Unix Shell Scripts

1a)Non recursive script, which prints reversed order of args.

```
Echo “arguments in command prompt”
while [ $# -ne 0 ]
do
temp= “$1 $temp”
shift
done
echo “arguments in reverse order:”
echo “$temp”
```

output

```
[user@localhost unix2]$ vi 1a.sh

[user@localhost unix2]$ sh 1a.sh what is your name
name
your
is
what
```

1b) c program to create child process to read command from standard input and execute them

```
#include<unistd.h>
#include<string.h>
#include<stdio.h>
#include<sys/types.h>
#define maxline 20

int main()
{
        pid_t pid;
        int status;
        char buf[maxline];
        pid=fork();
        if(pid==0)
        {
                printf("Enter a valid UNIX command\n");
                if(fgets(buf,maxline,stdin)!=NULL)
                     buf[strlen(buf)-1]=0;
                        system(buf);
        }
        pid=waitpid(pid,&status,0);

}
```

output:

```
[user@localhost unix2]$ vi 1b.c
[user@localhost unix2]$ cc -o x.out 1b.c
[user@localhost unix2]$ ./x.out
Enter a valid UNIX command
```

```
ps
  PID TTY          TIME CMD
 2147 pts/0    00:00:00 bash
 2459 pts/0    00:00:00 x.out
 2460 pts/0    00:00:00 x.out
 2461 pts/0    00:00:00 ps
```

2a) c program to create file with 16 bytes of ordinary data rom the beginning and other 16 bytes of ordinary data from an offset of 48

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
int main()
{
        char s1[16]="0123456789012345";
        char s2[16]="abcdefghijklmnop";
        int fp;
        fp=creat("a.dat",0);
        write(fp,s1,16);
        lseek(fp,48,SEEK_SET);
        write(fp,s2,16);
        system("chmod 777 a.dat");
        system("od -bc a.dat");

}
```

output:

```
[user@localhost unix2]$ vi 2a.c
[user@localhost unix2]$ cc 2a.c
[user@localhost unix2]$ ./a.out
0000000 060 061 062 063 064 065 066 067 070 071 060 061 062 063 064 065
          0   1   2   3   4   5   6   7   8   9   0   1   2   3   4   5
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
[user@localhost unix2]$ cat -v a.dat
0123456789012345^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^@^
@^@abcdefghijklmnop
```

2b) c program that accepts valid filename as command line argument and print the type of the file

```
#include<stdio.h>
#include<unistd.h>
#include<sys/types.h>
#include<sys/stat.h>
int main(int argc,char*argv[])
{
	struct stat buf;
	int I=1;
	if(argc==1)
	{
		printf("No Arguments");

	}
	else
	{
		do
		{
			lstat(argv[I],&buf);
			if(S_ISREG(buf.st_mode))
				printf("%s is Regular File\n",argv[I]);
			else if(S_ISBLK(buf.st_mode))
				printf("%s is Block File\n",argv[I]);
			else if(S_ISCHR(buf.st_mode))
				printf("%s is Charecter File\n",argv[I]);
			else if(S_ISDIR(buf.st_mode))
				printf("%s is Directory\n",argv[I]);
			else if(S_ISFIFO(buf.st_mode))
				printf("%s is FIFO File\n",argv[I]);
			else if(S_ISLNK(buf.st_mode))
				printf("%s is symbolic Link File\n",argv[I]);
			else
				printf("%s is Unknown File\n");
			I++;
		}while(I<argc);
	}
}
```

output:

```
[user@localhost unix2]$ vi 2b.c
[user@localhost unix2]$ cc 2b.c
[user@localhost unix2]$ ./a.out m.c
m.c is Regular File
[user@localhost unix2]$ ./a.out 1b.c
1b.c is Regular File
```

3a)Script to echo args 1-per line, translating lower to upper case.

```
if [ $# -eq 0 ]
then
 echo "Error - No args!"
exit
fi
for i
do
      echo $i|tr '[a-z]' '[A-Z]'
done
```

output:

```
[user@localhost unix2]$ vi 3a.sh
[user@localhost unix2]$ sh 3a.sh
Error - No args!
[user@localhost unix2]$ sh a.sh bangalore
BANGALORE
```

3b) c program to run command & determine the time taken by it

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/times.h>
#include<unistd.h>

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

        struct tms r1,r2;
        clock_t end,start;
        long clk;
        if(argc==1)
        printf("enter arguments\n");
        else
        {
        start= times(&r1);
        system("clear");
        system(argv[1]);
        end=times(&r2);
        clk=sysconf(_SC_CLK_TCK);
        printf("time taken=%f\n",(end-start)/(double)clk);
        }
}
```

output:

```
[user@localhost ~]$ vi 3b.c
[user@localhost ~]$ cc 3b.c
[user@localhost ~]$ ./a.out
enter arguments
[user@localhost ~]$ ./a.out ps

  PID TTY          TIME CMD
 2147 pts/0    00:00:00 bash
 2944 pts/0    00:00:00 a.out
 2946 pts/0    00:00:00 ps
time taken=0.070000
```

4a)Shell script to check file permission, process status, date & current user using case conditional statement.

```
echo "Menu
1: list of files
2: process status
3: date
4: users
5: quit to terminal
enter ur choice:"

read choice
case "$choice" in
1) ls -l;;
2) ps;;
3) date;;
4) who;;
5) exit;;
*) echo "invalid entry";;
esac
```

output:

```
[user@localhost ~]$ vi 4a.sh
[user@localhost ~]$ sh 4a.sh
Menu
1: list of files
2: process status
3: date
4: users
5: quit to terminal
enter ur choice:
4
user     :0           2014-05-15 09:04 (:0)
user     pts/0        2014-05-15 09:07 (:0)
```

4b) AWK script to print transpose of any NxM matrix.

```
BEGIN{}
        {
        for(i=1;i<=3;i++)
        a[i]=a[i]" " $i
        }

END{
        {
        for(i=1;i<=3;i++)
        print a[i]
        }
}
```

output

```
[user@localhost unix2]$ cat >m.c
1 2 5
2 3 4
3 6 7
[user@localhost unix2]$ vi 7b.awk


[user@localhost unix2]$ awk -f 7b.awk m.c
 1 2 3
 2 3 6
 5 4 7
```

5a)Script to print home dir of given login name.

```
if [ $# -eq 0 ]
then
        echo "enter atleast one Arguments"
else
for i in $*
do
temp=`grep "$i:" /etc/passwd|cut -d ":" -f6`
if [ -z "$temp" ]
then
echo "$i not a valid login name"
else
echo "$i is a valid login name "
echo "the directory $temp"
fi
done
fi
```

output:

```
[user@localhost unix2]$ vi 5a.sh
[user@localhost unix2]$ sh 5a.sh user
user is a valid login name
the directory /
/home/user
[user@localhost unix2]$ sh 5a.sh users
users not a valid login name
```

5b)Script to accept 2 files as args, sorts both to temp files, merges the sorted files to stdout and finally delete temporary files.

```
if [ $# -ne 2 ]
then
echo "Error - 2 args required!";
exit;
fi

sort -o temp1 $1
sort -o temp2 $2
sort -m temp1 temp2
rm temp?
```

Output:

```
[user@localhost unix2]$ cat >t1.txt
*****
$$$$$
@@@@@
[user@localhost unix2]$ cat >t2.txt
uvce
bangalore
abcd
1656
[user@localhost unix2]$ sh 5b.sh t1.txt t2.txt
$$$$$
*****
1656
@@@@@
abcd
bangalore
uvce
```

6a)Script to display calendar for current month, with date replaced by * or ** depending on current date.

```
day=`date +%d`
if [ $day -lt 10 ]
then
cal|sed "s/$day/*/"
else
cal|sed "s/$day/**/"
fi
```

output

```
[user@localhost unix2]$ vi 6a.sh
[user@localhost unix2]$ sh 6a.sh
      May 2014
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
```

6b)Shell script to implement terminal locking.

```
stty -echo
echo "Enter a Password"
read pswd
clear
npwd=
trap '' 0 1 2
echo "The Terminal is Locked!!"
while test "$npwd" != "$pswd"
do
      echo "Enter the password again:"
      read npwd
done
echo "Correct password"
echo "Terminal Lock has been Opened"
stty echo
```

output:

```
[user@localhost unix2]$ vi 6b.sh
[user@localhost unix2]$ sh 6b.sh
Enter a Password


The Terminal is Locked!!
Enter the password again:
Enter the password again:
Correct password
Terminal Lock has been Opened
```

SYSTEM SOFTWARE PROGRAMS

Execution of the following programs using LEX:

1. Program to count the number of vowels and consonants in a given string.

```
%{
#include<stdio.h> int vc=0,cc=0;
%}

%%
[aeiouAEIOU] vc++;
[a-zA-Z] cc++;
[ \n\t] ;
%%
int yywrap()
{
            return 1;
}  main()
{
         printf("enter a string\n");           yylex();
            printf("no. of vowels=%d\n no of consonant=%d\n",vc,cc);
}
```

```
***********output***********

$ lex pgm1.l
$ gcc -o pgm1 lex.yy.c
$ ./pgm1 enter a string uvce is
our college no. of vowels=8 no
of consonant=8
```

2. Program to count the number of characters, words, spaces and lines in a given input file.

```
%{
#include<stdio.h> int
cc=0,wc=0,sc=0,lc=0;
%}

%%
[^ \n\t]+ wc++,cc+=yyleng;
```

```
[ ] sc++,cc++;
[\n] lc++;
[\t] sc+=8,cc+=8;
%%

int yywrap()
{   return 1;
}  main()
{
          char fname[10];     printf("enter the file
name\n");
            scanf("%s",fname);
          yyin=fopen(fname,"r");   yylex();
           printf("character=%d\n wrds=%d\n spaces=%d\n
lines=%d\n",cc,wc,sc,lc);
}
```

**************output****************

```
$ lex pgm2.l
$ gcc -o pgm2 lex.yy.c
$ ./pgm2 enter the file name
ex.txt character=154 wrds=29
spaces=23 lines=9
```

3. Program to count the (i) positive and negative integers (ii) positive and negative fractions.

```
%{
#include<stdio.h> int
pi=0,ni=0,pf=0,nf=0;
%}
D[0-9]
%%
{D}+ pi++;
-{D}+ ni++;
{D}*"."{D}+ pf++;
-{D}*"."{D}+ nf++;
%%
int yywrap()
{ return 1;
}
void main()
```

```
{
          printf("enter the number");        yylex();
             printf("+ve i=%d\n-ve i=%d\n+ve f=%d\n-ve f=%d",pi,ni,pf,nf);
}
```

**********output*********

```
$ lex pgm3.l
$ gcc -o pgm3 lex.yy.c
$ ./pgm3 enter the number-1

2.5

2.3
2

-2.33

+ve i=1
-ve i=1
+ve f=2
-ve f=1
```

4. Program to count the number of comment lines in a given C program.Also eliminate them and copy that program into separate file.

```
%{
#include<stdio.h> int count;
%}

%%
"/*"[a-zA-Z0-9' '\n\t]*"*/" count++;
"//"[^ \n]* count++;
%%
int yywrap()
{ return 1;
}  main()
{
          char fname1[10],fname2[10];    printf("enter the
file1");         scanf("%s",fname1);
yyin=fopen(fname1,"r");           printf("enter the file2");
scanf("%s",fname2);
```

```
        yyout=fopen(fname2,"w");       yylex();
           printf("no of cmnts=%d\n",count);
}
```

```
**********output***********

$ lex pgm4.l
$ gcc -o lex.yy.c $ ./pgm4 enter the
file1 ex.txt enter the file2 ex2.txt no
of cmnts=3
```

5. Program to count the number of scanf and printf statements in a C program. Replace them with readf and writef statements respectively.

```
%{
#include<stdio.h> int pf=0,sf=0;
%}

%%
"printf" {fprintf(yyout,"writef"); pf++;}
"scanf" {fprintf(yyout,"readf"); sf++;}
%%  yywrap() {
return 1;
}  main() { char fname1[10],fname2[10];
printf("enter file1"); scanf("%s",fname1);
yyin=fopen(fname1,"r"); printf("enter
file2"); scanf("%s",fname2);
yyout=fopen(fname2,"w"); yylex();
printf("printf=%d\nscanf=%d\n",pf,sf);
}
```

```
**********output*********

$ lex pgm5.l
$ gcc -o pgm5 lex.yy.c
$ ./pgm5 enter file11.c enter
file2ex2.txt printf=2 scanf=0
```

Execution of the following programs using YACC:

1. Program to test the validity of a simple expression involving operators +,-,*,/

```
%{
#include "y.tab.h"
%}

%%
[a-zA-Z][a-zA-Z0-9]* return ID;
[0-9]+ return NUM;
[\n] return NL;
. return yytext[0];
%%
int yywrap()
{ return 1;
}
```

1.y

```
%{
#include<stdio.h>
#include<stdlib.h>
%}
%token NUM ID NL
%left '*''/'
%left '+''-'
%%
stmt:exp NL {printf("valid\n");exit(0);} exp : exp'+'exp
| exp'-'exp
     | exp'*'exp
     | exp'/'exp
     | '('exp')'
     | '['exp']'
     | '{'exp'}'
     | NUM
     | ID
     ;
%%
main()
{
          printf("enter expression\n");      yyparse();
```

2.L

```
} yyerror()
{
        printf("invalid\n");         exit(0);
}
************output************
$ yacc -d 1.y
$ cc y.tab.c lex.yy.c $ ./a.out enter
expression a+[b-(c+)*d] invalid $
./a.out enter expression a+[b-
(c+d)*e] valid
```

2. Program to recognize a valid arithmetic expression that user operators +,-,*,/.

```
%{
#include "y.tab.h"
%}
%%
[0-9]|[0-9]*"."[0-9]+ return NUM;
[\n] return NL;
. return yytext[0];
%% int yywrap()
{
           return 1;
}
%{
#include<stdio.h>
#include<stdlib.h>
%}
%token NUM NL
%left '*''/'
%left '+''-' %%
stmt:exp NL {printf("valid\n");exit(0);}
    ;exp: exp'+'exp
   | exp'-'exp
   | exp'*'exp
   | exp'/'exp
   | '('exp')'
   | '{'exp'}'
   | '['exp']'
   | NUM     ;%% main()
{
        printf("enter the exp\n");          yyparse();
} yyerror()
{
        printf("invalid\n");          exit(0);
}
************output***********
$ yacc -d 3.y
$ cc y.tab.c lex.yy.c
$ ./a.out enter the exp 1+7-
5*(4+4)/7 valid $ ./a.out enter
the exp 1++ invalid
```

3. Program to recognize nested IF control statements and display the number of levels of nesting.

```
%{
#include "y.tab.h"
%}

%%

"if" return IF;
[0-9]+ return NUM;
[a-zA-Z][a-zA-Z0-9]* return ID;
[*/+-] return BIN;
[=] return EQU;
"++"|"--" return INC;
"=="|"<"|">"|"<="|">="|"!=" return REL;
. return yytext[0];

%%
int yywrap()
{
          return 1;
}

$ gedit 2.y

%{
#include<stdio.h>
#include<stdlib.h> int count=0;
%}
```

```
%token IF NUM ID BIN EQU INC REL
 %%
st: com_nest {printf("valid\nno. of nesting:%d\n",count);exit(0);}
   ;  com_nest: nest {count++;}
            ;

nest: IF'('cond')' one_st
      IF'('cond')' '{'many_st'}'       IF'('cond')' '{'nest'}'
{count++;}
      ; cond: ID REL ID      | ID
REL NUM
     | NUM REL NUM
     | ID
     | NUM
     ;



  one_st: ID EQU ID BIN ID';'       | ID
EQU ID BIN NUM';'
      | ID EQU NUM BIN NUM';'
      | ID EQU NUM';'
      | ID INC';'
      ;  many_st: many_st one_st        |
one_st one_st
       ;

%%  main() {
   printf("enter the statement\n");    yyparse();
}  yyerror() {    printf("invalid\n");
exit(0);
}

*********output***********

$ lex 2.l
$ yacc -d 2.y
$ cc y.tab.c lex.yy.c $ ./a.out
enter the statement if(a=b){if(1){exit()}} no. of nesting:2
enter the statement if(a=b){exit()} no. of nesting:1 enter
the statement k no. of nesting:0
```