**Program**

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

#include <string.h>

#include <errno.h>

#define BUFFER\_SIZE 1024

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

{

    FILE \*fp1, \*fp2;

    char buf[BUFFER\_SIZE];

    fp1 = fopen(argv[1], "r");

    fp2 = fopen(argv[2], "w");

    while (fgets(buf, BUFFER\_SIZE, fp1) != NULL)

        fputs(buf, fp2);

    return 0;

}

**Output**

**os-lab> cat a.txt**

Graph Thoery

Operating Systems

Computer Architecture and Organization

DBMS

Constitiution of India

Professional Ethics

**os-lab> cat b.txt**

**os-lab> ./a.out a.txt b.txt**

**os-lab> cat b.txt**

Graph Thoery

Operating Systems

Computer Architecture and Organization

DBMS

Constitiution of India

Professional Ethics

**Program**

#include <stdio.h>

#include <stdlib.h>

#include <dirent.h>

#include <string.h>

#include <errno.h>

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

{

    DIR \*p;

    struct dirent \*d;

    p = opendir(argv[1]);

    if (p == NULL)

    {

        printf("Error: %s\n", strerror(errno));

        exit(1);

    }

    while ((d = readdir(p)) != NULL)

        printf("%s\t", d->d\_name);

    printf("\n");

    return 0;

}

**Output**

**os-lab> ./a.out /**

boot media dev opt var lib64 bin home libx32 lib32 . etc root srv sbin proc init lost+found sys usr Docker lib snap .. run mnt tmp

**os-lab> ./a.out .**

. .. a.out cp.c grep.c ls.c

GREP

**Program**

echo "Enter the value of n"

read n

a=0

b=1

echo "$a"

echo "$b"

for ((i=2; i<n; i++))

do

    c=$((a + b))

    echo "$c"

    a=$b

    b=$c

done

**Output**

**os-lab> ./fibonacci.sh**

Enter the value of n

10

0

1

1

2

3

5

8

13

21

34

**Program**

echo "Enter a number: "

read n

for((i=2; i<=n/2; i++))

do

    if [ $((n%i)) -eq 0 ]

    then

        echo "$n is not a prime number."

        exit

    fi

done

echo "$n is a prime number."

**Output**

**os-lab> ./prime.sh**

Enter a number:

5

5 is a prime number.

**os-lab> ./prime.sh**

Enter a number:

6

6 is not a prime number.

**Program**

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

#include <sys/types.h>

int main()

{

    int r = fork();

    if (r == -1)

    {

        printf("Error in process\n");

        exit(1);

    }

    else if (r == 0)

    {

        int pid = getpid();

        printf("Successfully forked process\n");

        printf("PID: %d\n", pid);

    }

    printf("Program to demonstrate fork()\n");

    return 0;

}

**Output**

**os-lab> ./a.out**

Program to demonstrate fork()

Successfully forked process

PID: 750

Program to demonstrate fork()