84
69
54
138
114
90
Shivanshus-MacBook-Pro:22 mohmaya$
```

```
Shivanshus-MacBook-Pro:22 mohmaya$ ./2.o
Memory attached at D693000

Matrix from Producer 1:
1 2 3
4 5 6
7 8 9

Enter the elements in the matrix: (Order =3)
9 8 7
6 5 4
3 2 1

Added Matrix:
10      10      10
10      10      10
10      10      10

Multiplied Matrix:
30      24      18
84      69      54
138     114     90
Shivanshus-MacBook-Pro:22 mohmaya$
```

```
Shivanshus-MacBook-Pro:22 mohmaya$ ./3.o
Memory attached at 9258000

Matrix from Producer 1:
1 2 3
4 5 6
7 8 9

Matrix from Producer 2:
9 8 7
6 5 4
3 2 1
```

23. Write a program in C to implement: A paragraph of text is entered using a producer program and then passed through a consumer which separates all the words depending on the length of the word and stores each identical length words in separate data files having name as 1.dat, 2.dat and so on. The consumer should then inform the producer that the data files have been generated and the producer prints the data files. Consider words up to five characters long.

1.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/shm.h>
#include "shm_com.h"

int main() {
    int running = 1,i;

    char c;
FILE *fp;

    char word[10];

    void *shared_memory = (void *)0;
    struct shared_use_st *shared_stuff;
    int shmid;
    srand((unsigned int)getpid());
    shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666 | IPC_CREAT);
    if(shmid == -1)
    {
        fprintf(stderr,"shmget failed\n");
        exit(EXIT_FAILURE);
    }
```

```
    shared_memory = shmat(shmid,(void *)0,0);
    if(shared_memory == (void *)-1)

    {
        fprintf(stderr,"shmat failed\n");
        exit(EXIT_FAILURE);
    }
    printf("Memory attached at %X\n",(int)shared_memory);
    shared_stuff = (struct shared_use_st *)shared_memory;
    shared_stuff -> written_by_you = 0;

while(running)
    {

      if(shared_stuff -> written_by_you == 2)
      {
        fp = fopen("1.txt","r");
        if(fp == NULL)
        {
            printf("Cannot open the file\n");
        }
        else {
            while(fscanf(fp,"%s",word) != EOF)
            {
              printf("%s\n",word);
            }
          }
        fclose(fp);
        fp = fopen("2.txt","r");
        if(fp == NULL)
        {
            printf("Cannot open the file\n");
        }
        else {
```

```
    while(fscanf(fp,"%s",word) != EOF)
    {
      printf("%s\n",word);
    }
  }
fclose(fp);
fp = fopen("3.txt","r");
if(fp == NULL)
{
    printf("Cannot open the file\n");
}
else {
    while(fscanf(fp,"%s",word) != EOF)
    {
      printf("%s\n",word);
    }
  }
fclose(fp);
fp = fopen("4.txt","r");
if(fp == NULL)
{
    printf("Cannot open the file\n");
}
else {
    while(fscanf(fp,"%s",word) != EOF)
    {
      printf("%s\n",word);
    }
  }
fclose(fp);
fp = fopen("5.txt","r");
if(fp == NULL)
{
```

```
            printf("Cannot open the file\n");
          }
          else {
            while(fscanf(fp,"%s",word) != EOF)
            {
              printf("%s\n",word);
            }
          }
          fclose(fp);

sleep(1);

shared_stuff -> written_by_you = 0;
      }

      if(shared_stuff -> written_by_you == 0)
      {
        c = 0;
        printf("Enter the text :\n");
        while(c != '\n')
        {
          scanf("%s",(shared_stuff -> text)[i]);
          c = getchar();

          i++;
        }

shared_stuff -> flag = i;
        shared_stuff -> written_by_you = 1;
      }
    }
    if(shmdt(shared_memory) == -1)
    {
      fprintf(stderr,"shmdt failed\n");
```

```
    exit(EXIT_FAILURE);
  }

  if(shmctl(shmid, IPC_RMID, 0) == -1)
  {
    fprintf(stderr,"shmctl(IPC_RMID) Failed\n");
    exit(EXIT_FAILURE);
  }
  exit(EXIT_SUCCESS);
}
```

2.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/shm.h>
#include "shm_com.h"
int main() {
  int running = 1,i,l;

  char *word;

FILE *fp;

  word = (char*)malloc(sizeof(char)*10);
  void *shared_memory = (void *)0;
  struct shared_use_st *shared_stuff;
  int shmid;
  srand((unsigned int)getpid());
  shmid = shmget((key_t)1234, sizeof(struct shared_use_st), 0666 | IPC_CREAT);
```

```
    if(shmid == -1)
    {
      fprintf(stderr,"shmget failed\n");
      exit(EXIT_FAILURE);
    }

    shared_memory = shmat(shmid,(void *)0,0);
    if(shared_memory == (void *)-1)

    {
      fprintf(stderr,"shmat failed\n");
      exit(EXIT_FAILURE);
    }

printf("Memory attached at %X\n",(int)shared_memory);
    shared_stuff = (struct shared_use_st *)shared_memory;
    while(running)

    {
      if(shared_stuff -> written_by_you == 1)
      {
        printf("Reading each word.....\n");
        for(i = 0; i < shared_stuff -> flag; i++)
        {
          printf("%s\n",(shared_stuff->text)[i]);
          l = strlen((shared_stuff -> text)[i]);
          switch(l)
{
                    case 1: fp = fopen("1.txt","a");
                            if(fp == NULL)
                            {
                              printf("Cannot open the file\n");
                            }

fprintf(fp,"%s",(shared_stuff -> text)[i]);
```

```
                fputc('\n',fp);
                fclose(fp);
                break;
        case 2: fp = fopen("2.txt","a");
                if(fp == NULL)
                {
                  printf("Cannot open the file\n");
                }
                fprintf(fp,"%s",(shared_stuff -> text)[i]);
                fputc('\n',fp);
                fclose(fp);
                break;
      case 3: fp = fopen("3.txt","a");
                if(fp == NULL)
                {
                  printf("Cannot open the file\n");
                }

fprintf(fp,"%s",(shared_stuff -> text)[i]);
                fputc('\n',fp);
                fclose(fp);
                break;
      case 4: fp = fopen("4.txt","a");
                if(fp == NULL)
                {
                  printf("Cannot open the file\n");
                }

fprintf(fp,"%s",(shared_stuff -> text)[i]);
                fputc('\n',fp);
                fclose(fp);
                break;
      case 5: fp = fopen("5.txt","a");
                if(fp == NULL)
```

```
                    {
                      printf("Cannot open the file\n");
                    }

fprintf(fp,"%s",(shared_stuff -> text)[i]);
                    fputc('\n',fp);
                    fclose(fp);
                    break;
          default : break;
        }
      }
      shared_stuff->written_by_you = 2;
    }
  }
  if(shmdt(shared_memory) == -1)
  {
    fprintf(stderr,"shmdt failed\n");
    exit(EXIT_FAILURE);
  }

  if(shmctl(shmid, IPC_RMID, 0) == -1)
  {
    fprintf(stderr,"shmctl(IPC_RMID) Failed\n");
    exit(EXIT_FAILURE);
  }
  exit(EXIT_SUCCESS);
}
```