INHA UNIVERSITY TASHKENT
DEPARTMENT OF CSE & ICE

FALL SEMESTER 2017

SOC 3010 - OPERATING SYSTEMS

HOME ASSIGNMENT 1

INSTRUCTIONS :

- **All Home assignments are to be completed in groups**
- **Screen shots are to be provided wherever necessary**
- **Home Assignment Report should be prepared using the Template provided**
- **One Hard Copy of the Home Assignment of each group should be handed in at the office by the Group Leader.**
- **Every member of the team must upload the softcopy of the report at the E-Class portal**
- **Last date for submission of the Home Assignment is 10th October 2017**
- **Late submissions are not entertained, Adhere to the deadline strictly**

QUESTIONS :

PART 1 : PRACTICE QUESTIONS

A. UNDERSTANDING UNIX/LINUX COMMANDS

Describe the purpose of the following Unix/Linux commands providing correct syntax. Try out these commands in default bash shell using the correct format and write the results/output obtained and provide the screen shots.

1) ls, ls -la	2) pwd	3) cd	4) cat	5) cp	6) mv
7) who	8) whoami	9) ps, ps -la	10) more	11) less	12) head
13) tail	14) fg	15) rm	16) mkdir	17) rmdir	18) date
19) sudo	20) apt-get	21) chmod	22) echo	23) find	24) free
25) diff	26) grep	27) passwd	28) wc	29) man	30) sort
31) cmp	32) bg	33) file	34) time	35) kill	36) link
37) uname	38) df	39) du	40) chown	41) chgrp	42) adduser
43) addgroup	44) deluser	45) delgroup	46) touch	47) top	48) cut
49) sed	50) uptime				

B. PERFORMING FILE OPERATIONS USING UNIX I/O SYSTEM CALLS

Using gedit or any other Editor, create the following file in your user directory :

```
/* Program using UNIX I/O primitives to perform file operations */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>

#define DEF_MODE S_IRUSR|S_IWUSR|S_IXUSR|S_IRGRP|S_IWGRP|S_IXGRP|S_IROTH

int main()
{
int fd, fd1, len, i, fsize, nbytes;
char ch=0, cf, buf[512], fname[25], cname[25];

printf("CREATING A NEW FILE WITH ALL ACCESS RIGHTS TO USER AND GROUP AND
NO EXECUTE ACCESS TO OTHERS\n");
printf("ENTER FILE NAME : ");
scanf("%s", fname);
fd = open(fname, O_CREAT|O_TRUNC|O_WRONLY|O_APPEND, DEF_MODE);
if (fd < 0)
  printf ("cannot create FILE %s \n", fname);
else
{
printf("NOW ENTER YOUR PROGRAM OR TEXT LINE BY LINE- ONCE YOU FINISH
PRESS KEYS Ctrl D together\n");
i=0;
ch=getchar();        /* to remove the last newline character entered*/
while((ch=getchar()) != EOF)
     buf[i++]=ch;
fsize=i-2;
buf[fsize]='\0';
printf("Total characters stored in your file = %d\n", fsize);
write(fd, buf, fsize);
close(fd);
}

printf("OPENING AN EXISTING  FILE\n");
printf("ENTER FILE NAME : ");
scanf("%s", fname);
fd = open(fname, O_RDWR|O_APPEND, DEF_MODE);
if (fd < 0)
  printf ("cannot open FILE %s - does not exist \n", fname);
else
{
printf(" READING YOUR FILE CONTENTS\n");
nbytes=read(fd, buf, sizeof(buf));
```

```
len=strlen(buf);
printf("CONTENTS OF YOUR FILE %s - size= %d\n",fname, len);
puts(buf);
close(fd);
}

printf("COPYING A FILE \n");
printf("ENTER NAME OF CURRENT FILE  TO BE COPIED FROM :");
scanf("%s", fname);
printf("ENTER NAME OF NEW FILE NAME TO BE COPIED TO :");
scanf("%s", cname);
fd = open(fname, O_RDONLY, DEF_MODE);
if (fd < 0)
  printf ("cannot open FILE %s - does not exist \n", fname);
else
 {
   if(nbytes=read(fd, buf, sizeof(buf)) <0)
      printf("FILE READ ERROR\n");
   else
   {
      fd1 = open(cname, O_CREAT|O_TRUNC|O_WRONLY, DEF_MODE);
      if (fd1 < 0)
        printf("Cannot create New file %s\n", cname);
      else
       {
         len=strlen(buf);
         if( nbytes=write(fd1, buf, len) < 0)
           printf("FILE WRITE ERROR\n");
         else
          printf("FILE %s  has been copied to %s successfully OK ......\n",fname,cname);
         close(fd1);
       }
       close(fd);
     }
   }

printf("RENAMING  A FILE \n");
printf("ENTER CURRENT FILE NAME :");
scanf("%s", fname);
printf("ENTER NEW FILE NAME :");
scanf("%s", cname);
printf("FILE %s  has been renamed to %s OK......\n", fname, cname);
rename(fname, cname);

printf("DELETING A FILE \n");
printf("ENTER FILE NAME :");
scanf("%s", fname);
ch=getchar();  /* to remove the last newline character entered*/
printf("PLEASE CONFIRM -SURE YOU WANT TO DELETE ....PRESS y/n :");
scanf("%c", &cf);
if (cf == 'y')
{
  unlink(fname);
  printf("FILE %s deleted OK......\n", fname);
}
else
printf("FILE %s not deleted OK......\n", fname);


}
```

Compile the above program using gcc and execute the program. Write the file operations performed by this program with suitable commentsand UNIX I/O primitives used. Provide the results and also the screenshots.

C. BASH SHELL PROGRAMMING

Using gedit or any other Editor, create the following shell scripts in your user directory, make them executable and run the shell scripts. Write the results/outputs obtained and provide the screen shots.

a) Sample Shell script

Shell script name : **sample.sh**

```
#!/bin/bash
echo Hi, What is your name
read NAME
echo Good Morning $NAME
echo ==================================================
echo Your Home directory : $PWD
echo Enviroment variable PATH = $PATH
echo PRESS Enter KEY
read key
echo ==================================================
echo HARD DISK FREE SPACE INFO
free
echo PRESS Enter KEY
read key
echo ==================================================
echo PROCESSES CURRENTLY RUNNING
ps -la
echo PRESS Enter KEY
read key
echo ===================================================
echo LINUX VERSION
uname -a
echo PRESS Enter KEY
read key
echo ===================================================
echo WHO ARE CURRENTLY LOGGED IN
who -a
echo PRESS Enter KEY
read key
```

```
echo ===================================================
read -p  'Login name:' LOGIN
read -sp  'Password:' PASS
echo \n
echo Your LOGIN NAME IS $LOGIN
echo Your PASSWORD IS $PASS
```

b) File – Line, word and bye count

Shell script name : **fcount.sh**

```
#!/bin/bash
 echo ===================================================
 echo Enter a filename to find number of characters, words and lines
 read FNAME
 cat $FNAME
 echo No of lines
 wc -l $FNAME
 echo No of words
 wc -w $FNAME
 echo No of bytes
 wc -c $FNAME
 echo PRESS Enter KEY
 read key
```

c) File Sorting

Shell script name : **fsort.sh**

```
#!/bin/bash
 echo ================================================================
 echo ENTER A FILE NAME TO STORE 10 NAMES
 read FILE
 echo ENTER 10 NAMES  ONE BELOW THE OTHER ON SEPARAT LINEs and THEN
 PRESS ^D
 cat >> $FILE
 echo PRESS Enter KEY
 read key
 echo ================================================================
 echo your FILE $FILE CONTENTS
 cat $FILE
 echo PRESS Enter KEY
 read key
 echo ================================================================
 echo FILE SORTING IN ALPHABETICAL ORDER
 sort $FILE
 echo PRESS Enter KEY
 read key
 echo ================================================================
 echo FILE SORTING IN REVERSE ALPHABETICAL ORDER
 sort -r $FILE
 echo PRESS Enter KEY
 read key
```

d) Examples for Conditional construct if …then….else …fi

Shell script name : **flist.sh**

```
#!/bin/bash
echo =======================================================
echo Do you want directory listing long or short \( enter 1 or 0 \)
read LONG
if [ $LONG -eq 1 ]
then
  ls -la
else
  ls
fi
echo PRESS Enter KEY
read key
```

e) Examples for Loop Constructs

i) while do ….. done construct

Shell script name : **sumodd.sh**

```
#!/bin/bash
   echo ==============================================
   A=1
   B=0
   C=1
   echo Computing SUM OF FIRST n ODD INTEGERS
   echo ENTER VALUE OF n=
   read n
   while [ $A -le $n ]
   do
     B=$(expr $B + $C)
     A=$(expr $A + 1)
     C=$(expr $C + 2)
   done
   echo Sum of FIRST $n odd numbers = $B
   echo PRESS Enter KEY
   read key
```

ii) for do …..done construct

Shell script name : **cfiles.sh**

```
#!/bin/bash
echo  Files in your HOME Directory with extension .c
for FILE in $HOME/*.c
do
  echo $FILE
done
```

Shell script name : **sum100.sh**

```
#!/bin/bash
echo COMPUTING SUM OF FIRST 100 INTEGERS
sum=0
for i in {1..100}
do
  sum=$sum+$i
done
echo SUM OF FIRST 100 INTEGERS = $sum
```

iii) until do….. done construct

Shell script name : **fact.sh**

```
#!/bin/bash
echo ==================================================
A=1
B=1
echo Computing FACTORIAL OF n
echo ENTER VALUE OF n=
read n
until [ $A -gt $n ]
do
  B=$(expr $B \*  $A)
  A=$(expr $A + 1)
done
echo FACTORIAL OF $n = $B
echo PRESS Enter KEY
read key
```

f) Example for Multi-way Branch - case …..esac

Shell script name : **case.sh**

```
#!/bin/bash
#Menu Driven program to perform listed operations
loop=1
while [ $loop -eq 1 ]
do
echo ........................................
echo .          MENU                    .
echo .                                  .
echo .  1. LIST DIRECTORY CONTENTS          .
echo .  2. SHOW CURRENT WORKING DIRECTORY   .
echo .  3. DISPLAY LINUX VERSION            .
echo .  4. SHOW FREE SPACE ON DISK          .
echo .  5. SHOW WHO ARE LGGED IN            .
echo .  6. DISPLAY CONTENTS OF A FILE       .
echo .  7. CREATE OR OPEN A FILE            .
echo .  8. COPY A FILE                      .
echo .  9. RENAME A FILE                    .
echo .  10. REMOVE A FILE                   .
echo .  11. QUIT                            .
echo ........................................
echo  ENTER YOUR CHOICE :
read CH
case $CH in
1)
    ls
    ;;
2)
    pwd
    ;;
3)
    uname -r
    ;;
```

```
4)
     free
     ;;
5)
     who -a
     ;;
6)
     echo Enter File Name :
     read FILE
     cat $FILE
     ;;
7)
     echo Enter File Name :
     read FILE
     gedit $FILE
     ;;
8)
      echo Enter Name of the File to be copied from :
      read FILE1
      echo Enter Name of the File to be copied to :
      read FILE2
      cp $FILE1 $FILE2
      ;;
9)
       echo Enter OLD File Name:
       read FILE1
       echo Enter NEW File Name:
       read FILE2
       mv $FILE1 $FILE2
       ;;
10)
       echo Enter Name of the File to delete:
       read FILE
       echo ARE YOU SURE - CONFIRM 1 OR 0
       read OK
       if [ $OK -eq 1 ]
       then
       rm $FILE
       echo File $FILE  deleted.....OK
       else
       echo File $FILE not deleted.....OK
       fi
       ;;
11)
        echo QUITING.......GOOD BYE
        break
        ;;
*)
        echo INVALID CHOICE - READ MENU CORRECTLY
        ;;
esac
done
```

g) Example for Select do ….done and if …elif ….fi constructs

Shell script name : **select.sh**

```
#!/bin/bash
#A simple Menu System
OPTIONS='ls longls who free linuxversion cat mv cp QUIT'
PS3='Choose an option:'
select CHOICE  in $OPTIONS
do
if [ $CHOICE == ls ]
then
   ls
elif [ $CHOICE == longls ]
then
   ls -la
  elif [ $CHOICE == who ]
  then
     who -a
  elif [ $CHOICE == free ]
  then
     free
   elif [ $CHOICE == linuxversion ]
   then
      uname -a
   elif [ $CHOICE == cat ]
   then
       echo ENTER FILE NAME :
       read FILE
       cat $FILE
    elif [ $CHOICE == mv ]
    then
        echo ENTER OLD FILE NAME :
        read FILE1
        echo ENTER NEW FILE NAME :
        read FILE2
        mv $FILE1 $FILE2
     elif [ $CHOICE == cp ]
     then
         echo ENTER NAME OF FILE TO BE COPIED FROM:
         read FILE1
         echo ENTER NAME OF FILE TO BE COPIED TO :
         read FILE2
         cp $FILE1 $FILE2
     elif [ $CHOICE == QUIT ]
     then
             echo BYE ... BYE
             break
fi
done
```

h) Passing parameters using Command Line Arguments

Shell script name : **copy.sh**

usage syntax : copy file1 file2

```
#!/bin/bash
echo Copying file $1 to $2
cp $1 $2
echo file $1 CONTENTS
cat $1
echo file $2 CONTENTS
cat $2
```

i) Length of a string

Shell script name : **len.sh**

```
#!/bin/bash
#Show the length of a String
echo Enter a string :
read STR
echo  LENGTH OF THE string : ${#STR}
```

j) Use of Special Variables

Shell script name : **var.sh**

```
#!/bin/bash
echo ================================================
echo Name of the Bash script - $0
echo How many arguments were passed to the Bash script - $#
echo All the arguments supplied to the bash script - $@
echo The exit status of the most recently run process  $?
echo The process ID of the current script - $$
echo User Name of the user running the script - $USER
echo The hostname of the machine the script is running on - $HOSTNAME
echo The number of seconds since the script was started - $SECONDS
echo Current line number in the Bash script : $LINENO
echo Random number returned by the RANDOM variable - $RANDOM
echo VALUES OF ALL ENVIRONMENT VARIABLES SET UP IN THE CURRENT
  ENVIRONMENT
env | more
echo ================================================
```

k) Using Arrays

(i) Shell script name : **array.sh**

```
#!/bin/bash
list=(12 67 123 49 88 123 -9 0 456 126)
let I=0
while [ $I -le 9 ]
do
echo ${list[$I]}
let I=$I+1
done
```

(ii) Shell script name : max**.sh**

```
#!/bin/bash
list=(12 67 123 49 88 123 -9 0 456 126)
```

```
let I=0
max=${list[0]}
let I=+$I
while [ $I -le 9 ]
do
if [ ${list[$I]} -gt $max ]
   then
    max=${list[$I]}
fi
let I=$I+1
done
echo Maximum element in the list is : $max
```

(iii) Shell script name : maxlist**.sh**

```
#!/bin/bash
echo Enter size of the list :
read N
echo Enter a list of $N numbers :
let K=0
while [ $K -lt $N ]
do
read VAL
list[$K]=$VAL
let K=$K+1
done
let I=0
max=${list[0]}
let I=+$I
while [ $I -lt $N ]
do
if [ ${list[$I]} -gt $max ]
   then
    max=${list[$I]}
fi
let I=$I+1
done
echo Maximum element in the list is : $max
```

l) Procedure Invocation

```
#!/bin/bash
len_str() {
     #Show the length of a String
     echo Enter a string :
     read STR
      L=${#STR}
       return $L
}
 print_str(){
  len=$?
  echo Length of the string  $1 is $len
}
len_str
print_str  $STR
```

PART II : ACTIVITIES

INSTRUCTIONS :

PART II QUESTIONS SHOULD BE ATTEMPTED ONLY AFTER COMPLETING ALL THE PART-I QUESTIONS. YOU NEED ALL THE FILES CREATED IN PART I FOR ANSWERING PART II QUESTIONS. QUESTIONS IN PART II MUST BE ATTEMPTED ONE AFTER THE OTHER AS THE COMMANDS DEPEND ON THE PREVIOUS OPERATIONS.

a) **State the Purpose of these commands and Indicate the outputs generated when these commands are executed. Execute the commands one after another in the same given order 1) to 230). Clearly explain the format of the output with all the fields and their meaning. Provide all the screen shots.**

1) PS1=”\d \t \u@\h \#\$”
2) ls -lt
3) ls -lrt
4) ls --help
5) cd ~
6) cat .profile
7) uname –r
8) uname –a
9) uname -n
10) uname -o
11) uname -p
12) arch
13) touch nfact.c
14) ls –l nfact.c
15) chmod +x nfact.c
16) ls –l nfact.c
17) cat nfact.c
18) history
19) history 25
20) df
21) df –h
22) top
23) ps –la
24) ps –ef
25) ps –ejH
26) cat /proc/cpuinfo | more
27) cat /proc/cpuinfo > infocpu
28) grep –e “processor” –e “cpu cores” –e “model name” –e “cpu MHz” –e “cache size” infocpu
29) grep –c “processor” infocpu
30) grep –c “cpu” infocpu
31) lsof | tee proc_usefile
32) head -1 infocpu
33) head -5 infocpu
34) head -24 infocpu > core0info
35) tail -1 infocpu

```
36) tail  -5 infocpu
37) tail -24 infocpu > core4info
38) cat core4info
39) cut  -b1-25 core4info
40) sed s/cpu/CPU/g core0info
41) ls –la |  listdir
42) cut  -b30-60 listdir
43) history | tail -20
44) uptime
45) du –h
46) du –sh
47) du –sh *  | sort –n
48) du –sh  * | sort –r
49) gzip infocpu
50) ls –l info*
51) gunzip infocpu.gz
52) ls –l info*
53) bzip2 infocpu
54) ls –la  info*
55) bunzip2 cpuinfo.bz2
56) cat infocpu
57) cat –n infocpu
58) nl infocpu
59) pr –n infocpu
60) grep –o “Unix” | wc –l
61) df –kl
62) df -kh
63) echo $PATH
64) a=$(expr 15 + 10)
65) echo $a
66) let b=75
67) echo $b
68) let c=$a+$b
69) echo $c
70)  D=$((15+25))
71)  echo $D
72) echo $HOME
73) find –name  linux
74) ls > list
75) cat list
76) rev list
77) cat list
78) echo “HELLO GOOD MORNING! HOW ARE YOU” | rev
79) ln list list1
80) ls  -l   lis*
81) ln  -s   list   list2
82) PS1=”LS$”
83) ls  -l lis*
84) ls   -lrt  | grep “^l”
85) cat list1
86) cat list2
87) file list
```

```
88) file list1
89) file list2
90) file Desktop
91) file *.o
92) file *.c
93) file *.sh
94) file *
95) cat core4info
96) head -14  core4info | tail -1
97) sed  –i  ‘1 d’  core4info
98) cat core4info
99) sed  -i  ‘$ d’ core4info
100)    cat core4info
101)    sed –n ’13 p’ core4info
102)    sed –n ’13 p’ core4info | wc –c
103)    sed –n ’13 p’ core4info | wc –w
104)    sed –n ’13 p’ core4info | rev
105)    sed –n ’13,18 p’ core4info
106)    sed –n ’13,18 p’ core4info | wc -l
107)    sed –n ’13,18 p’ core4info | wc -w
108)    sed –n ’13,18 p’ core4info | wc -c
109)    sed –n ‘$ p’ core4info
110)    echo “GNU is Not Unix” | rev |  cut –f1 –d’ ‘  | rev
111)    echo “GNU is Not Unix” | cut –f4 –d’ ‘
112)    cut –c4 core4info
113)    cut –c4,6 core4info
114)    cut –c1-7 core4info
115)    cut –c-6  core4info
116)    cut –c10- file.txt
117)    cut  -c- core4info
118)    cut  –f2  –d’ ‘ core4info
119)    cut  -f1-2 –d’ ‘ core4info
120)    cut –d’ ‘  -f2  core4info
121)    cut  -d’ ‘  -f2,3  core4info
122)    cut  -d’ ‘ –f1-3  core4info
123)    cut  -d’ ‘  -f-3   core4info
124)    cut  -d’ ‘ –f2-  core4info
125)    cut  -d’:’  -f1  /etc/passwd
126)    cut  -d’:’  -f2,4   /etc/passwd
127)    cp  /etc/passwd  password
128)    cat password
129)    cut -d’ ‘ -f1 password
130)    cut -d’ ‘ -f1  password |  cut -c1-10
131)    rev  password  | cut –d’ ‘ –f1
132)    rev  password  | cut –d’ ‘ –f1  | cut -c10-50
133)    grep “CHOICE”  select.sh
134)    grep “echo” case.sh
135)    grep “^[a-z]”  select.sh
136)    grep “^[A-Z]”  select.sh
137)    grep “^[0-9]”  select.sh
138)    grep “^[  ]” select.sh
139)    grep  –r  -x  “esac”  *
```

140) grep –L “esac” *
141) grep –l “file” *
142) grep –o “esac” *
143) grep –o “esac” *.*
144) grep –o “case” *.*
145) grep -n “case” *.*
146) grep –o “esac” case.sh
147) grep –n “esac” case.h
148) grep –n “case” case.sh
149) grep –o “case” case.sh
150) grep –o “echo” case.sh
151) grep –o “echo” case.sh | wc -l
152) grep –o –b “case” case.sh
153) grep –o –b –n “case” case.h
154) grep –b –n “case” case.h
155) grep -b “echo” case.sh
156) grep –o –b –n “echo” case.h
157) grep –b –n “echo” case.h
158) grep -b “echo” case.sh
159) grep –n “echo” case.h
160) grep “^case” case.sh
161) grep “OK$” case.sh
162) grep “ne$” case.sh
163) grep “:$” case.sh
164) grep “^while” case.sh
165) grep “^echo” case.sh
166) grep -v “echo” case.sh
167) grep -v “^echo” case.sh
168) grep -v “^while” case.sh
169) grep -c “case” case.sh
170) grep -c “echo” case.sh
171) grep –c “unix” file.txt
172) grep “read” case.sh
173) grep -ow “read” case.sh
174) grep -ow “read” case.sh | wc –l
175) cat >> part2q
type this text at the console each on separate lines including this line
your PART2 question - State the Purpose of these commands and Indicate the outputs generated when these commands are executed. Execute the commands one after another in the same given order 1) to 200). Clearly explain the format of the output with all the fields and their meaning.
Provide all the screen shots.
176) sed –n ‘3 p’ part2q
177) sed –n ‘3 p’ part2q | grep – o “the”
178) sed –n ‘3 p’ part2q | grep – o “the” | wc –l
179) sed –-n ‘1 p’ part2q
180) sed –-n ‘1 p’ part2q | grep –o “the” | wc –l
181) sed –-n ‘1,4 p’ part2q | grep –o “the” | wc –l
182) sed ‘s/the/THE/’ part2q
183) sed ‘s/THE/the/g’ part2q
184) sed ‘s/the/THE/2’ part2q
185) sed ‘s/THE /the/3’ part2q

186) sed ‘s/the/{&}/’ part2q
187) sed ‘s/the/THE/p’ part2q
188) sed –n ‘s/THE/the/p’ part2q
189) sed ‘s/the/THE/’ part2q | sed ‘s/commands/instructions/’
190) sed -e ‘s/THE/the/’ -e ‘s/commands/instructions/’ part2q
191) sed ‘s/instructions/commands/’ part2q
192) sed ‘4 s/the/THE/’ part2q
193) sed ‘1,3 s/the/THE/’ part2q
194) sed ‘2,$ s/THE/the/’ part2q
195) sed ‘/commands/ s/the/THE/’ part2q
196) cp part2q part3q
197) sed ‘3 d’ part3q
198) sed ‘p’ part3q
199) sed ‘2,$ d’ part3q
200) cat part3q
201) sed –n ‘/commands/ p’ part2q
202) grep ‘commands‘ part2q
203) grep –v ‘commands’ part2q
204) grep – w “commands” part2q
205) grep -B 2 “State” part2q
206) grep -A 2 “State” part2q
207) grep -C 2 “State” part2q
208) rep -B 2 “and” part2q
209) grep -A 2 “and” part2q
210) grep -C 2 “and” part2q
211) sed –n ‘/commands/ !p’ part2q
212) sed ‘/commands/ a “You need to learn these commands which are very essential” ‘ part2q
213) cal
214) cal -3
215) cal –y
216) passwd
217) whoami
218) users
219) mail –s “Test Mail” youremailid@inha.uz (Press Enter key – You get Cc: here enter the other address you wan to cc and the enter and hellnow type your message and after completing the message Press Ctrl D)
220) mail –s “Test Mail” youremailid@inha.uz < message.txt (assuming that your message is stored in a file message.txt)
221) mail
222) ping inha.uz (to come out of this command press Ctrl C)
223) ping inha.kr (to come out of this command press Ctrl C)
224) ping google.com
225) ping amazon.com
226) whereis linux
227) which case.sh
228) mount
229) vmstat
230) netstat

b) Answer the following questions

1. Write a linux command to list all the symbolic links in the current working directory
2. Write a linux command to create an empty file fileop.c in your directory (without using any editor)
3. Give a linux command to make the file fileop.c only readable for the user and no access rights for group and owner
4. How will you find which operating system your system is running on in linux?
5. How will you run a process in background? How will you bring that into foreground and how will you kill that process?
6. How do you know if a remote host is alive or not?
7. Give a linux command to know all the earlier commands entered by you at the terminal.
8. How do you find which process is taking how much CPU?
9. How do you check how much space left in current drive ?
10. What is the difference between ps -ef and ps –a?
11. How do you find how many cpu are there in your system and their details?
12. How do you find zombie process in linux?
13. There is a file somewhere in your system which has the name part2q. Give a linux command to find that file.
14. How do you find whether your system is 32 bit or 64 bit ?
15. How do you find which processes are using a particular file?
16. There is a file part2q which contains words **the**, Give a linux command to replace all the occurrences of **the** in the file with THE?
17. You have listdir file created in your directory (using PARTII a) question). Give a linux command to display second column valuses from the fiile.
18. Your application home directory is full? How will you find which directory is taking how much space?
19. How do you find for how long your System is up?
20. Give a linux command to reverse all lines in your file listdir
21. Give a linux command to display processes sorted on CPU usage
22. How do you get (display) only first 10 lines from file case.sh
23. How do you get (display) last 5 lines from file case.sh
24. Give linux commands to compress and decompress files in your directory. Give an example in each case taking a large file.
25. Give a linux command to display line numbers along with each line of text in a file
26. Give a linux command to assign execute rights to all for the file case.sh
27. What does rm –r * do?
28. What is the behavioral difference between cmp and diff commands. Take an example and illustrate.
29. Give two commands in linux to display first line of a file.
30. Give two commands in linux to display last line of a file.
31. What is the command to find maximum memory taking process on your system?
32. What is the command to find hidden files in your current directory?
33. What is the command to find currently running process in your system?
34. What is the command to find remaining disk space on your system?
35. What is the command to count only the number of lines in the file listdir
36. What is the command to count only the number of words in the file listdir
37. What is the command to count only the number of characters in the file listdir
38. Is there a way to erase all files in the current directory, including all its sub-directories, using only one command?
39. What is the use of the tee command? Explain with an example
40. What Linux operating system command would you use to display the shell's environment variables?

41. What is the difference between the commands
    ls –la > file1 and ls –la > file1 2>&1
42. How do you find the length of 8th line in your file listdir
43. Give a command to delete first line from your file listdir
44. Give a command to delete last line from your file listdir
45. Give a command to get 5th word in a line in file part2q
46. Give a command to get first word in a line in file part2q
47. Give a command to get last word from a line in file part2q
48. Give a command in linux to reverse a string "WINE Is Not windows Emulator"
49. Give a command to delete lines from 1 to 5 in file listdir
50. Give a command to delete lines from 15 to the end of the file listdir

c) Write a shell script to replace the 5th line in your file listdir with a new line "Hi, this new line has been inserted here"

d) Linux shell provides following file test operators that can be used to test various properties associated with a linux file.

Operator	Description	Example
-b file	Checks if file is a block special file; if yes, then the condition becomes true.	[-b $file] is false.
-c file	Checks if file is a character special file; if yes, then the condition becomes true.	[-c $file] is false.
-d file	Checks if file is a directory; if yes, then the condition becomes true.	[-d $file] is not true.
-f file	Checks if file is an ordinary file as opposed to a directory or special file; if yes, then the condition becomes true.	[-f $file] is true.
-p file	Checks if file is a named pipe; if yes, then the condition becomes true.	[-p $file] is false.
-r file	Checks if file is readable; if yes, then the condition becomes true.	[-r $file] is true.
-w file	Checks if file is writable; if yes, then the condition becomes true.	[-w $file] is true.
-x file	Checks if file is executable; if yes, then the condition becomes true.	[-x $file] is true.
-s file	Checks if file has size greater than 0; if yes, then condition becomes true.	[-s $file] is true.
-e file	Checks if file exists; is true even if file is a directory but exists.	[-e $file] is true.

Write a Shell script to check whether a file exists in your home directory and if it exists then check whether it is a regular file or empty file, directory or symbolic link or block or character device file or named pipe. Also test whether the file is readable, writeable and executable.

e) Write a shell script to read size of the list **n** and list of n numbers from the keyboard and store the elements in an array **list** and then perform the following operations :
   (i) find and print the maximum element in the list
   (ii) find and print minimum element in the list
   (iii) Search for a given key (key is to be read from the keyboard) in the list – if the given key is found in the list then display the message KEY FOUND and print its position in the list, otherwise print NOT FOUND

f) Write shell scripts to perform the following operations on the file **part2q**:
   (i) substitute all lower case alphabets with uppercase and the store the output in file **part2ql**
   (ii) substitute all upper case alphabets with lowercase and store the output in file **part2qu**
   (iii) Replace all the occurrences of commands with COMMANDS and replace all words **"the'** in the lines containing word **"commands"** to **"THE"** and store the output in file **part2qr**
   (iv) First copy the contents of file **part2q to part2m**. Now delete first and last lines of the file **part2m**

(v) copy the contents of the files part2q, part2ql, part2qu, part2qr and part2m to one single file **part2qa (**file concatenation operation)

g) Unix/Linux provides several commands to perform shutdown, poweroff and reboot as shown below. Write a bash shell script to perform shutdown by providing the following menu driven screen to the user to select an option to perform the selected operation.

Halt - Brings the system down immediately

init 0 - Powers off the system using predefined scripts to synchronize and clean up the system prior to shutting down

init 6 - Reboots the system by shutting it down completely and then restarting it

Poweroff - Shuts down the system by powering off

Reboot - Reboots the system

Shutdown - Shuts down the system

h) Write a Menu driven 'C" program to perform the following file operations using Unix I/O primitives.

1. **Create a new file** by taking the filename from the user and assigning read, write and execute to user and only read access to group and owner. If file already exists then truncate its contents. Provide appropriate error message if there is any problem in file creation.
2. **Open an existing file** by taking the filename from the user and prompting the user to enter whether file to be opened in Read only mode, Write only mode with Append or Read & Write only mode with Append. Provide appropriate error message if there is any problem in opening a file or filename entered by the user doesn't exist.
3. **Read from a file** by asking the user to specify the number of bytes to be read from the opened file or entire contents of the file (till EOF). Then display the specified contents of the file read.
4. **Write to a file** by prompting the user to enter the text (512 bytes or more – several lines or program or script) to be written to the file from the keyboard and ask him to press Ctrl D once the user completes entering the text.
5. **Seek the contents of an opened file** by prompting the user to provide the origin(from the beginning or current position or end of the file) and how many bytes to be skipped(offset). After seeking to required position in the file, prompt the user whether user wants to read or write. If read option is selected, ask the user how many bytes the user wants to read and then display the requested content. If write option is selected then ask the user to enter the text to be written into the file from the keyboard ending with Cntrl D, write to the file starting from the specified position and then display the contents of entire file contents from the beginning.
6. **Delete a file** by prompting the user to enter the name of the file to be deleted and also further taking confirmation from the user willingness to delete. Only after receiving the confirmation, delete the file.
7. **Rename a file** by asking the user to enter the old(existing) file name and the new file name.
8. **Copy a file** by prompting the user to enter the name of the file to be copied from and name of the file to be copied to.
9. **Quit** – signing off from the menu driven screen by closing all the opened files

PART III : LINUX KERNEL SOURCE CODE INSTALLATION

QUESTIONS :

a) Perform installation of Linux Kernel source code into the user directory – Describe clearly all the steps in the installation of kernel source code providing all the required Linux commands and indicating clearly the main directory where the Linux kernel source code is installed. Provide the screen shots of all the steps involved in Linux kernel source code installation.

b) Understand the organization of the Linux kernel source directory tree clearly by going through the entire directory tree visiting all the sub-directories and provide the complete kernel directory tree (in the form of figure) and list out all the source code files in each of the sub-directories. Provide all the screenshots.

PART IV : LINUX KERNEL SOURCE CODE COMPILATION

QUESTIONS : (PART -III MUST BE COMPLETED BEFORE ATTEMPTING PART IV)

a) Compile and Build Linux Kernel from the source code installed.
b) Describe clearly all the steps in the compilation and building of Linux Kernel from the kernel source files providing all the required steps and indicating Linux commands used.
c) Test the new kernel compiled by rebooting with the new built kernel.

****