School of Engineering and Applied Science

Ahmedabad University

Name: Charvik Patel

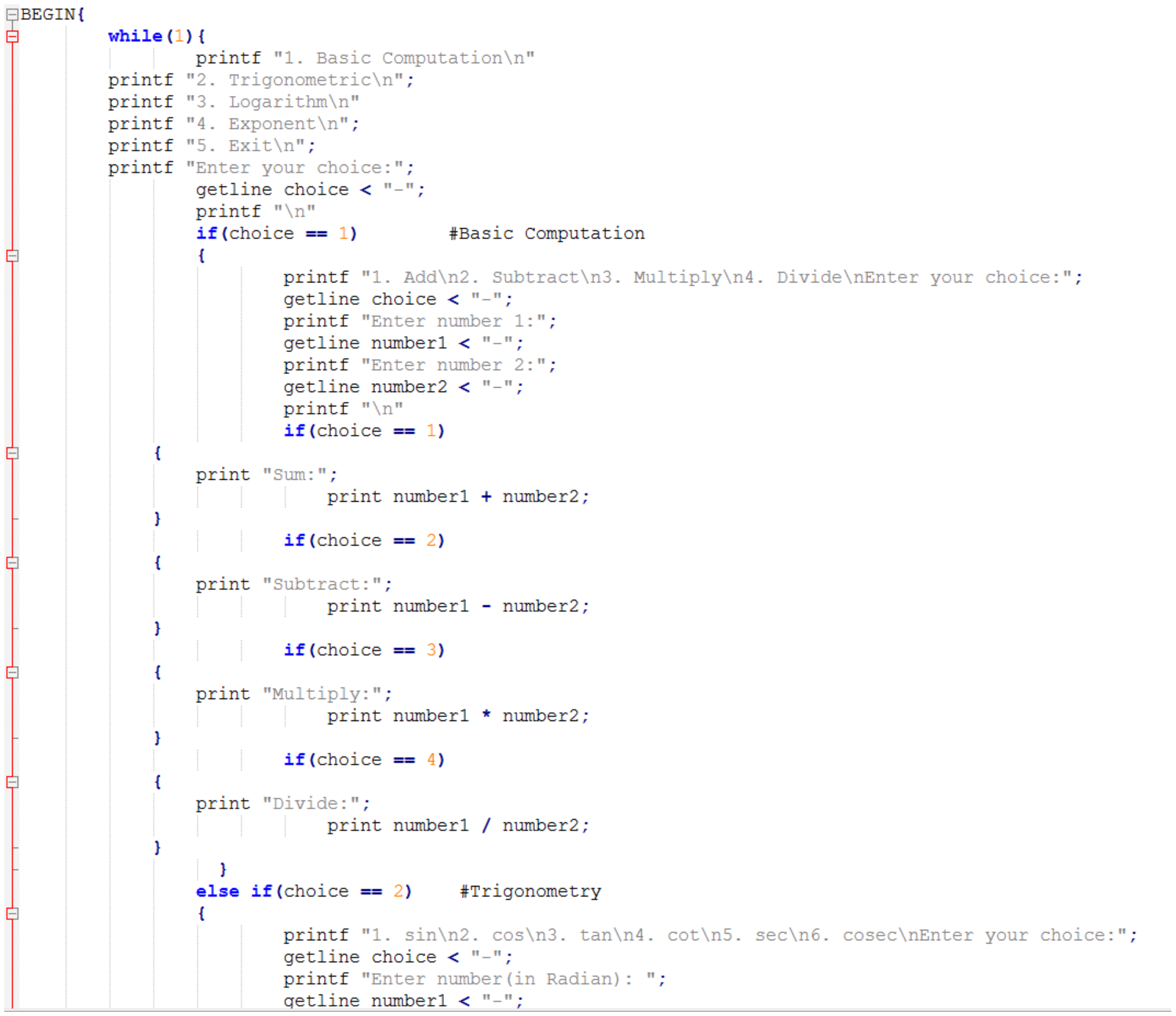
Roll no: 1401079

Assignment: 2

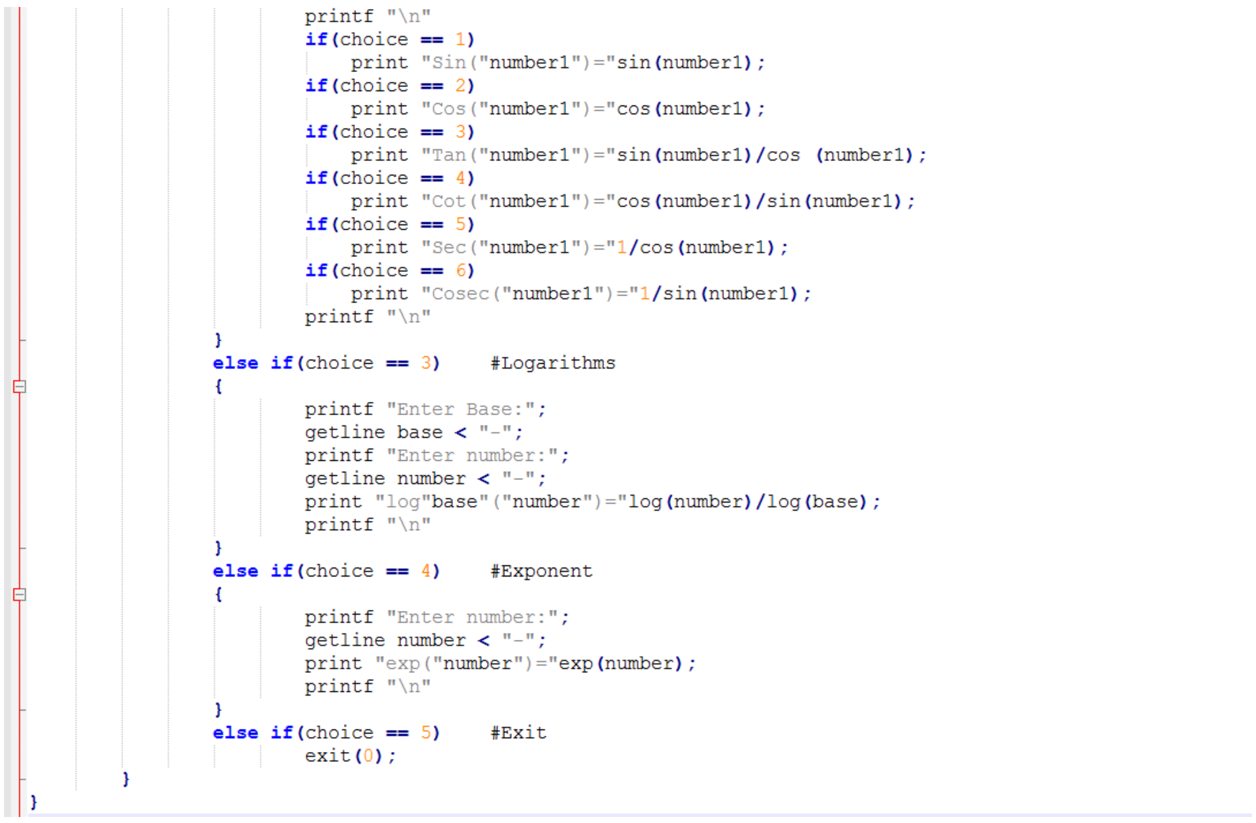
Subject: Operating System

1. Write down a program which works like a scientific calculator and performs following operations
   1) Floating point calculations
   2) Trigonometric operations
   3) Logarithms
   4) Exponent etc.

Code:

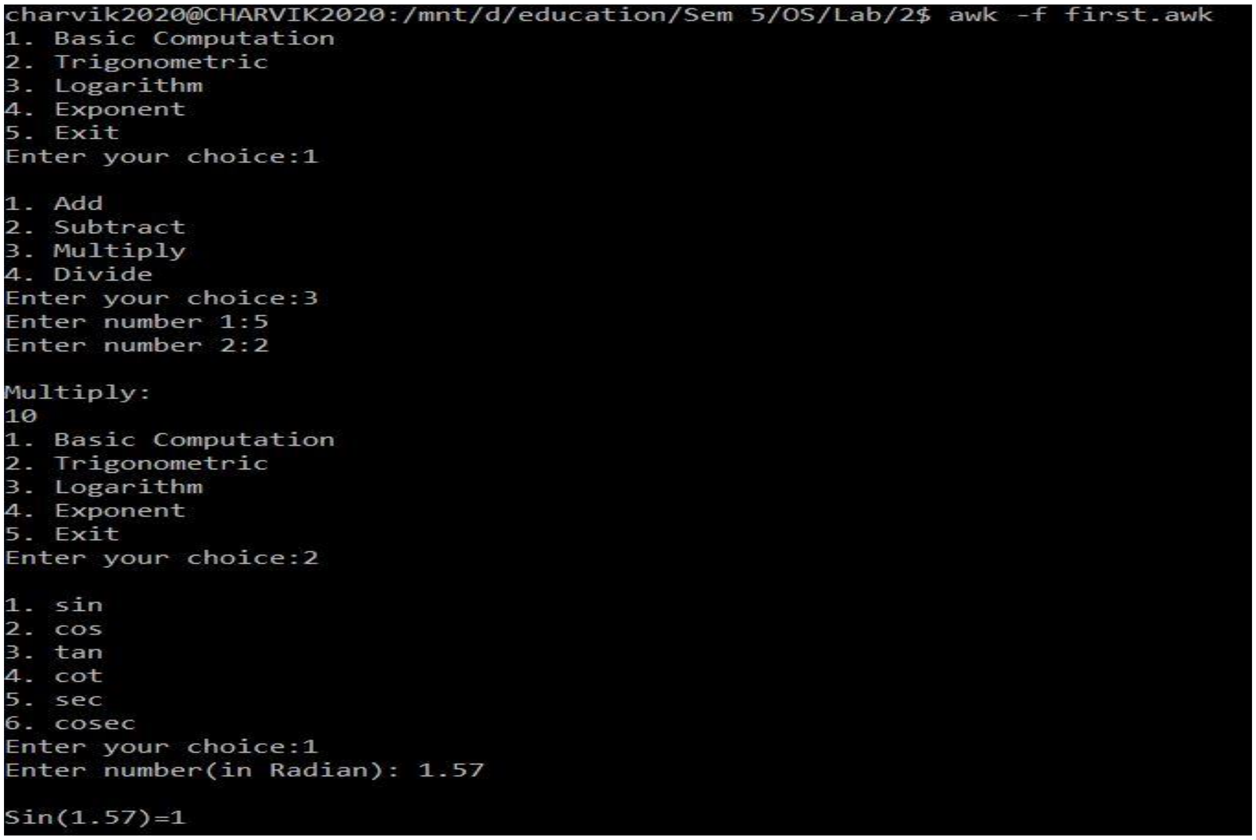


```
BEGIN{
        while(1){
                printf "1. Basic Computation\n"
        printf "2. Trigonometric\n";
        printf "3. Logarithm\n"
        printf "4. Exponent\n";
        printf "5. Exit\n";
        printf "Enter your choice:";
                getline choice < "-";
                printf "\n"
                if(choice == 1)        #Basic Computation
                {
                        printf "1. Add\n2. Subtract\n3. Multiply\n4. Divide\nEnter your choice:";
                        getline choice < "-";
                        printf "Enter number 1:";
                        getline number1 < "-";
                        printf "Enter number 2:";
                        getline number2 < "-";
                        printf "\n"
                        if(choice == 1)
            {
                print "Sum:";
                            print number1 + number2;
            }
                        if(choice == 2)
            {
                print "Subtract:";
                            print number1 - number2;
            }
                        if(choice == 3)
            {
                print "Multiply:";
                            print number1 * number2;
            }
                        if(choice == 4)
            {
                print "Divide:";
                            print number1 / number2;
            }
                  }
                else if(choice == 2)     #Trigonometry
                {
                        printf "1. sin\n2. cos\n3. tan\n4. cot\n5. sec\n6. cosec\nEnter your choice:";
                        getline choice < "-";
                        printf "Enter number(in Radian): ";
                        getline number1 < "-";
```

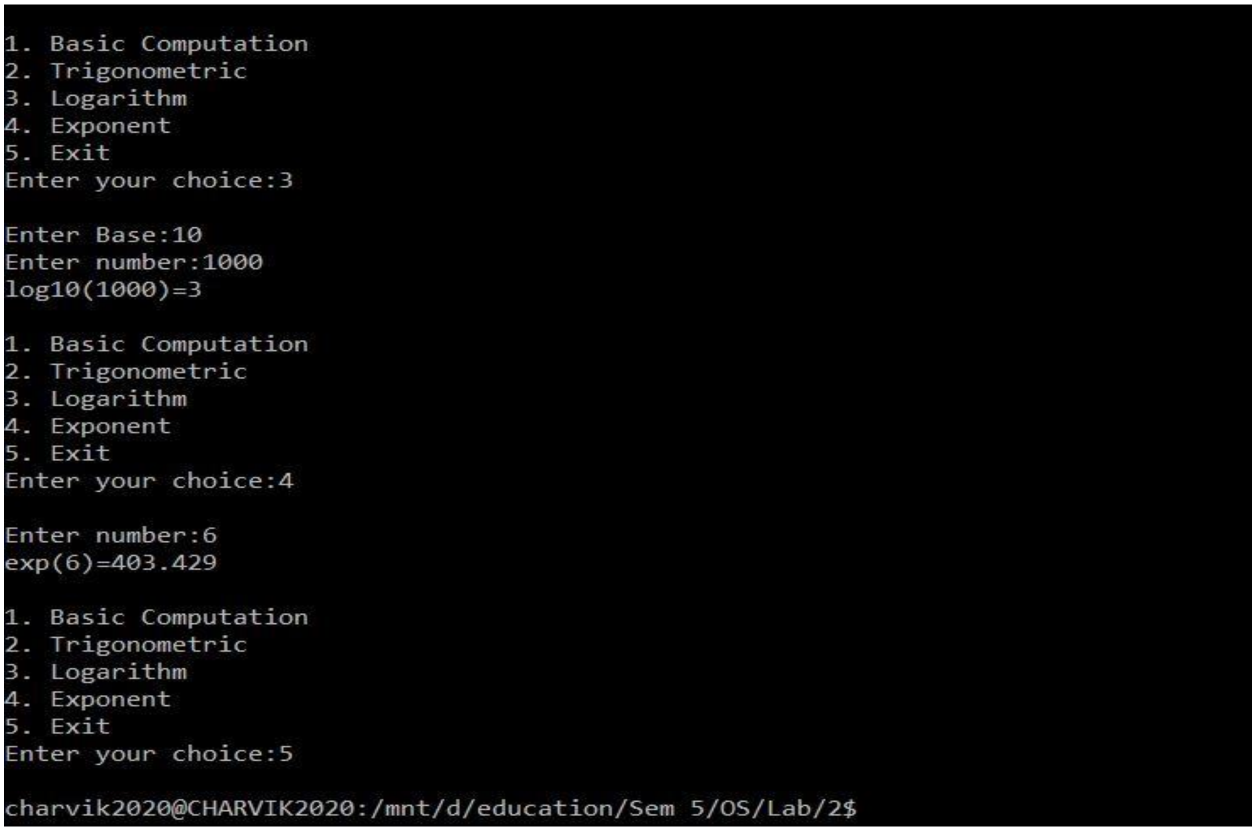


```
                        printf "\n"
                        if(choice == 1)
                            print "Sin("number1")="sin(number1);
                        if(choice == 2)
                            print "Cos("number1")="cos(number1);
                        if(choice == 3)
                            print "Tan("number1")="sin(number1)/cos  (number1);
                        if(choice == 4)
                            print "Cot("number1")="cos(number1)/sin(number1);
                        if(choice == 5)
                            print "Sec("number1")="1/cos(number1);
                        if(choice == 6)
                            print "Cosec("number1")="1/sin(number1);
                        printf "\n"
                }
                else if(choice == 3)     #Logarithms
                {
                        printf "Enter Base:";
                        getline base < "-";
                        printf "Enter number:";
                        getline number < "-";
                        print "log"base"("number")="log(number)/log(base);
                        printf "\n"
                }
                else if(choice == 4)     #Exponent
                {
                        printf "Enter number:";
                        getline number < "-";
                        print "exp("number")="exp(number);
                        printf "\n"
                }
                else if(choice == 5)     #Exit
                        exit(0);
        }
}
```

Output:



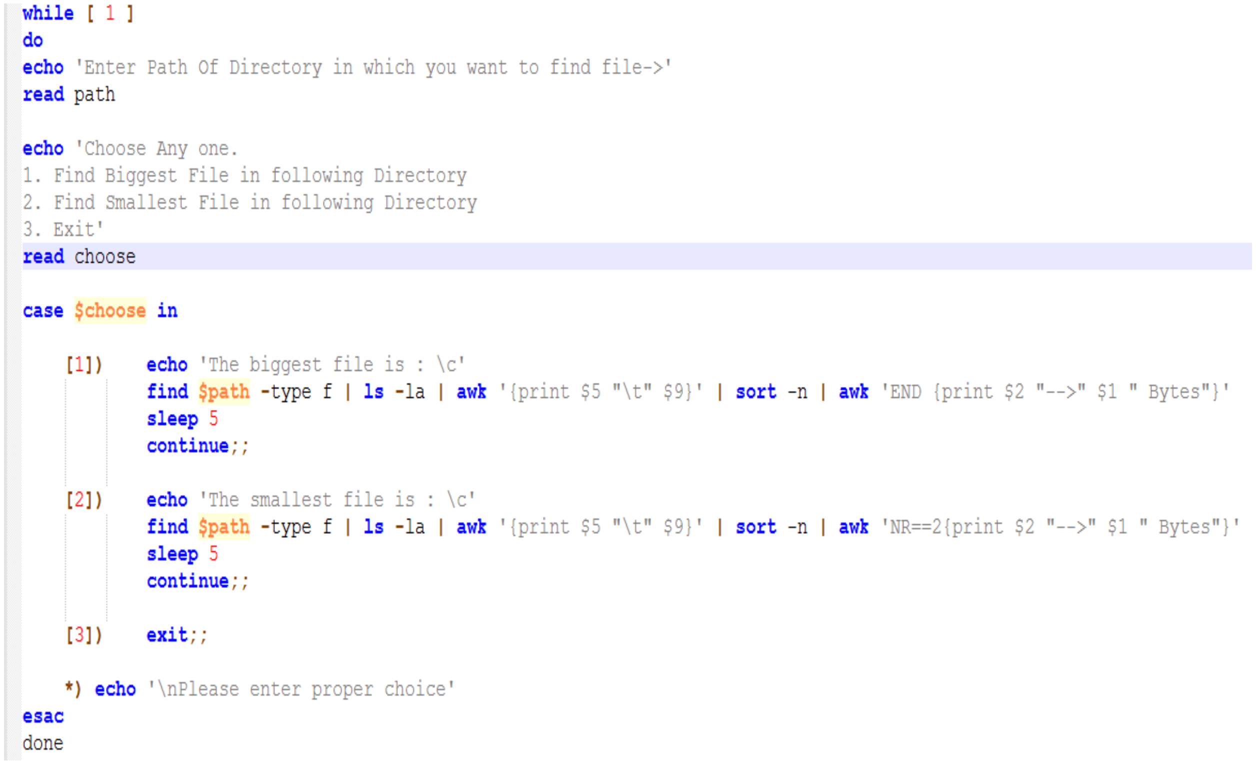
```
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2$ awk -f first.awk
1. Basic Computation
2. Trigonometric
3. Logarithm
4. Exponent
5. Exit
Enter your choice:1

1. Add
2. Subtract
3. Multiply
4. Divide
Enter your choice:3
Enter number 1:5
Enter number 2:2

Multiply:
10
1. Basic Computation
2. Trigonometric
3. Logarithm
4. Exponent
5. Exit
Enter your choice:2

1. sin
2. cos
3. tan
4. cot
5. sec
6. cosec
Enter your choice:1
Enter number(in Radian): 1.57

Sin(1.57)=1
```



```

1. Basic Computation
2. Trigonometric
3. Logarithm
4. Exponent
5. Exit
Enter your choice:3

Enter Base:10
Enter number:1000
log10(1000)=3

1. Basic Computation
2. Trigonometric
3. Logarithm
4. Exponent
5. Exit
Enter your choice:4

Enter number:6
exp(6)=403.429

1. Basic Computation
2. Trigonometric
3. Logarithm
4. Exponent
5. Exit
Enter your choice:5

charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2$
```

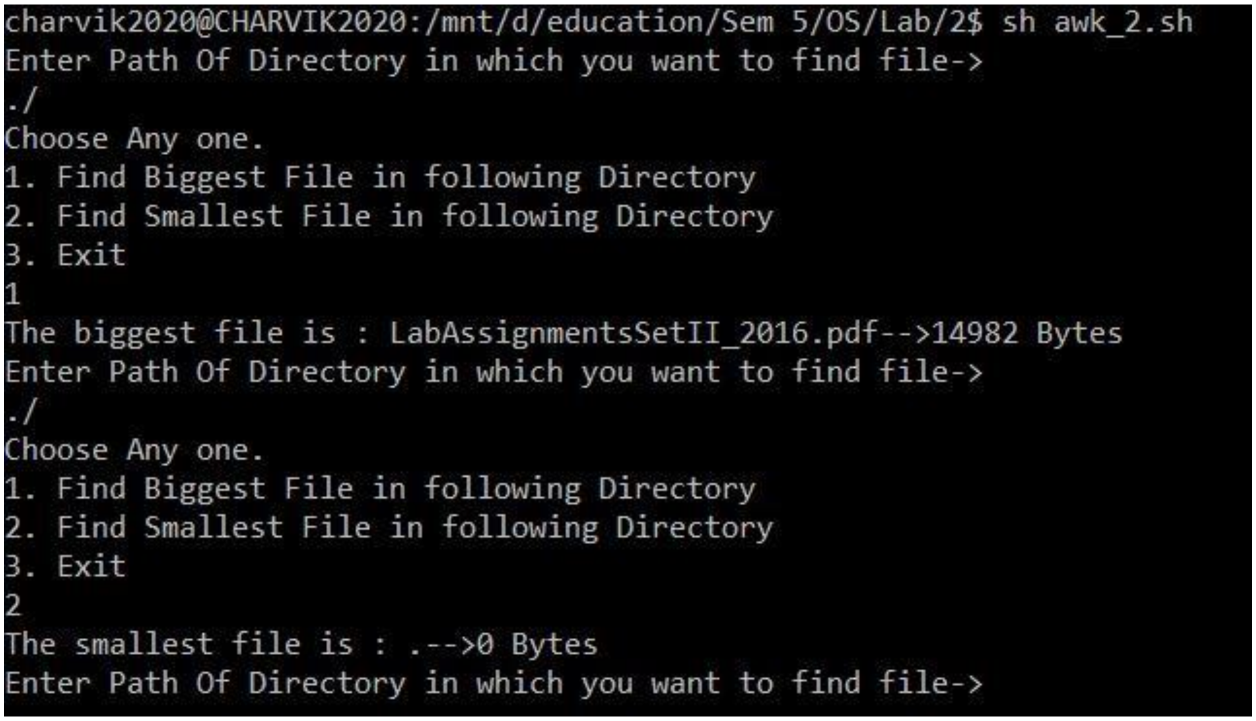
2. Find out following files on your machine.
   a. Biggest file, if more than one exists than list them
   b. Smallest file, if more than one exists than list them

Code:



```
while [ 1 ]
do
echo 'Enter Path Of Directory in which you want to find file->'
read path

echo 'Choose Any one.
1. Find Biggest File in following Directory
2. Find Smallest File in following Directory
3. Exit'
read choose

case $choose in

    [1])    echo 'The biggest file is : \c'
            find $path -type f | ls -la | awk '{print $5 "\t" $9}' | sort -n | awk 'END {print $2 "-->" $1 " Bytes"}'
            sleep 5
            continue;;

    [2])    echo 'The smallest file is : \c'
            find $path -type f | ls -la | awk '{print $5 "\t" $9}' | sort -n | awk 'NR==2{print $2 "-->" $1 " Bytes"}'
            sleep 5
            continue;;

    [3])    exit;;

    *) echo '\nPlease enter proper choice'
esac
done
```

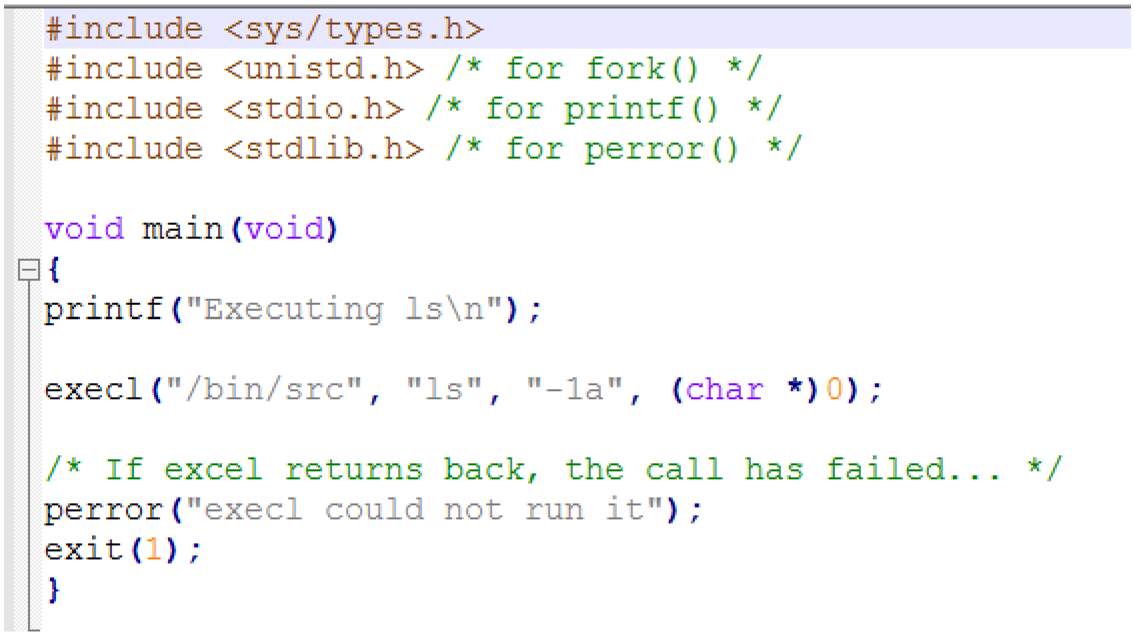
Output:



```
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2$ sh awk_2.sh
Enter Path Of Directory in which you want to find file->
./
Choose Any one.
1. Find Biggest File in following Directory
2. Find Smallest File in following Directory
3. Exit
1
The biggest file is : LabAssignmentsSetII_2016.pdf-->14982 Bytes
Enter Path Of Directory in which you want to find file->
./
Choose Any one.
1. Find Biggest File in following Directory
2. Find Smallest File in following Directory
3. Exit
2
The smallest file is : .-->0 Bytes
Enter Path Of Directory in which you want to find file->
```

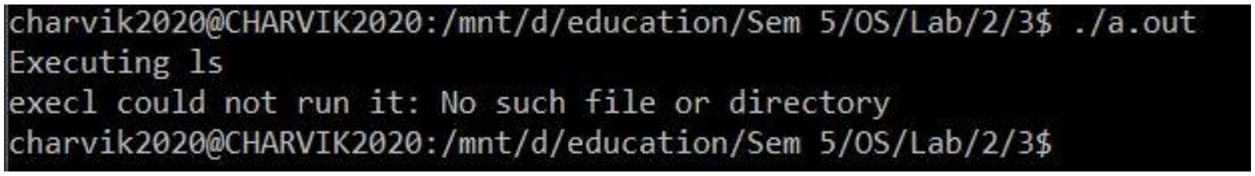
3. Run all the programs related to process creation/termi nati on shown i n the lectures (Also given at ftp).

Code 1: execlfail.c



```c
#include <sys/types.h>
#include <unistd.h> /* for fork() */
#include <stdio.h> /* for printf() */
#include <stdlib.h> /* for perror() */

void main(void)
{
printf("Executing ls\n");

execl("/bin/src", "ls", "-1a", (char *)0);

/* If excel returns back, the call has failed... */
perror("execl could not run it");
exit(1);
}
```

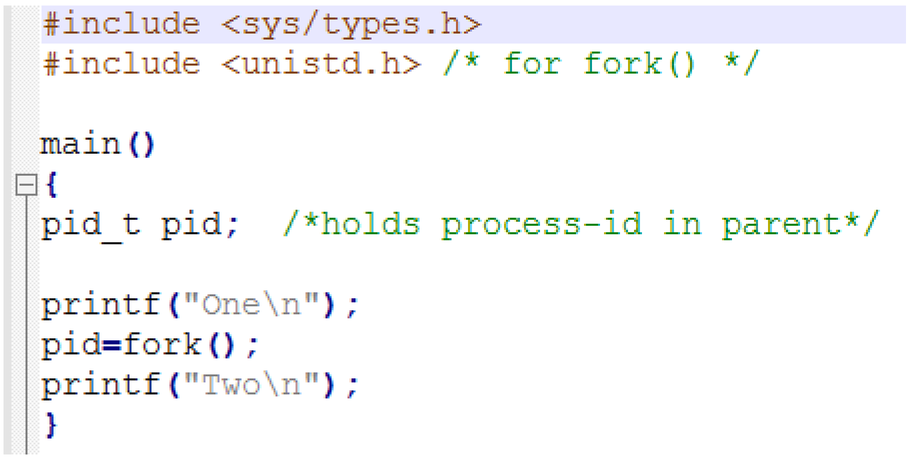
Output 1: execlfail.c



```
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2/3$ ./a.out
Executing ls
execl could not run it: No such file or directory
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2/3$
```

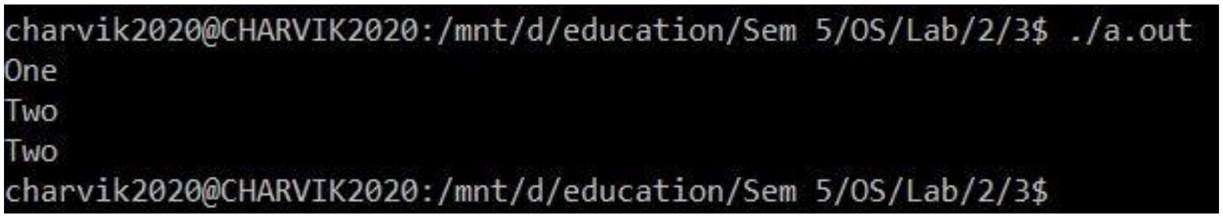
Explanation: execl command is used for executing file when a full pathname of the file is given. As in above case full path is not describe thus error message pop up.

Code 2: fork01.c



```c
#include <sys/types.h>
#include <unistd.h> /* for fork() */

main()
{
pid_t pid;  /*holds process-id in parent*/

printf("One\n");
pid=fork();
printf("Two\n");
}
```

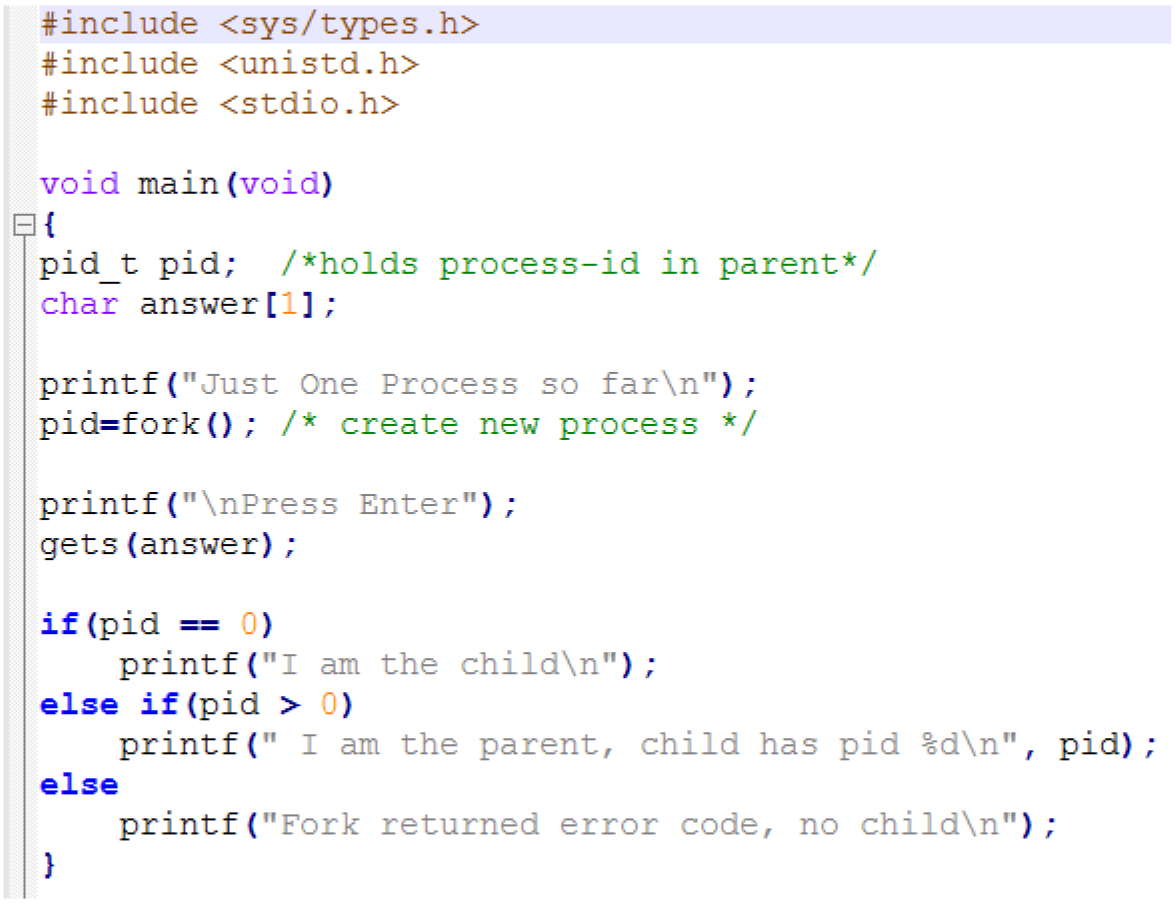
Output 2: fork01.c



```
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2/3$ ./a.out
One
Two
Two
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2/3$
```

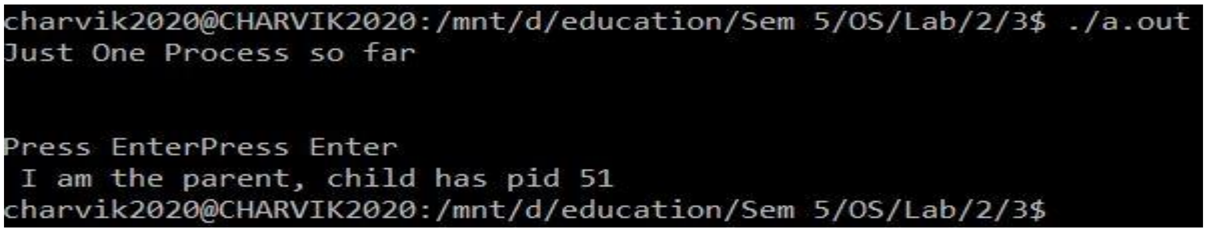
Explanation: fork() function called the child process is created which is duplicate of its running process.

Code 3: fork02.c



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

void main(void)
{
pid_t pid;  /*holds process-id in parent*/
char answer[1];

printf("Just One Process so far\n");
pid=fork(); /* create new process */

printf("\nPress Enter");
gets(answer);

if(pid == 0)
    printf("I am the child\n");
else if(pid > 0)
    printf(" I am the parent, child has pid %d\n", pid);
else
    printf("Fork returned error code, no child\n");
}
```

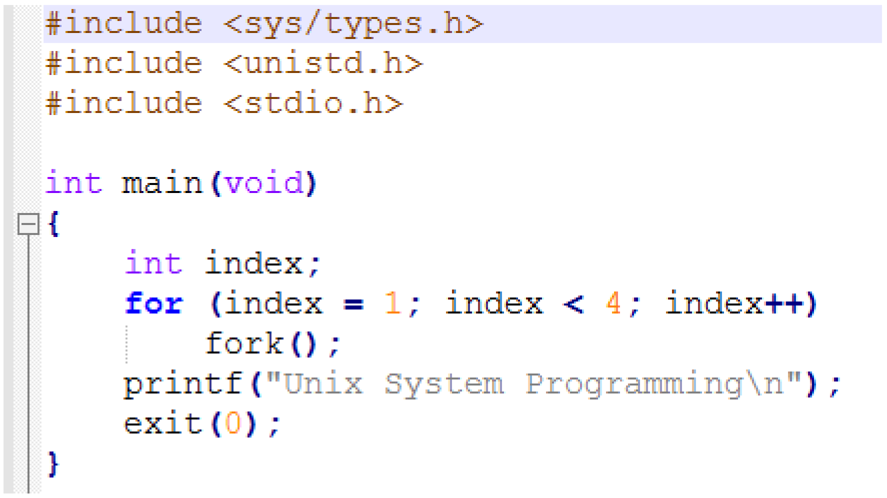
Output 3: fork02.c



```
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2/3$ ./a.out
Just One Process so far


Press EnterPress Enter
 I am the parent, child has pid 51
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2/3$
```

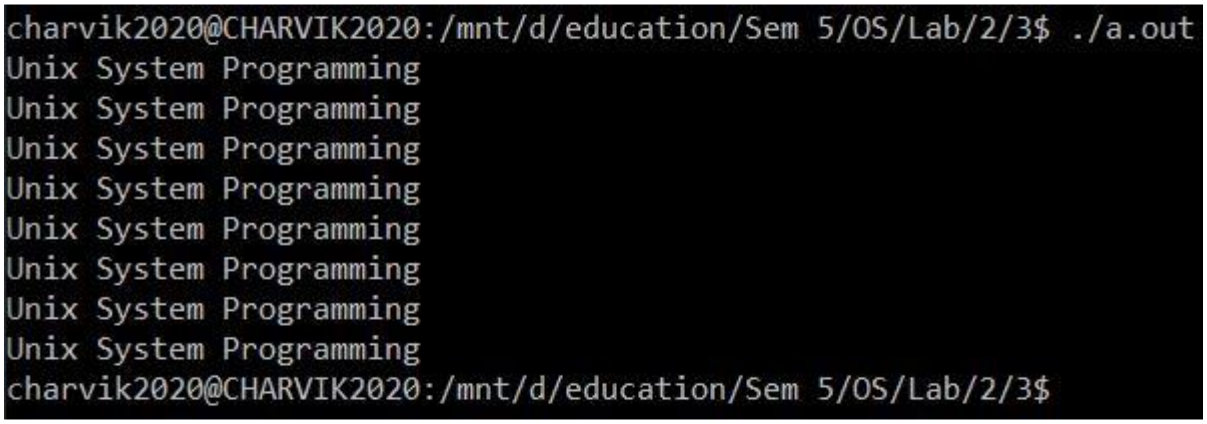
Explanation: here fork () function is called before the **press Enter** statement, hence **press Enter** Statement is printed twice and process id is given to child process.

Code 4: fork04.c



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

int main(void)
{
    int index;
    for (index = 1; index < 4; index++)
        fork();
    printf("Unix System Programming\n");
    exit(0);
}
```

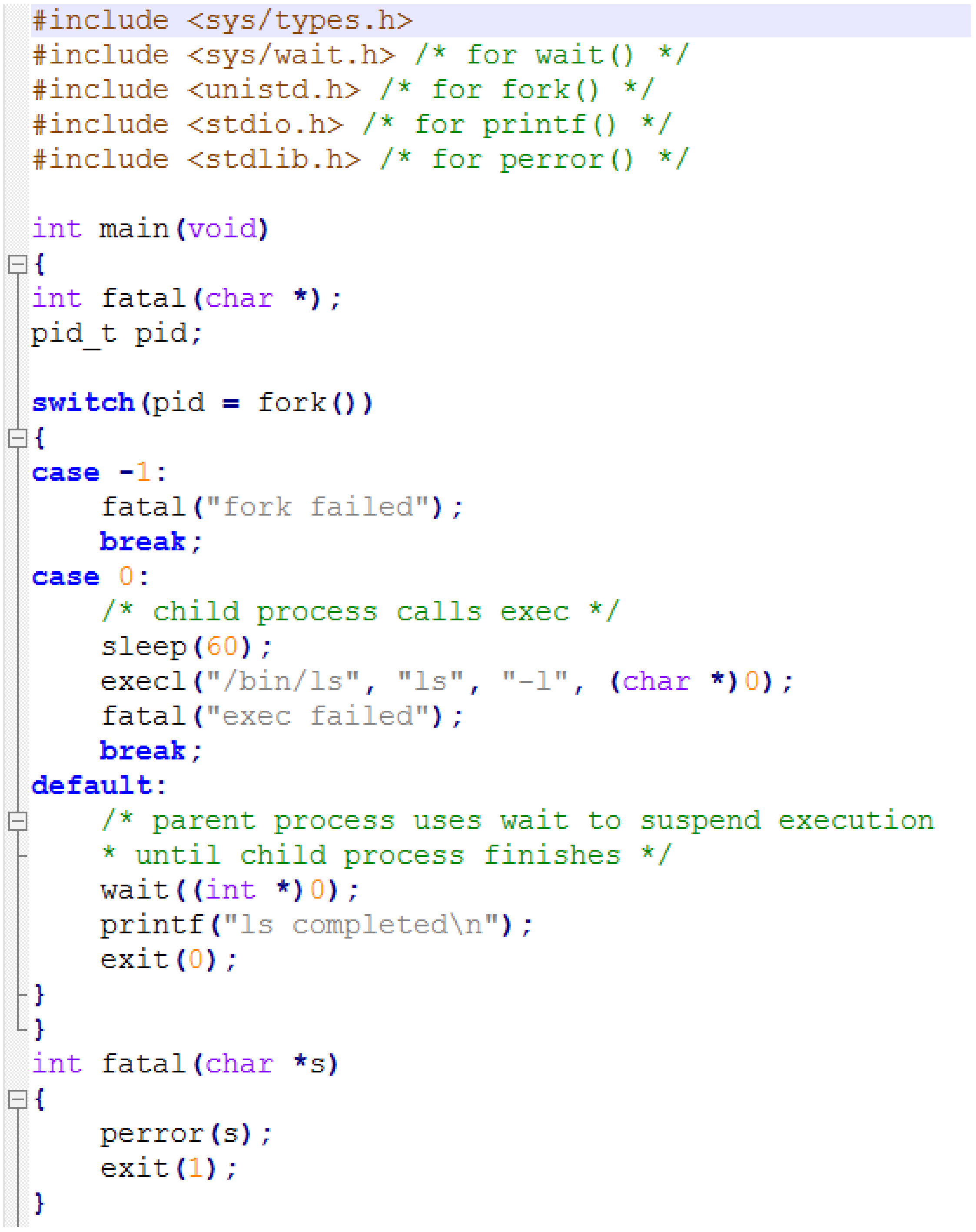
Output 4: fork04.c



```
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2/3$ ./a.out
Unix System Programming
Unix System Programming
Unix System Programming
Unix System Programming
Unix System Programming
Unix System Programming
Unix System Programming
Unix System Programming
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2/3$
```

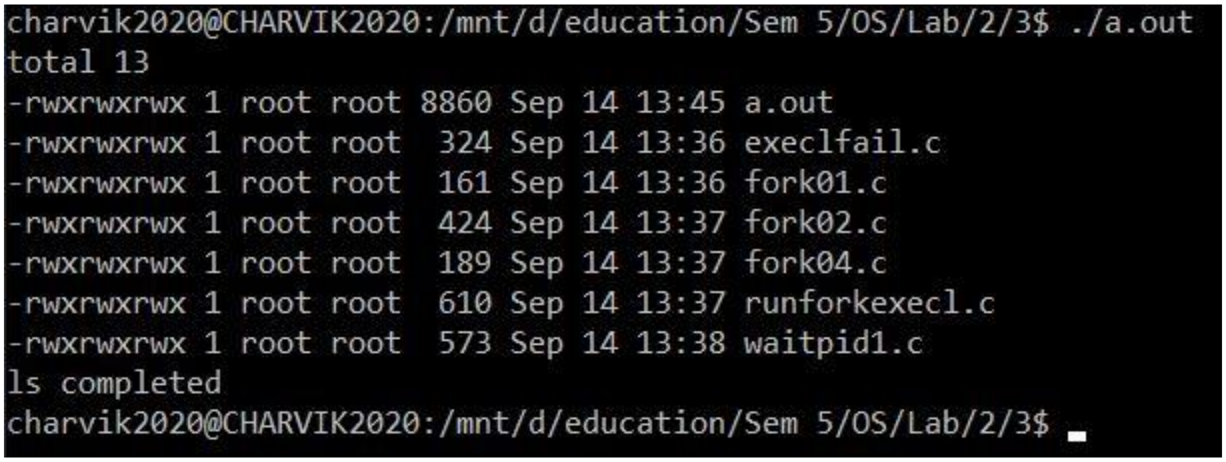
Explanation: here fork () function is written in for loop thus it will iterate for twice and thus as it is in loop print statement will print for 8 times

Code 5: runforkexecl.c



```c
#include <sys/types.h>
#include <sys/wait.h> /* for wait() */
#include <unistd.h> /* for fork() */
#include <stdio.h> /* for printf() */
#include <stdlib.h> /* for perror() */

int main(void)
{
int fatal(char *);
pid_t pid;

switch(pid = fork())
{
case -1:
    fatal("fork failed");
    break;
case 0:
    /* child process calls exec */
    sleep(60);
    execl("/bin/ls", "ls", "-l", (char *)0);
    fatal("exec failed");
    break;
default:
    /* parent process uses wait to suspend execution
    * until child process finishes */
    wait((int *)0);
    printf("ls completed\n");
    exit(0);
}
}
int fatal(char *s)
{
    perror(s);
    exit(1);
}
```

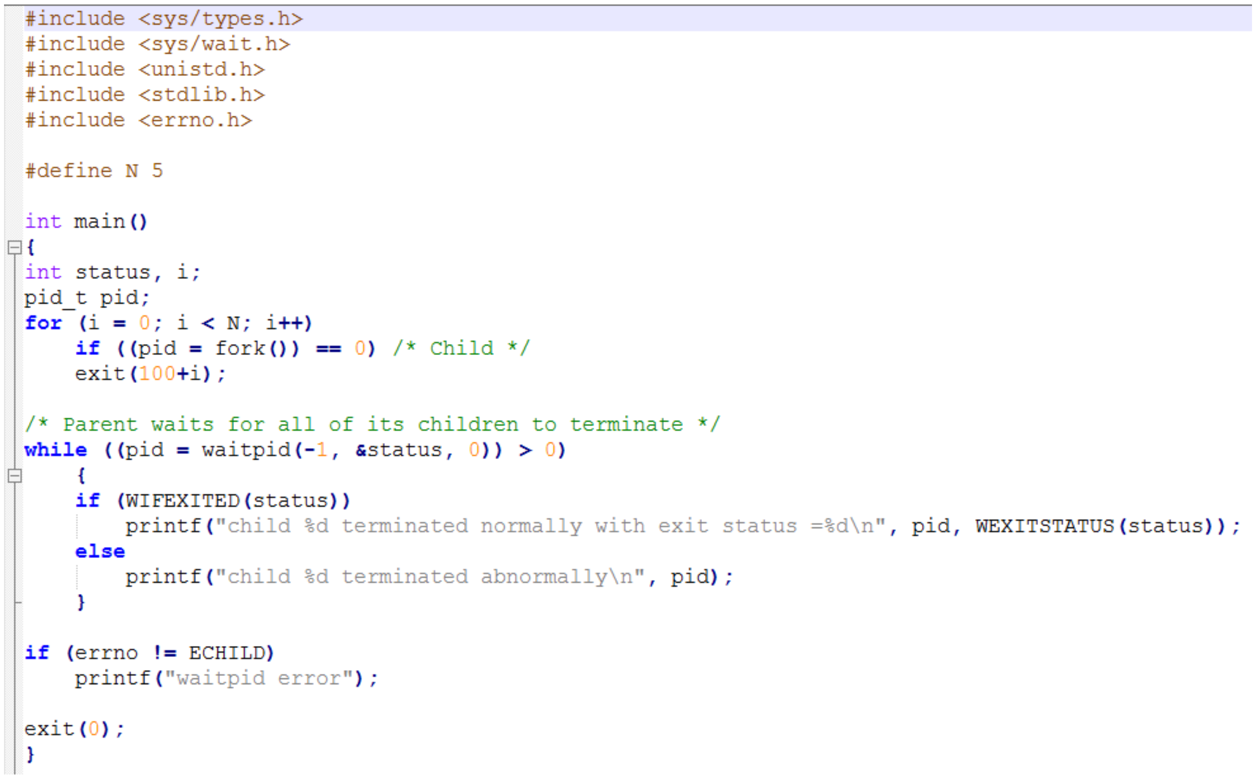
Output 5: runforkexecl.c



```
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2/3$ ./a.out
total 13
-rwxrwxrwx 1 root root 8860 Sep 14 13:45 a.out
-rwxrwxrwx 1 root root  324 Sep 14 13:36 execlfail.c
-rwxrwxrwx 1 root root  161 Sep 14 13:36 fork01.c
-rwxrwxrwx 1 root root  424 Sep 14 13:37 fork02.c
-rwxrwxrwx 1 root root  189 Sep 14 13:37 fork04.c
-rwxrwxrwx 1 root root  610 Sep 14 13:37 runforkexecl.c
-rwxrwxrwx 1 root root  573 Sep 14 13:38 waitpid1.c
ls completed
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2/3$
```

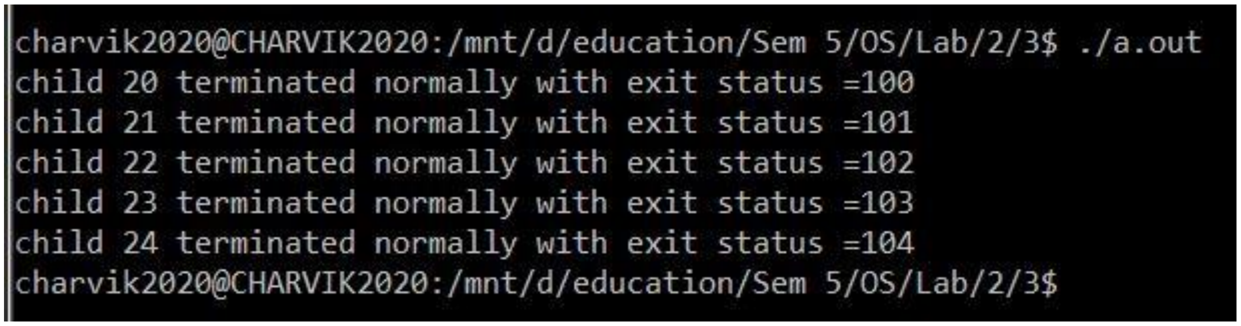
Explanation: Here there are three condition for executing the parent-child process Condition 1) if 0 then child process created successfully 2) if -1 child process not created successfully. 3) if other parent waits for the child process to finish the execution of ls command and performs its own task concurrently.

Code 6: waitpid1.c



```c
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <stdlib.h>
#include <errno.h>

#define N 5

int main()
{
int status, i;
pid_t pid;
for (i = 0; i < N; i++)
    if ((pid = fork()) == 0) /* Child */
    exit(100+i);

/* Parent waits for all of its children to terminate */
while ((pid = waitpid(-1, &status, 0)) > 0)
    {
    if (WIFEXITED(status))
        printf("child %d terminated normally with exit status =%d\n", pid, WEXITSTATUS(status));
    else
        printf("child %d terminated abnormally\n", pid);
    }

if (errno != ECHILD)
    printf("waitpid error");

exit(0);
}
```

Output 6: waitpid1.c

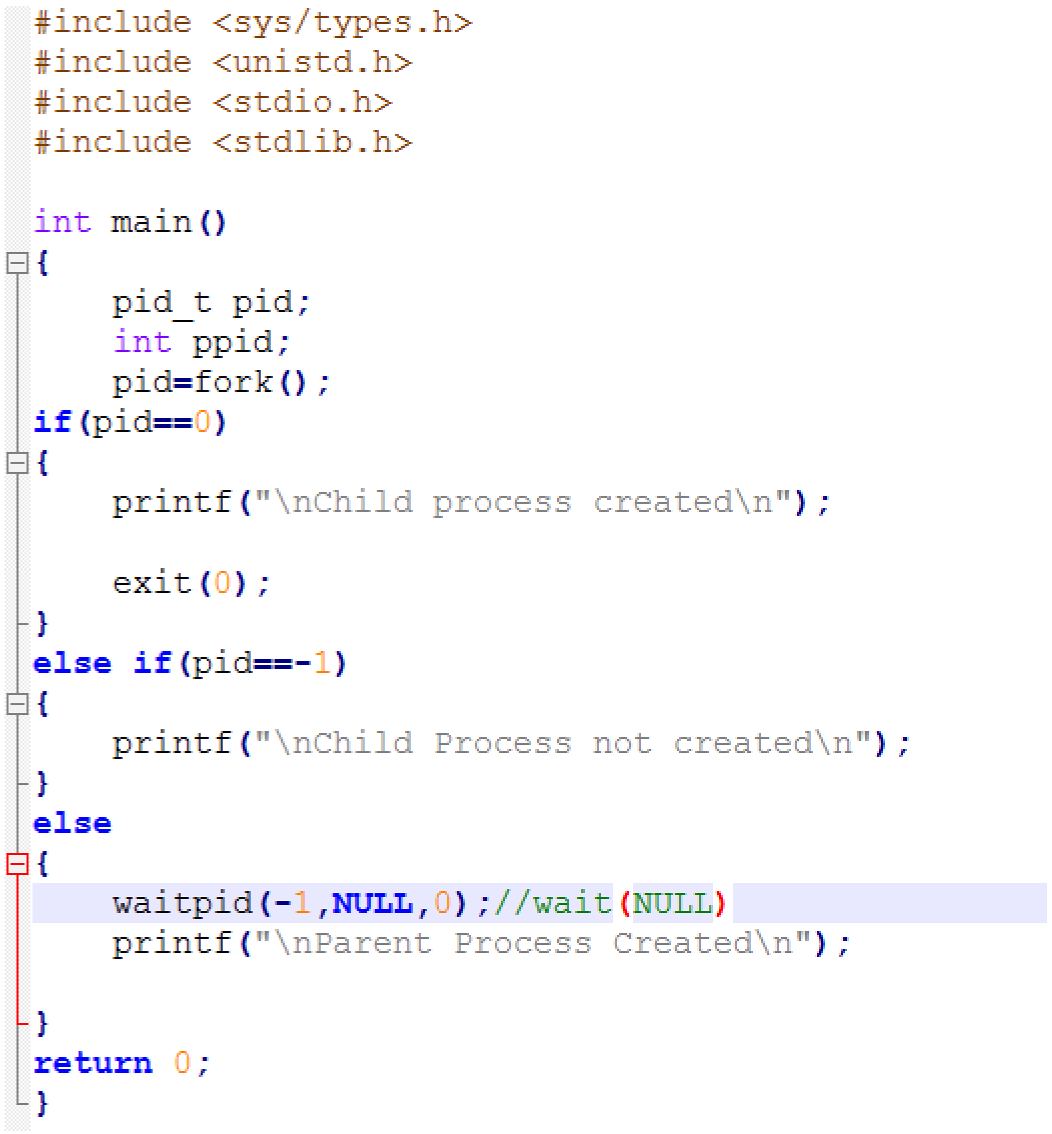


```
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2/3$ ./a.out
child 20 terminated normally with exit status =100
child 21 terminated normally with exit status =101
child 22 terminated normally with exit status =102
child 23 terminated normally with exit status =103
child 24 terminated normally with exit status =104
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2/3$
```

Explanation: The **waitpid** () system call suspends execution of the current process until a child specified by *pid* argument has changed state. By default, **waitpid** () waits only for terminated children, but this behavior is modifiable via the *options* argument
WIFEXITED(Status): The **waitpid** () system call suspends execution of the current process until a child specified by *pid* argument has changed state. By default, **waitpid** () waits only for terminated children, but this behavior is modifiable via the *options* argument

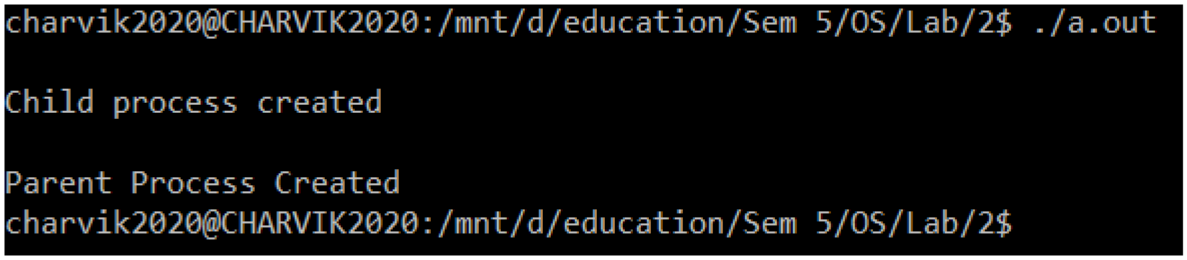
4. Create one child process and make sure that child process runs first and then parent executes and vice versa (if it is possible).

Code:



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

int main()
{
    pid_t pid;
    int ppid;
    pid=fork();
if(pid==0)
{
    printf("\nChild process created\n");

    exit(0);
}
else if(pid==-1)
{
    printf("\nChild Process not created\n");
}
else
{
    waitpid(-1,NULL,0);//wait(NULL)
    printf("\nParent Process Created\n");

}
return 0;
}
```

Output:

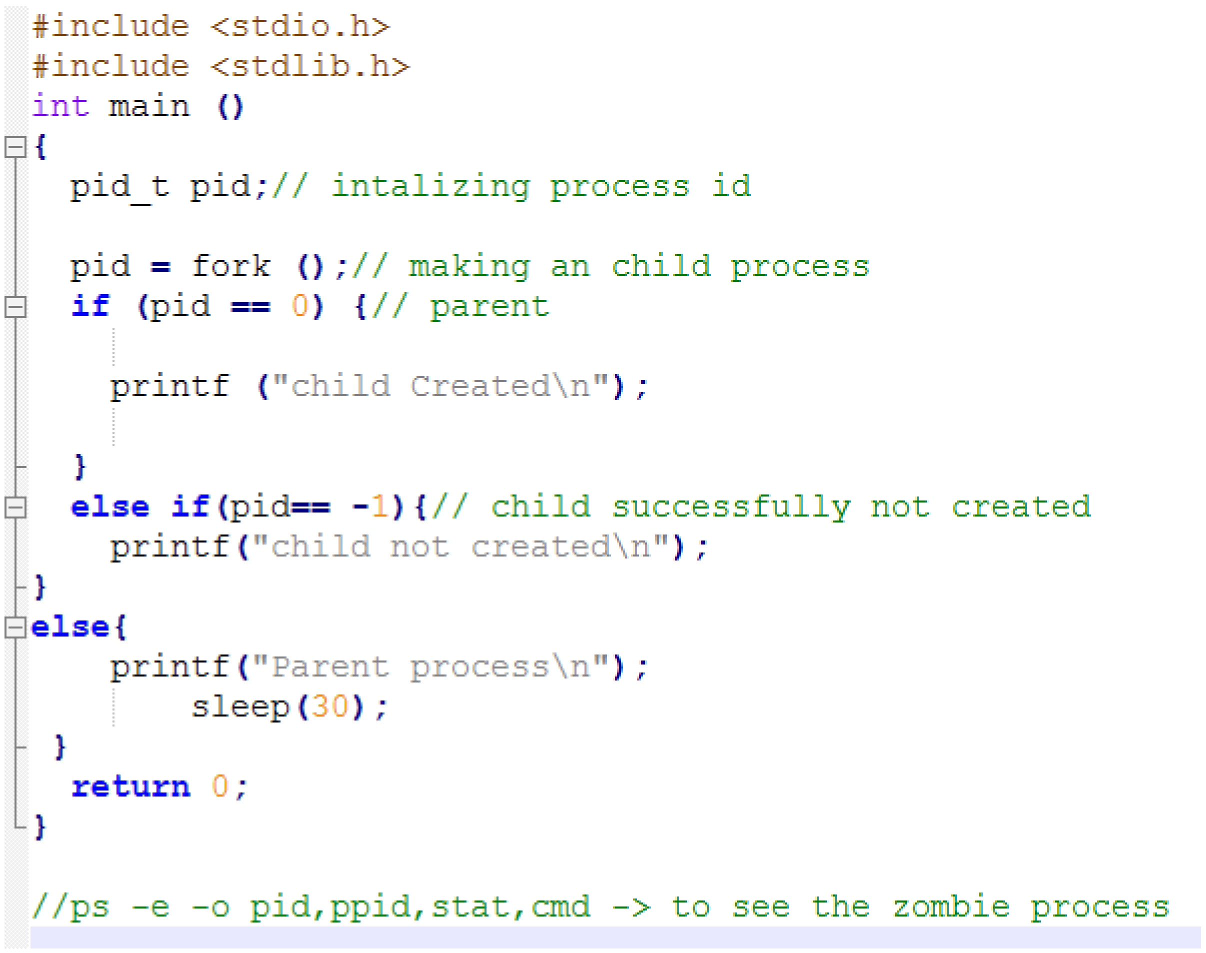


```
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2$ ./a.out

Child process created

Parent Process Created
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2$
```

Not able to do vice-versa

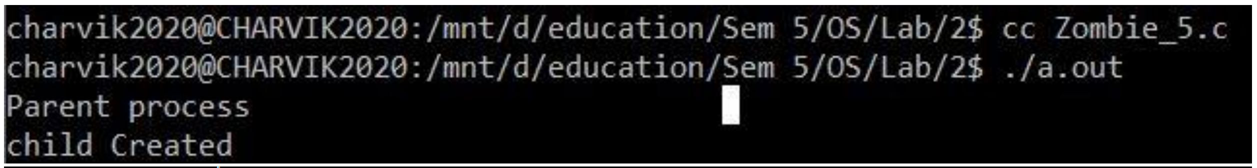
5. Create one child process and make it as zombie. (Use sleep(time);)

Code:

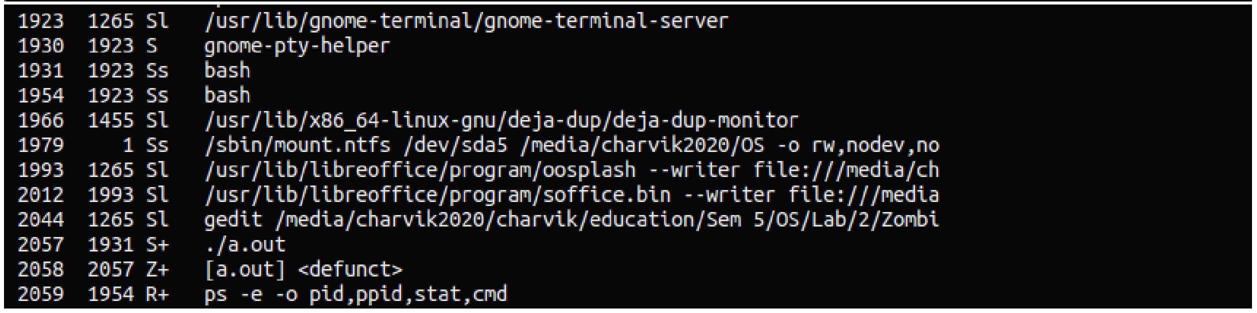


```c
#include <stdio.h>
#include <stdlib.h>
int main ()
{
  pid_t pid;// intalizing process id

  pid = fork ();// making an child process
  if (pid == 0) {// parent

    printf ("child Created\n");

  }
  else if(pid== -1){// child successfully not created
    printf("child not created\n");
}
else{
    printf("Parent process\n");
        sleep(30);
 }
  return 0;
}

//ps -e -o pid,ppid,stat,cmd -> to see the zombie process
```

Output:



```
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2$ cc Zombie_5.c
charvik2020@CHARVIK2020:/mnt/d/education/Sem 5/OS/Lab/2$ ./a.out
Parent process
child Created
```



```
 1923  1265 Sl   /usr/lib/gnome-terminal/gnome-terminal-server
 1930  1923 S    gnome-pty-helper
 1931  1923 Ss   bash
 1954  1923 Ss   bash
 1966  1455 Sl   /usr/lib/x86_64-linux-gnu/deja-dup/deja-dup-monitor
 1979     1 Ss   /sbin/mount.ntfs /dev/sda5 /media/charvik2020/OS -o rw,nodev,no
 1993  1265 Sl   /usr/lib/libreoffice/program/oosplash --writer file:///media/ch
 2012  1993 Sl   /usr/lib/libreoffice/program/soffice.bin --writer file:///media
 2044  1265 Sl   gedit /media/charvik2020/charvik/education/Sem 5/OS/Lab/2/Zombi
 2057  1931 S+   ./a.out
 2058  2057 Z+   [a.out] <defunct>
 2059  1954 R+   ps -e -o pid,ppid,stat,cmd
```