Laboratory Assignment 2
Subject: Design Principles of Operating Systems
Subject code: CSE 3249

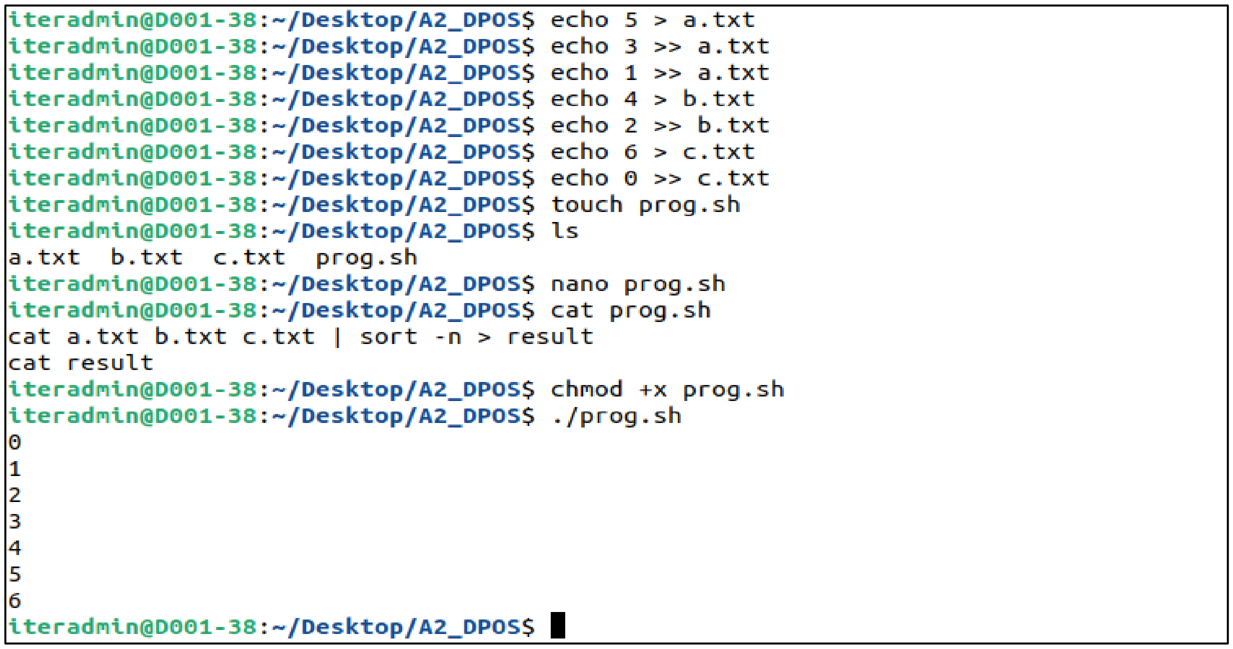
Assignment 2: Familiarization with basic Commands in Unix Operating System and Shell Programming

Objective of this Assignment:

- To learn basic concepts of shell programming
- To lean concept of command line argument in shell script.

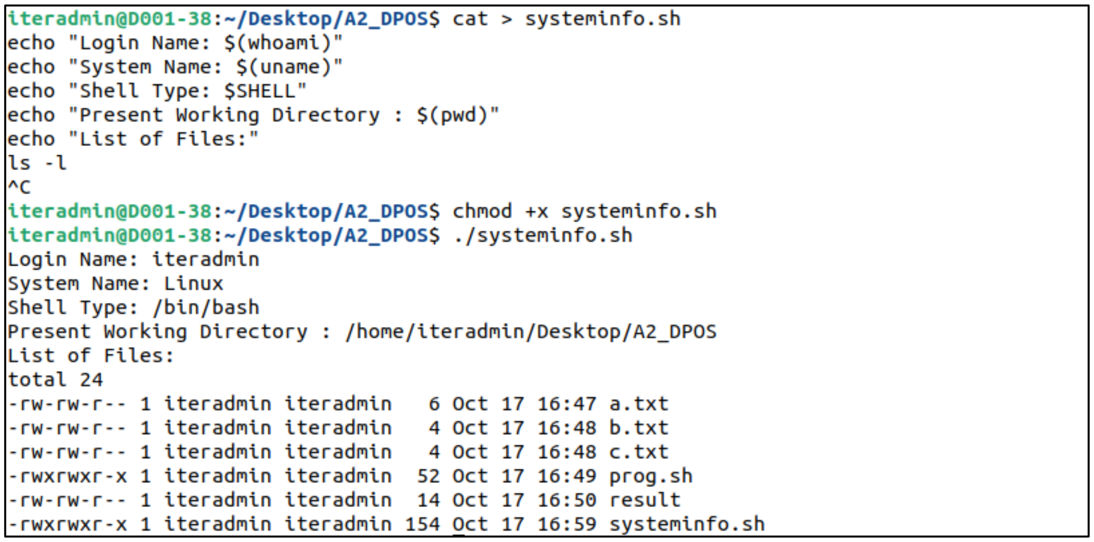
1. Write a shell script named as **prog** for merge the content of files a.txt, b.txt, and c.txt sort them and save the result in a file called **result** and display the sorted output on the screen.

   (Note: a.txt, b.txt and c.txt file contain some numerical value. Make the script an executable file and run it as a command using its name only.)



```
iteradmin@D001-38:~/Desktop/A2_DPOS$ echo 5 > a.txt
iteradmin@D001-38:~/Desktop/A2_DPOS$ echo 3 >> a.txt
iteradmin@D001-38:~/Desktop/A2_DPOS$ echo 1 >> a.txt
iteradmin@D001-38:~/Desktop/A2_DPOS$ echo 4 > b.txt
iteradmin@D001-38:~/Desktop/A2_DPOS$ echo 2 >> b.txt
iteradmin@D001-38:~/Desktop/A2_DPOS$ echo 6 > c.txt
iteradmin@D001-38:~/Desktop/A2_DPOS$ echo 0 >> c.txt
iteradmin@D001-38:~/Desktop/A2_DPOS$ touch prog.sh
iteradmin@D001-38:~/Desktop/A2_DPOS$ ls
a.txt  b.txt  c.txt  prog.sh
iteradmin@D001-38:~/Desktop/A2_DPOS$ nano prog.sh
iteradmin@D001-38:~/Desktop/A2_DPOS$ cat prog.sh
cat a.txt b.txt c.txt | sort -n > result
cat result
iteradmin@D001-38:~/Desktop/A2_DPOS$ chmod +x prog.sh
iteradmin@D001-38:~/Desktop/A2_DPOS$ ./prog.sh
0
1
2
3
4
5
6
iteradmin@D001-38:~/Desktop/A2_DPOS$ 
```

2. Write a shell script named as **systeminfo** that will display the information about the login name of the user, name of the Unix system used by the user, type of the SHELL, Path of current working directory of the user and list of file contain in current working directory. (Make the script an executable file and run it as a command using its name only.)



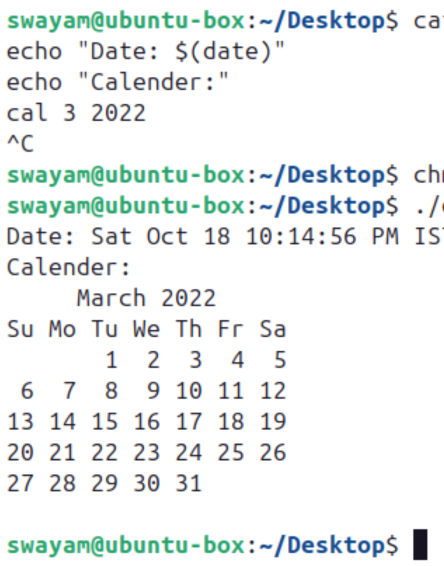
```
iteradmin@D001-38:~/Desktop/A2_DPOS$ cat > systeminfo.sh
echo "Login Name: $(whoami)"
echo "System Name: $(uname)"
echo "Shell Type: $SHELL"
echo "Present Working Directory : $(pwd)"
echo "List of Files:"
ls -l
^C
iteradmin@D001-38:~/Desktop/A2_DPOS$ chmod +x systeminfo.sh
iteradmin@D001-38:~/Desktop/A2_DPOS$ ./systeminfo.sh
Login Name: iteradmin
System Name: Linux
Shell Type: /bin/bash
Present Working Directory : /home/iteradmin/Desktop/A2_DPOS
List of Files:
total 24
-rw-rw-r-- 1 iteradmin iteradmin   6 Oct 17 16:47 a.txt
-rw-rw-r-- 1 iteradmin iteradmin   4 Oct 17 16:48 b.txt
-rw-rw-r-- 1 iteradmin iteradmin   4 Oct 17 16:48 c.txt
-rwxrwxr-x 1 iteradmin iteradmin  52 Oct 17 16:49 prog.sh
-rw-rw-r-- 1 iteradmin iteradmin  14 Oct 17 16:50 result
-rwxrwxr-x 1 iteradmin iteradmin 154 Oct 17 16:59 systeminfo.sh
```

3. Write a shell script named as **dtcal** for displaying both the system date and calendar for specific month, say march 2022, in the given format:-

   Date : specific date Calender

   : current calendar

   (Make the script an executable file and run it as a command using its name only.)



```
swayam@ubuntu-box:~/Desktop$ cat > dtcal.sh
echo "Date: $(date)"
echo "Calender:"
cal 3 2022
^C
swayam@ubuntu-box:~/Desktop$ chmod +x dtcal.sh
swayam@ubuntu-box:~/Desktop$ ./dtcal.sh
Date: Sat Oct 18 10:14:56 PM IST 2025
Calender:
     March 2022
Su Mo Tu We Th Fr Sa
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30 31

swayam@ubuntu-box:~/Desktop$
```

4. Write a shell script named as **nvwc** which will display the filename and linecount, wordcount and char count of the file dtcal in the following format:

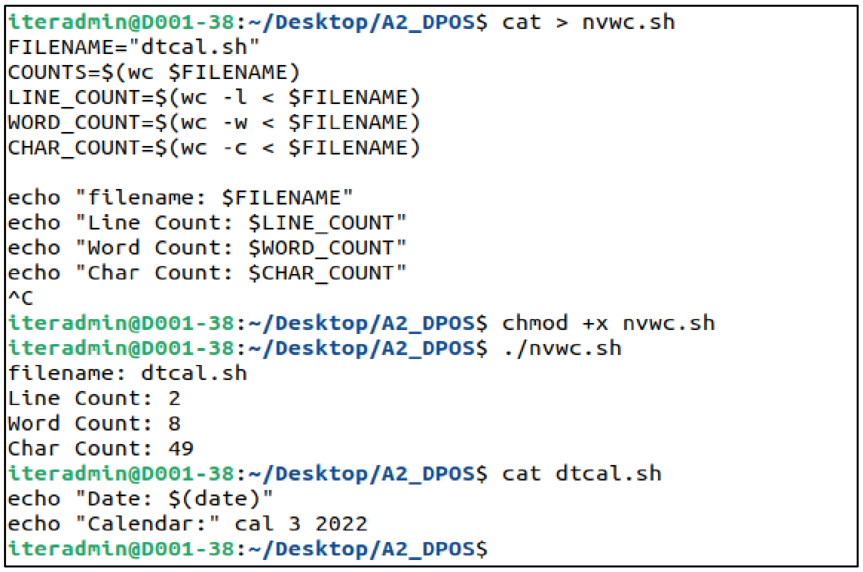
   Filename: dtcal

   Line count: -

   Word count: -

   Charcout: -

   (Make the script an executable file and run it as a command using its name only.)

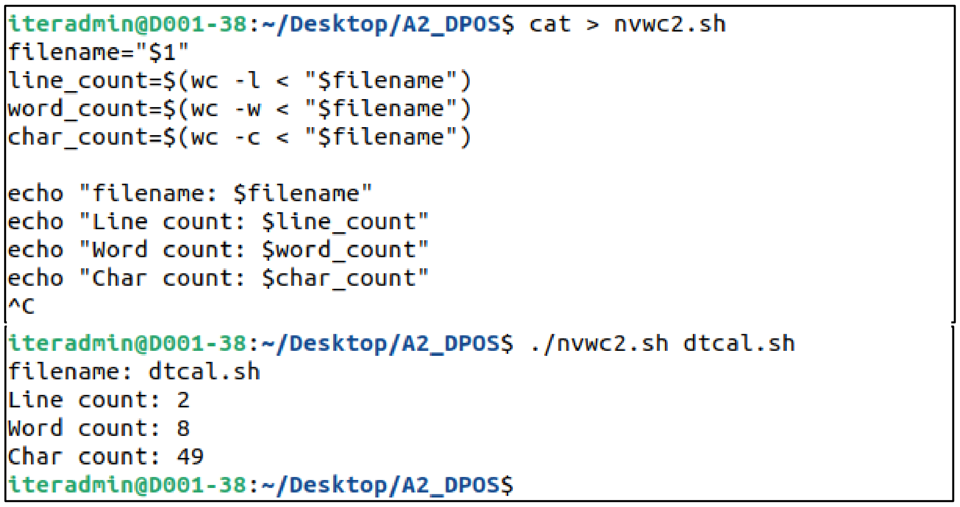


```
iteradmin@D001-38:~/Desktop/A2_DPOS$ cat > nvwc.sh
FILENAME="dtcal.sh"
COUNTS=$(wc $FILENAME)
LINE_COUNT=$(wc -l < $FILENAME)
WORD_COUNT=$(wc -w < $FILENAME)
CHAR_COUNT=$(wc -c < $FILENAME)

echo "filename: $FILENAME"
echo "Line Count: $LINE_COUNT"
echo "Word Count: $WORD_COUNT"
echo "Char Count: $CHAR_COUNT"
^C
iteradmin@D001-38:~/Desktop/A2_DPOS$ chmod +x nvwc.sh
iteradmin@D001-38:~/Desktop/A2_DPOS$ ./nvwc.sh
filename: dtcal.sh
Line Count: 2
Word Count: 8
Char Count: 49
iteradmin@D001-38:~/Desktop/A2_DPOS$ cat dtcal.sh
echo "Date: $(date)"
echo "Calendar:" cal 3 2022
iteradmin@D001-38:~/Desktop/A2_DPOS$
```

5. Write a shell script named as **nvwc2** which will display the filename and linecount, word count and char count of **any file** given as argument to nvwc2 in the following format:

```
          filename        linecount       wordcount       charcount

          file1               -               -               -
```

(Make the script an executable file and run it as a command using its name only.)

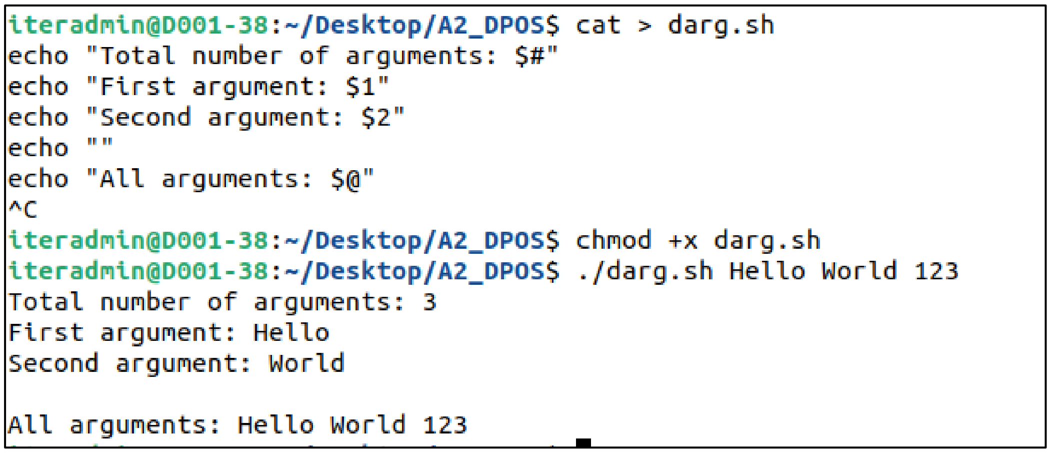


```
iteradmin@D001-38:~/Desktop/A2_DPOS$ cat > nvwc2.sh
filename="$1"
line_count=$(wc -l < "$filename")
word_count=$(wc -w < "$filename")
char_count=$(wc -c < "$filename")

echo "filename: $filename"
echo "Line count: $line_count"
echo "Word count: $word_count"
echo "Char count: $char_count"
^C
iteradmin@D001-38:~/Desktop/A2_DPOS$ ./nvwc2.sh dtcal.sh
filename: dtcal.sh
Line count: 2
Word count: 8
Char count: 49
iteradmin@D001-38:~/Desktop/A2_DPOS$
```

6. Write a shell script named as **darg** to display the total number of command line arguments along with the first two arguments.

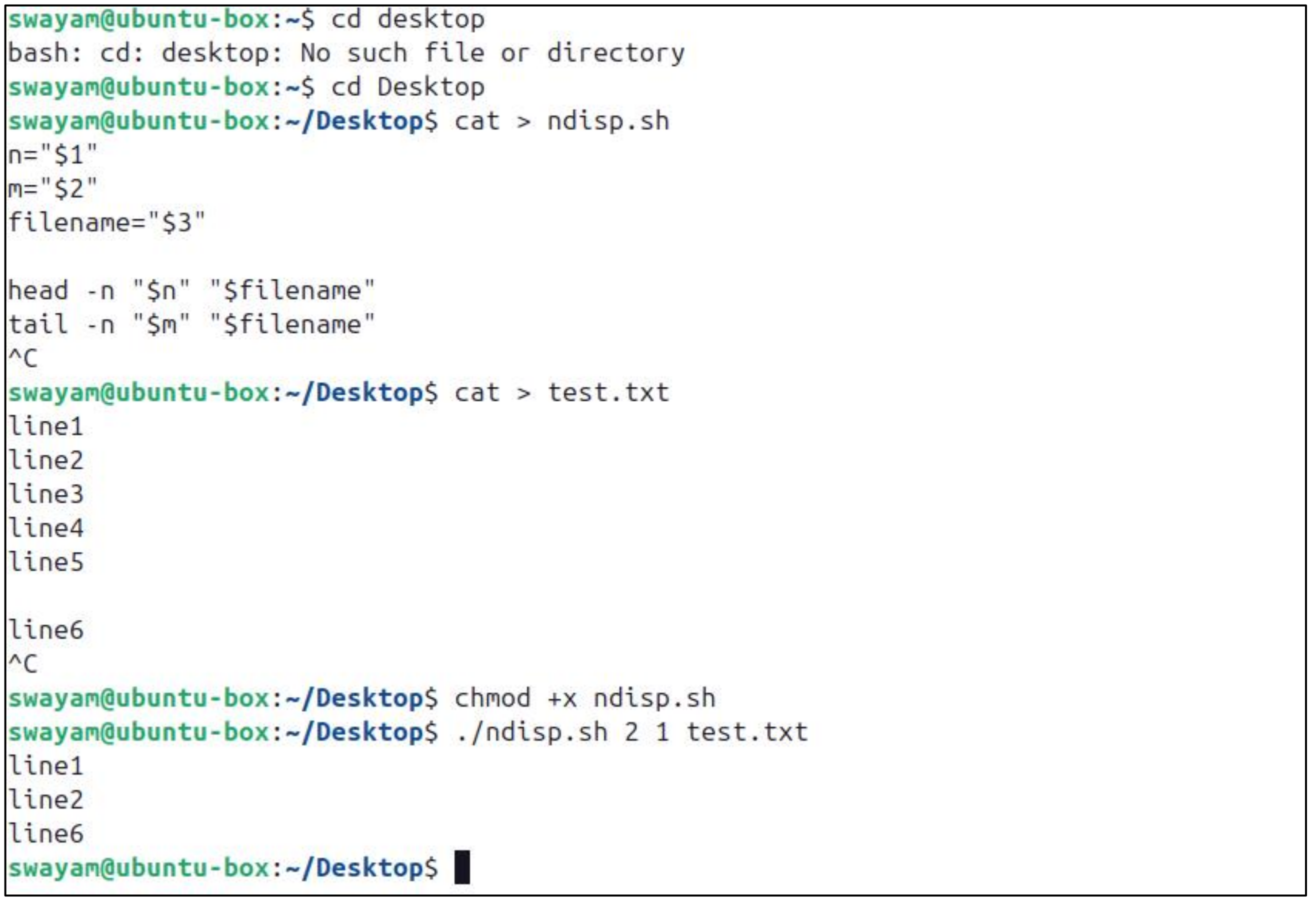
-Modify the script to display all the arguments.

(Make the script an executable file and run it as a command using its name only.)



```
iteradmin@D001-38:~/Desktop/A2_DPOS$ cat > darg.sh
echo "Total number of arguments: $#"
echo "First argument: $1"
echo "Second argument: $2"
echo ""
echo "All arguments: $@"
^C
iteradmin@D001-38:~/Desktop/A2_DPOS$ chmod +x darg.sh
iteradmin@D001-38:~/Desktop/A2_DPOS$ ./darg.sh Hello World 123
Total number of arguments: 3
First argument: Hello
Second argument: World

All arguments: Hello World 123
```

7. Write a shell script named as **ndisp** that will take three command line arguments specifying the value of n, m and a filename and display the first n number of lines and last m number of lines of the file given as argument.
(Make the script an executable file and run it as a command using its name only.)



```
swayam@ubuntu-box:~$ cd desktop
bash: cd: desktop: No such file or directory
swayam@ubuntu-box:~$ cd Desktop
swayam@ubuntu-box:~/Desktop$ cat > ndisp.sh
n="$1"
m="$2"
filename="$3"

head -n "$n" "$filename"
tail -n "$m" "$filename"
^C
swayam@ubuntu-box:~/Desktop$ cat > test.txt
line1
line2
line3
line4
line5

line6
^C
swayam@ubuntu-box:~/Desktop$ chmod +x ndisp.sh
swayam@ubuntu-box:~/Desktop$ ./ndisp.sh 2 1 test.txt
line1
line2
line6
swayam@ubuntu-box:~/Desktop$ 
```

Laboratory Assignments 3

Subject: Design Principles of Operating Systems

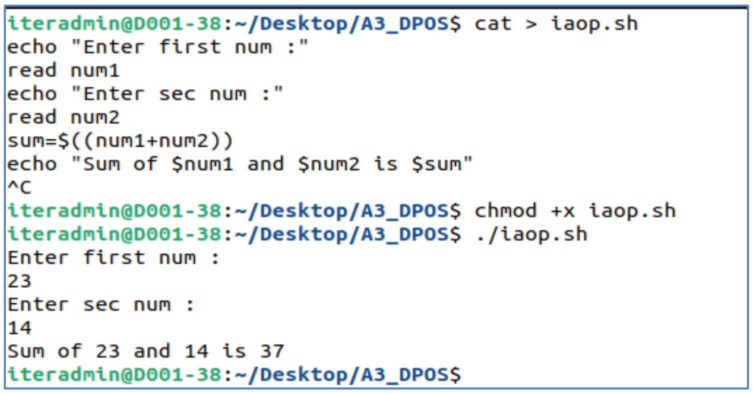
Subject code: CSE 3249

Assignment 3: Shell Programming using user defined variables, arithmetic operators, conditional statements.

Objective of this Assignment:

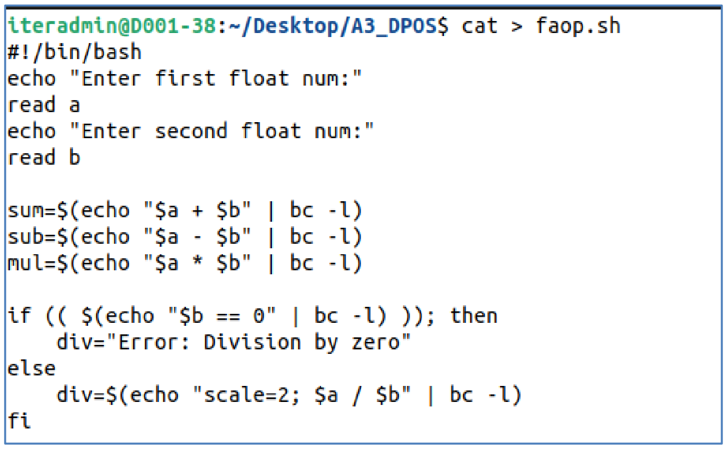
- To learn the proper use of user defined variables and arithmetic operators in shell programming.
- To write shell script producing solution to decision making problems.

1. Write a shell script **iaop** to perform integer arithmetic on two numbers, where the value of the two numbers will be given during runtime.

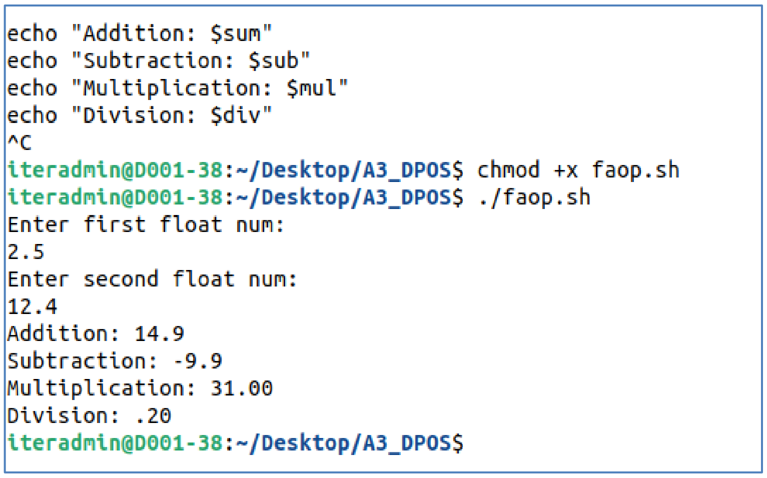


```
iteradmin@D001-38:~/Desktop/A3_DPOS$ cat > iaop.sh
echo "Enter first num :"
read num1
echo "Enter sec num :"
read num2
sum=$((num1+num2))
echo "Sum of $num1 and $num2 is $sum"
^C
iteradmin@D001-38:~/Desktop/A3_DPOS$ chmod +x iaop.sh
iteradmin@D001-38:~/Desktop/A3_DPOS$ ./iaop.sh
Enter first num :
23
Enter sec num :
14
Sum of 23 and 14 is 37
iteradmin@D001-38:~/Desktop/A3_DPOS$
```

2. Write a shell script **faop** to perform floating point arithmetic on two numbers, where the value of the two numbers will be given during runtime.

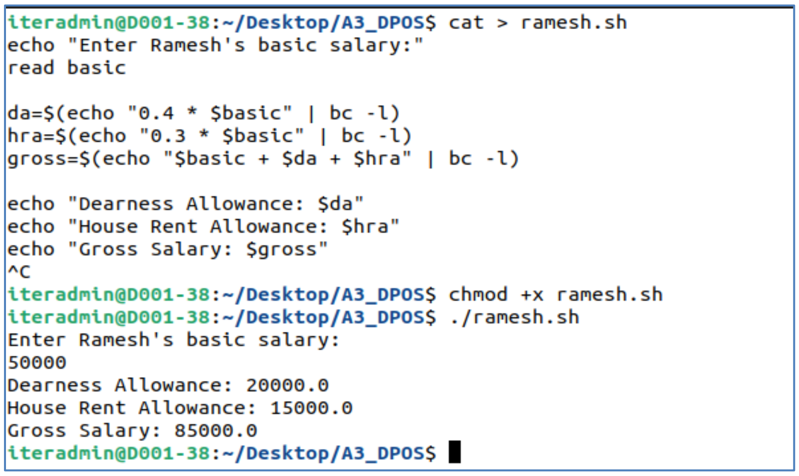


```
iteradmin@D001-38:~/Desktop/A3_DPOS$ cat > faop.sh
#!/bin/bash
echo "Enter first float num:"
read a
echo "Enter second float num:"
read b

sum=$(echo "$a + $b" | bc -l)
sub=$(echo "$a - $b" | bc -l)
mul=$(echo "$a * $b" | bc -l)

if (( $(echo "$b == 0" | bc -l) )); then
    div="Error: Division by zero"
else
    div=$(echo "scale=2; $a / $b" | bc -l)
fi
```



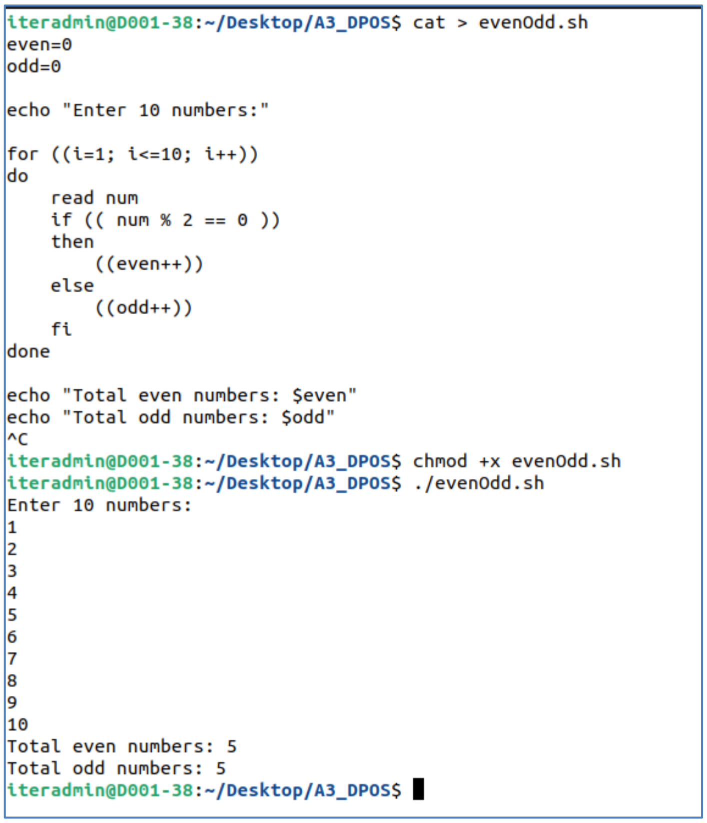
```
echo "Addition: $sum"
echo "Subtraction: $sub"
echo "Multiplication: $mul"
echo "Division: $div"
^C
iteradmin@D001-38:~/Desktop/A3_DPOS$ chmod +x faop.sh
iteradmin@D001-38:~/Desktop/A3_DPOS$ ./faop.sh
Enter first float num:
2.5
Enter second float num:
12.4
Addition: 14.9
Subtraction: -9.9
Multiplication: 31.00
Division: .20
iteradmin@D001-38:~/Desktop/A3_DPOS$
```

3. Ramesh's basic salary is input through the keyboard. His dearness allowance is 40% of basic salary, and house rent allowance is 30% of basic salary. Write a program to calculate his gross salary.



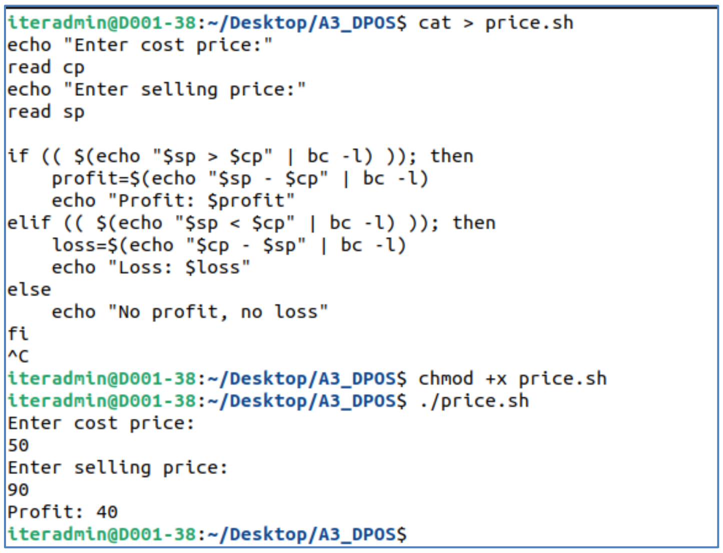
```
iteradmin@D001-38:~/Desktop/A3_DPOS$ cat > ramesh.sh
echo "Enter Ramesh's basic salary:"
read basic

da=$(echo "0.4 * $basic" | bc -l)
hra=$(echo "0.3 * $basic" | bc -l)
gross=$(echo "$basic + $da + $hra" | bc -l)

echo "Dearness Allowance: $da"
echo "House Rent Allowance: $hra"
echo "Gross Salary: $gross"
^C
iteradmin@D001-38:~/Desktop/A3_DPOS$ chmod +x ramesh.sh
iteradmin@D001-38:~/Desktop/A3_DPOS$ ./ramesh.sh
Enter Ramesh's basic salary:
50000
Dearness Allowance: 20000.0
House Rent Allowance: 15000.0
Gross Salary: 85000.0
iteradmin@D001-38:~/Desktop/A3_DPOS$
```

4. Write a shell script to accept a list of 10 numbers from the user and count how many are **even** and how many are **odd**.



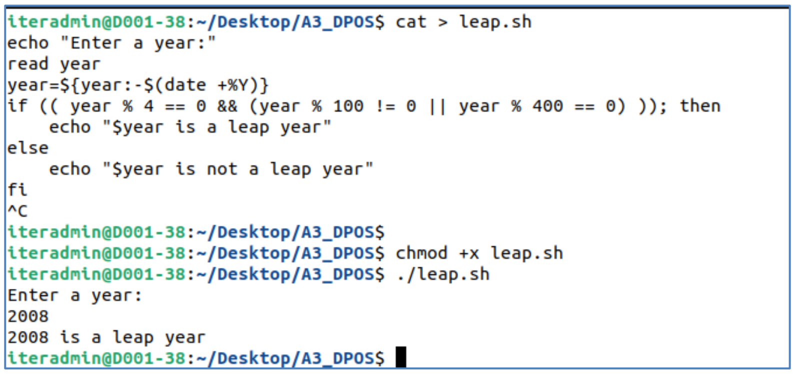
```
iteradmin@D001-38:~/Desktop/A3_DPOS$ cat > evenOdd.sh
even=0
odd=0

echo "Enter 10 numbers:"

for ((i=1; i<=10; i++))
do
    read num
    if (( num % 2 == 0 ))
    then
        ((even++))
    else
        ((odd++))
    fi
done

echo "Total even numbers: $even"
echo "Total odd numbers: $odd"
^C
iteradmin@D001-38:~/Desktop/A3_DPOS$ chmod +x evenOdd.sh
iteradmin@D001-38:~/Desktop/A3_DPOS$ ./evenOdd.sh
Enter 10 numbers:
1
2
3
4
5
6
7
8
9
10
Total even numbers: 5
Total odd numbers: 5
iteradmin@D001-38:~/Desktop/A3_DPOS$
```

5. If cost price and selling price of an item is input through the keyboard, write a program to determine whether the seller has made profit or incurred loss. Also determine how much profit was made or loss incurred.



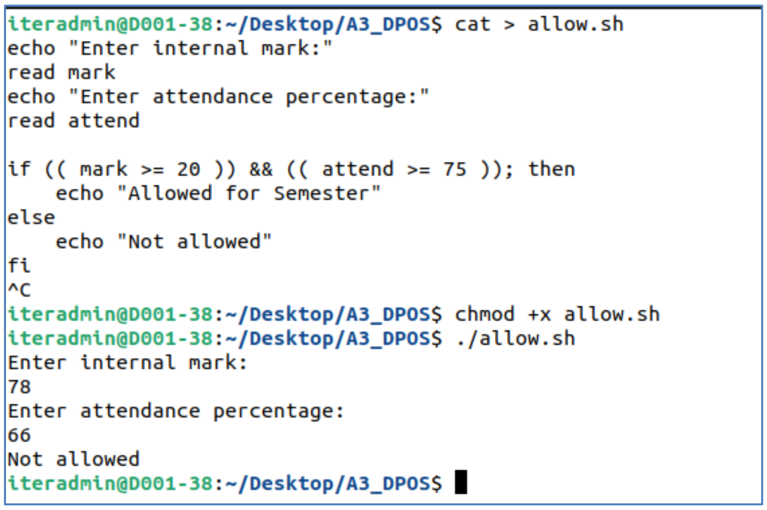
```
iteradmin@D001-38:~/Desktop/A3_DPOS$ cat > price.sh
echo "Enter cost price:"
read cp
echo "Enter selling price:"
read sp

if (( $(echo "$sp > $cp" | bc -l) )); then
    profit=$(echo "$sp - $cp" | bc -l)
    echo "Profit: $profit"
elif (( $(echo "$sp < $cp" | bc -l) )); then
    loss=$(echo "$cp - $sp" | bc -l)
    echo "Loss: $loss"
else
    echo "No profit, no loss"
fi
^C
iteradmin@D001-38:~/Desktop/A3_DPOS$ chmod +x price.sh
iteradmin@D001-38:~/Desktop/A3_DPOS$ ./price.sh
Enter cost price:
50
Enter selling price:
90
Profit: 40
iteradmin@D001-38:~/Desktop/A3_DPOS$
```

6. Write a shell script which receives any year from the keyboard and determines, whether the year is a leap year or not. If no argument is supplied the current year should be assumed.



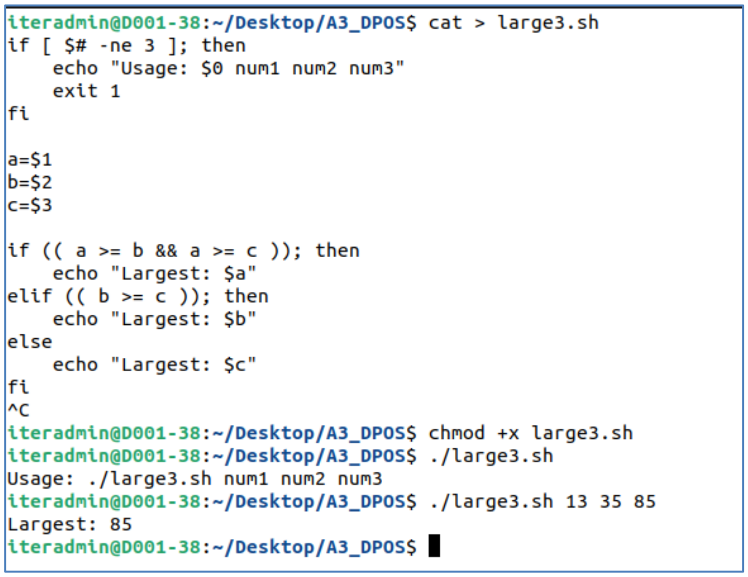
```
iteradmin@D001-38:~/Desktop/A3_DPOS$ cat > leap.sh
echo "Enter a year:"
read year
year=${year:-$(date +%Y)}
if (( year % 4 == 0 && (year % 100 != 0 || year % 400 == 0) )); then
    echo "$year is a leap year"
else
    echo "$year is not a leap year"
fi
^C
iteradmin@D001-38:~/Desktop/A3_DPOS$
iteradmin@D001-38:~/Desktop/A3_DPOS$ chmod +x leap.sh
iteradmin@D001-38:~/Desktop/A3_DPOS$ ./leap.sh
Enter a year:
2008
2008 is a leap year
iteradmin@D001-38:~/Desktop/A3_DPOS$
```

7. Write a shell script **allow** that will display a message to enter internal mark and percentage in attendance, if the entered mark is greater than equal to 20 and entered percentage in attendance is greater that equal to 75 then display the message Allowed for Semester otherwise display the message Not allowed.



```
iteradmin@D001-38:~/Desktop/A3_DPOS$ cat > allow.sh
echo "Enter internal mark:"
read mark
echo "Enter attendance percentage:"
read attend

if (( mark >= 20 )) && (( attend >= 75 )); then
    echo "Allowed for Semester"
else
    echo "Not allowed"
fi
^C
iteradmin@D001-38:~/Desktop/A3_DPOS$ chmod +x allow.sh
iteradmin@D001-38:~/Desktop/A3_DPOS$ ./allow.sh
Enter internal mark:
78
Enter attendance percentage:
66
Not allowed
iteradmin@D001-38:~/Desktop/A3_DPOS$
```

8. Write a shell script **large3** that will compare three numbers passed as command line arguments and display the largest one.

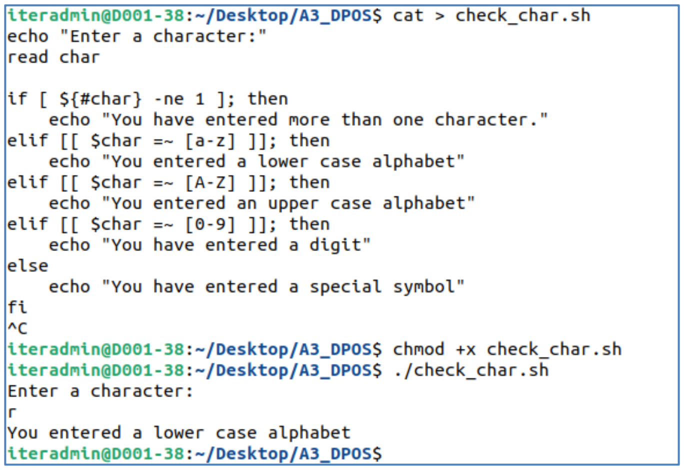


```
iteradmin@D001-38:~/Desktop/A3_DPOS$ cat > large3.sh
if [ $# -ne 3 ]; then
    echo "Usage: $0 num1 num2 num3"
    exit 1
fi

a=$1
b=$2
c=$3

if (( a >= b && a >= c )); then
    echo "Largest: $a"
elif (( b >= c )); then
    echo "Largest: $b"
else
    echo "Largest: $c"
fi
^C
iteradmin@D001-38:~/Desktop/A3_DPOS$ chmod +x large3.sh
iteradmin@D001-38:~/Desktop/A3_DPOS$ ./large3.sh
Usage: ./large3.sh num1 num2 num3
iteradmin@D001-38:~/Desktop/A3_DPOS$ ./large3.sh 13 35 85
Largest: 85
iteradmin@D001-38:~/Desktop/A3_DPOS$ █
```

9. Write a shell script **check_char** which will display one message to enter a character and according to the character entered it will display appropriate message from the following options:

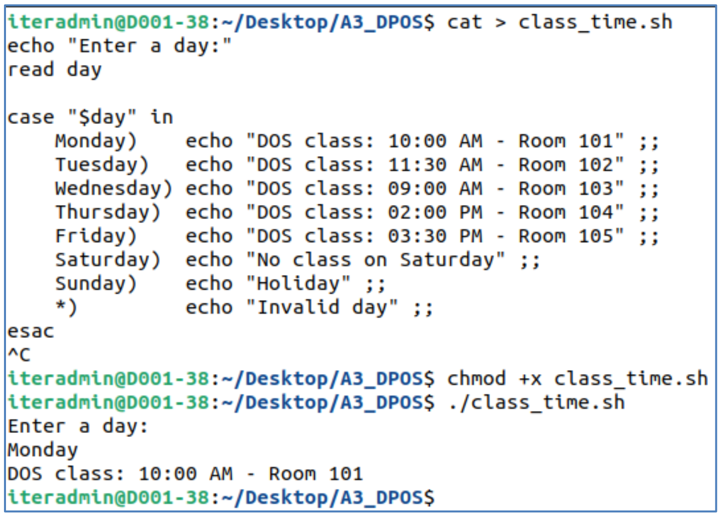
- You entered a lower case alphabet
- You entered an upper case alphabet.
- You have entered a digit.
- You have entered a special symbol.

You have entered more than one character.



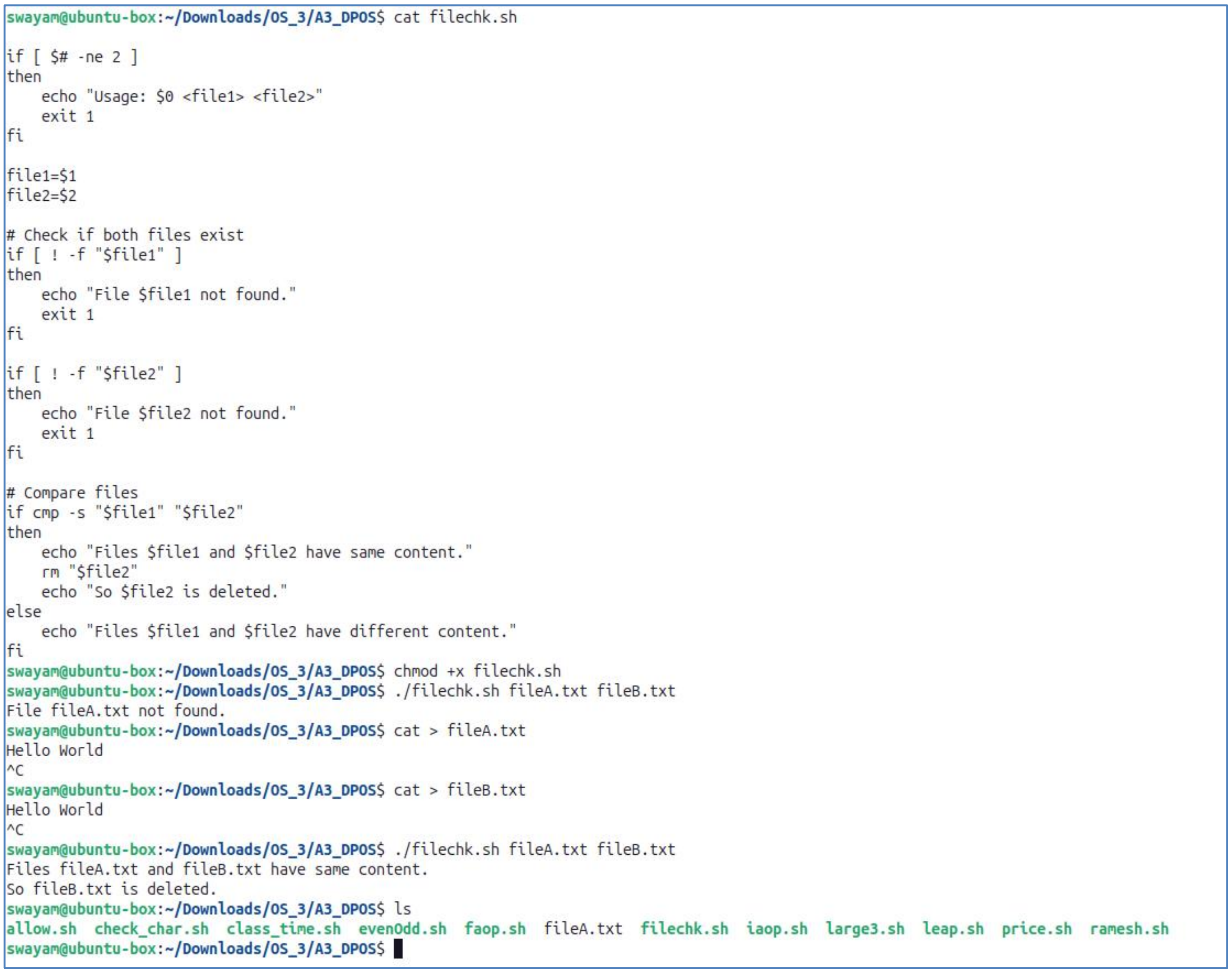
```
iteradmin@D001-38:~/Desktop/A3_DPOS$ cat > check_char.sh
echo "Enter a character:"
read char

if [ ${#char} -ne 1 ]; then
    echo "You have entered more than one character."
elif [[ $char =~ [a-z] ]]; then
    echo "You entered a lower case alphabet"
elif [[ $char =~ [A-Z] ]]; then
    echo "You entered an upper case alphabet"
elif [[ $char =~ [0-9] ]]; then
    echo "You have entered a digit"
else
    echo "You have entered a special symbol"
fi
^C
iteradmin@D001-38:~/Desktop/A3_DPOS$ chmod +x check_char.sh
iteradmin@D001-38:~/Desktop/A3_DPOS$ ./check_char.sh
Enter a character:
r
You entered a lower case alphabet
iteradmin@D001-38:~/Desktop/A3_DPOS$
```

10. Write a shell script **class_time** which will display one message to enter a day andaccording to the day entered it will display the DOS class time along with the room information or the message “No class on day_name” or “Holiday” for Sunday.



```
iteradmin@D001-38:~/Desktop/A3_DPOS$ cat > class_time.sh
echo "Enter a day:"
read day

case "$day" in
    Monday)    echo "DOS class: 10:00 AM - Room 101" ;;
    Tuesday)   echo "DOS class: 11:30 AM - Room 102" ;;
    Wednesday) echo "DOS class: 09:00 AM - Room 103" ;;
    Thursday)  echo "DOS class: 02:00 PM - Room 104" ;;
    Friday)    echo "DOS class: 03:30 PM - Room 105" ;;
    Saturday)  echo "No class on Saturday" ;;
    Sunday)    echo "Holiday" ;;
    *)         echo "Invalid day" ;;
esac
^C
iteradmin@D001-38:~/Desktop/A3_DPOS$ chmod +x class_time.sh
iteradmin@D001-38:~/Desktop/A3_DPOS$ ./class_time.sh
Enter a day:
Monday
DOS class: 10:00 AM - Room 101
iteradmin@D001-38:~/Desktop/A3_DPOS$
```

11. Write a shell script **filechk** that will take two file names as command line arguments, and check whether the content of two files are same or not . If contents of two files are same, then it will display the message: Files filename1 and filename2 have same content.then delete the second file and display the message: So filename2 is deleted. Otherwise display the message: Files filename1 and filename2 have different content.



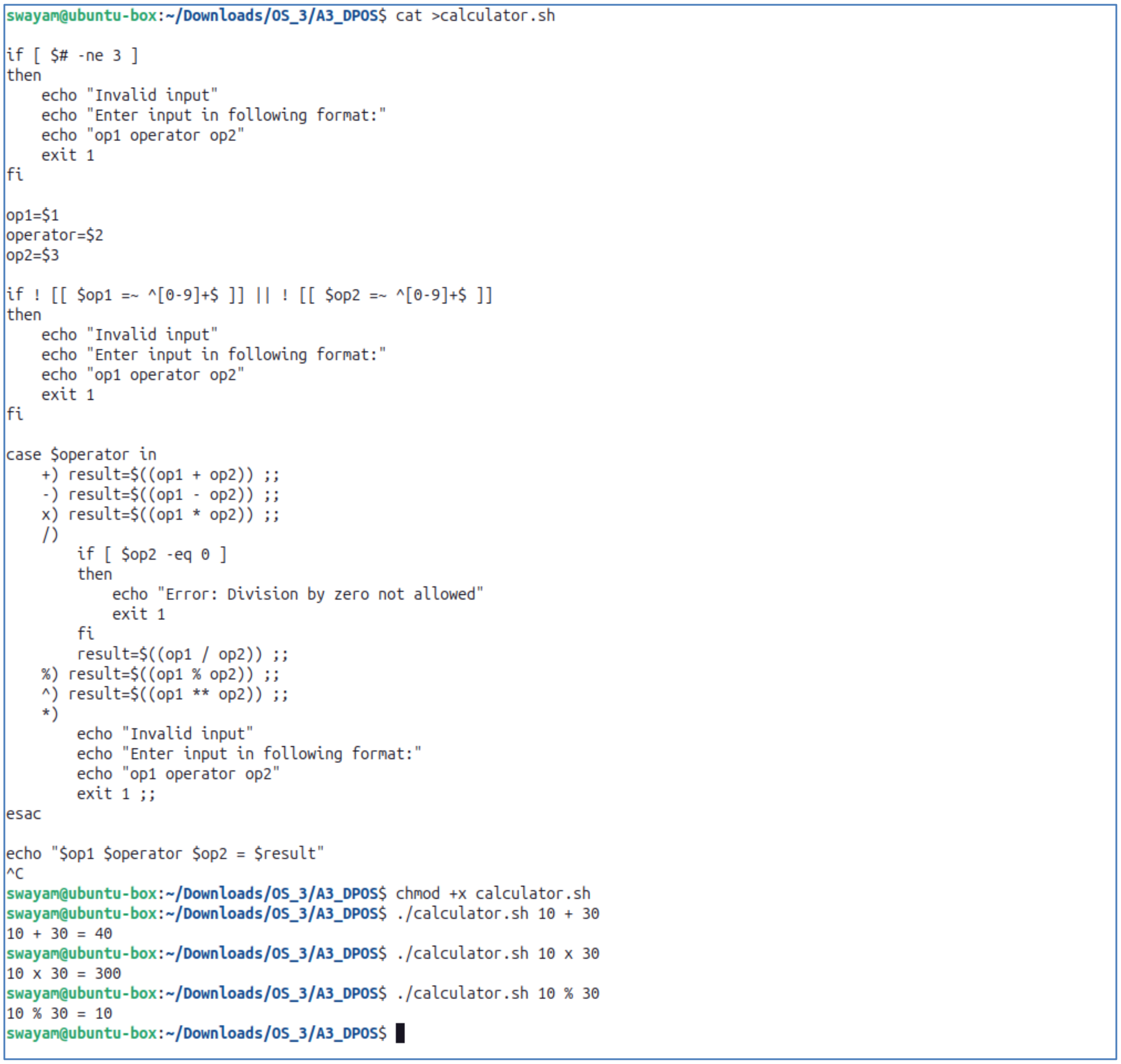
```
swayam@ubuntu-box:~/Downloads/OS_3/A3_DPOS$ cat filechk.sh

if [ $# -ne 2 ]
then
    echo "Usage: $0 <file1> <file2>"
    exit 1
fi

file1=$1
file2=$2

# Check if both files exist
if [ ! -f "$file1" ]
then
    echo "File $file1 not found."
    exit 1
fi

if [ ! -f "$file2" ]
then
    echo "File $file2 not found."
    exit 1
fi

# Compare files
if cmp -s "$file1" "$file2"
then
    echo "Files $file1 and $file2 have same content."
    rm "$file2"
    echo "So $file2 is deleted."
else
    echo "Files $file1 and $file2 have different content."
fi
swayam@ubuntu-box:~/Downloads/OS_3/A3_DPOS$ chmod +x filechk.sh
swayam@ubuntu-box:~/Downloads/OS_3/A3_DPOS$ ./filechk.sh fileA.txt fileB.txt
File fileA.txt not found.
swayam@ubuntu-box:~/Downloads/OS_3/A3_DPOS$ cat > fileA.txt
Hello World
^C
swayam@ubuntu-box:~/Downloads/OS_3/A3_DPOS$ cat > fileB.txt
Hello World
^C
swayam@ubuntu-box:~/Downloads/OS_3/A3_DPOS$ ./filechk.sh fileA.txt fileB.txt
Files fileA.txt and fileB.txt have same content.
So fileB.txt is deleted.
swayam@ubuntu-box:~/Downloads/OS_3/A3_DPOS$ ls
allow.sh  check_char.sh  class_time.sh  evenOdd.sh  faop.sh  fileA.txt  filechk.sh  iaop.sh  large3.sh  leap.sh  price.sh  ramesh.sh
swayam@ubuntu-box:~/Downloads/OS_3/A3_DPOS$ 
```

12. Write a shell script **calculator** that will take three command line arguments, where the first argument will specify the first operand, second argument will specify the operator and the third argument will specify the second operand and display the output of the arithmetic operation specified in the following format: op1 operator op2 = result .

If the arguments will be passed in any other sequence, it will display the message:

"Invalid input "

Enter input in following format: op1 operator op2



```
swayam@ubuntu-box:~/Downloads/OS_3/A3_DPOS$ cat >calculator.sh

if [ $# -ne 3 ]
then
    echo "Invalid input"
    echo "Enter input in following format:"
    echo "op1 operator op2"
    exit 1
fi

op1=$1
operator=$2
op2=$3

if ! [[ $op1 =~ ^[0-9]+$ ]] || ! [[ $op2 =~ ^[0-9]+$ ]]
then
    echo "Invalid input"
    echo "Enter input in following format:"
    echo "op1 operator op2"
    exit 1
fi

case $operator in
    +) result=$((op1 + op2)) ;;
    -) result=$((op1 - op2)) ;;
    x) result=$((op1 * op2)) ;;
    /)
        if [ $op2 -eq 0 ]
        then
            echo "Error: Division by zero not allowed"
            exit 1
        fi
        result=$((op1 / op2)) ;;
    %) result=$((op1 % op2)) ;;
    ^) result=$((op1 ** op2)) ;;
    *)
        echo "Invalid input"
        echo "Enter input in following format:"
        echo "op1 operator op2"
        exit 1 ;;
esac

echo "$op1 $operator $op2 = $result"
^C
swayam@ubuntu-box:~/Downloads/OS_3/A3_DPOS$ chmod +x calculator.sh
swayam@ubuntu-box:~/Downloads/OS_3/A3_DPOS$ ./calculator.sh 10 + 30
10 + 30 = 40
swayam@ubuntu-box:~/Downloads/OS_3/A3_DPOS$ ./calculator.sh 10 x 30
10 x 30 = 300
swayam@ubuntu-box:~/Downloads/OS_3/A3_DPOS$ ./calculator.sh 10 % 30
10 % 30 = 10
swayam@ubuntu-box:~/Downloads/OS_3/A3_DPOS$ 
```

Laboratory Assignments 4

Subject: Design Principles of Operating Systems

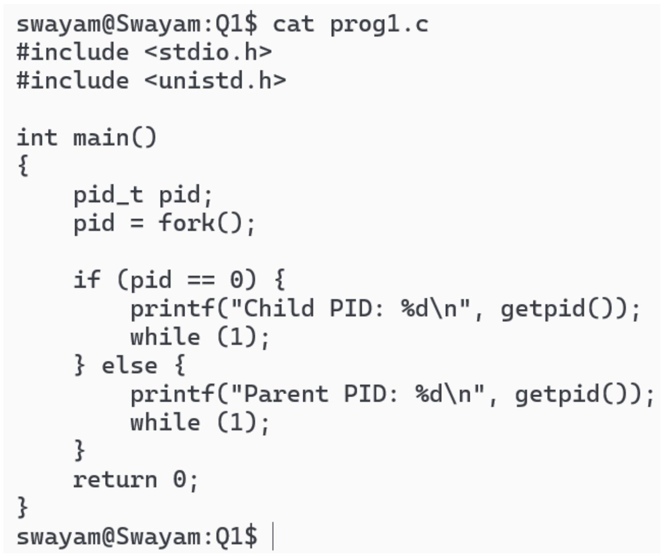
Subject code: CSE 3249

Assignment 4: Familiarization with Process Management in Linux environment.

Objective of this Assignment:

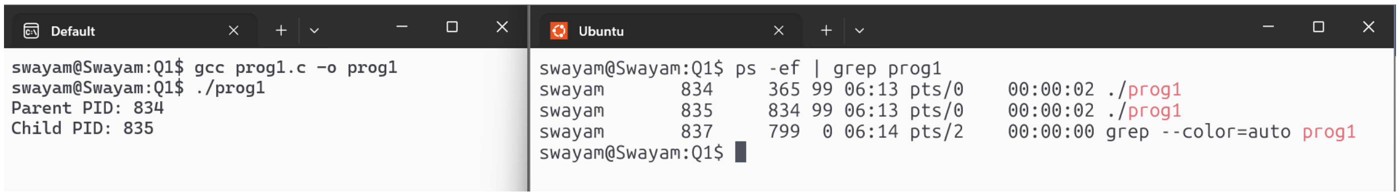
- To trace the different states of a process during its execution.
- To learn the use of different system calls such as (fork(),vfork(),wait(),exec()) for process handling in Unix/Linux environment.

1. Write a C program to create a child process using fork() system call. The child process will print the message “Child” with its process identifier and then continue in an indefinite loop. The parent process will print the message “Parent” with its process identifier and then continue in an indefinite loop.



```
swayam@Swayam:Q1$ cat prog1.c
#include <stdio.h>
#include <unistd.h>

int main()
{
    pid_t pid;
    pid = fork();

    if (pid == 0) {
        printf("Child PID: %d\n", getpid());
        while (1);
    } else {
        printf("Parent PID: %d\n", getpid());
        while (1);
    }
    return 0;
}
swayam@Swayam:Q1$
```

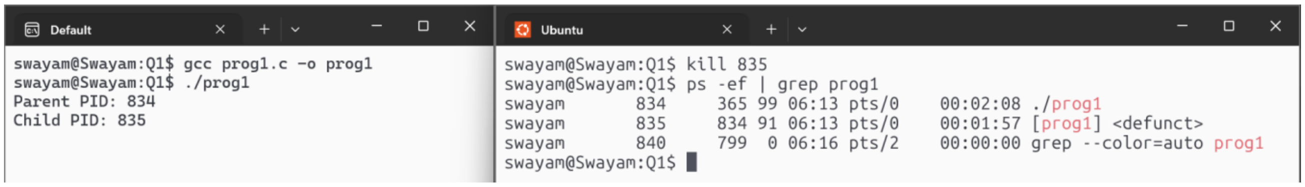
   a) Run the program and trace the state of both processes.



```
swayam@Swayam:Q1$ gcc prog1.c -o prog1
swayam@Swayam:Q1$ ./prog1
Parent PID: 834
Child PID: 835
```

```
swayam@Swayam:Q1$ ps -ef | grep prog1
swayam       834     365 99 06:13 pts/0    00:00:02 ./prog1
swayam       835     834 99 06:13 pts/0    00:00:02 ./prog1
swayam       837     799  0 06:14 pts/2    00:00:00 grep --color=auto prog1
swayam@Swayam:Q1$
```

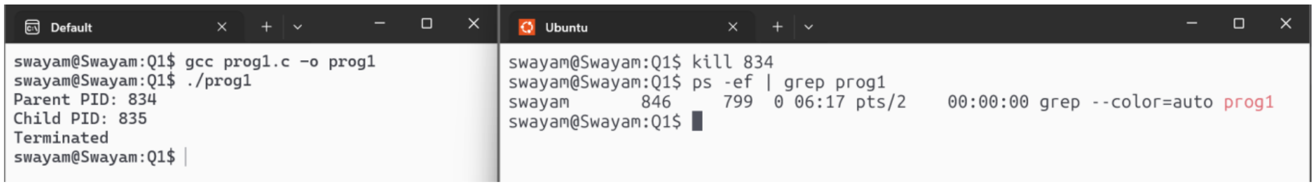
b) Terminate the child process. Then trace the state of processes.



```
swayam@Swayam:Q1$ gcc prog1.c -o prog1
swayam@Swayam:Q1$ ./prog1
Parent PID: 834
Child PID: 835
```

```
swayam@Swayam:Q1$ kill 835
swayam@Swayam:Q1$ ps -ef | grep prog1
swayam       834     365 99 06:13 pts/0    00:02:08 ./prog1
swayam       835     834 91 06:13 pts/0    00:01:57 [prog1] <defunct>
swayam       840     799  0 06:16 pts/2    00:00:00 grep --color=auto prog1
swayam@Swayam:Q1$
```

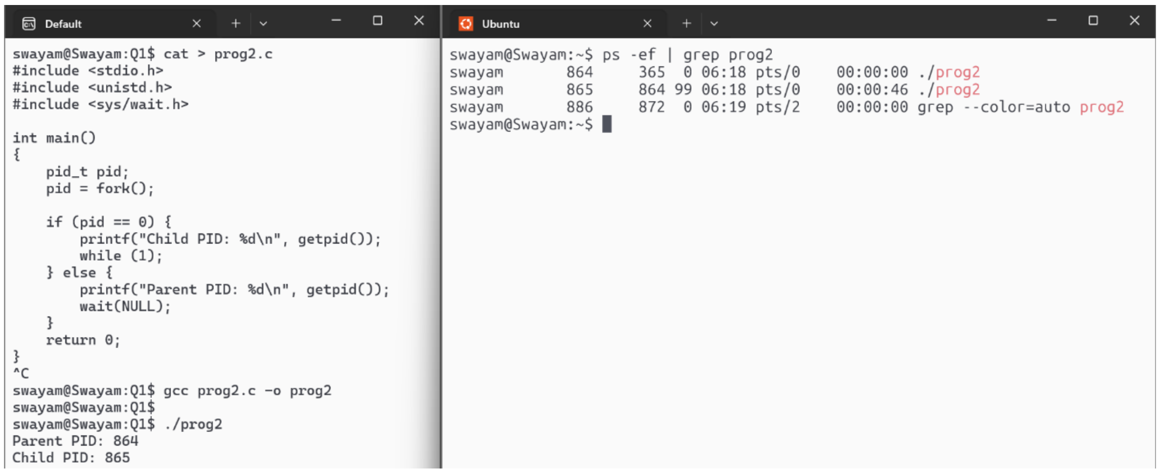
c) Run the program and trace the state of both processes. Terminate the parent process. Then trace the state of processes.



```
swayam@Swayam:Q1$ gcc prog1.c -o prog1
swayam@Swayam:Q1$ ./prog1
Parent PID: 834
Child PID: 835
Terminated
swayam@Swayam:Q1$
```

```
swayam@Swayam:Q1$ kill 834
swayam@Swayam:Q1$ ps -ef | grep prog1
swayam       846     799  0 06:17 pts/2    00:00:00 grep --color=auto prog1
swayam@Swayam:Q1$
```

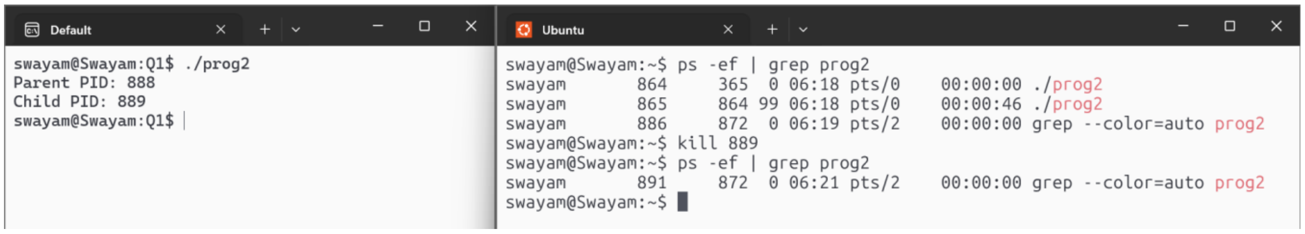
d) Modify the program so that the parent process after displaying the message will wait for child process to complete its task. Again run the program and trace the state of both processes.



```
swayam@Swayam:Q1$ cat > prog2.c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main()
{
    pid_t pid;
    pid = fork();

    if (pid == 0) {
        printf("Child PID: %d\n", getpid());
        while (1);
    } else {
        printf("Parent PID: %d\n", getpid());
        wait(NULL);
    }
    return 0;
}
^C
swayam@Swayam:Q1$ gcc prog2.c -o prog2
swayam@Swayam:Q1$
swayam@Swayam:Q1$ ./prog2
Parent PID: 864
Child PID: 865
```

```
swayam@Swayam:~$ ps -ef | grep prog2
swayam       864     365  0 06:18 pts/0    00:00:00 ./prog2
swayam       865     864 99 06:18 pts/0    00:00:46 ./prog2
swayam       886     872  0 06:19 pts/2    00:00:00 grep --color=auto prog2
swayam@Swayam:~$
```

e) Terminate the child process. Then trace the state of processes.

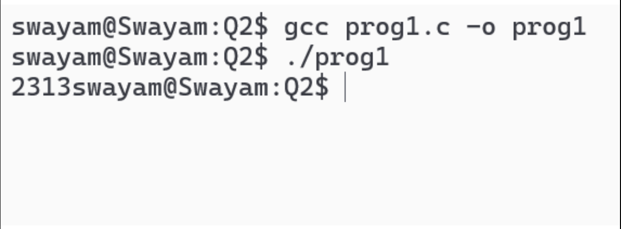


```
swayam@Swayam:Q1$ ./prog2
Parent PID: 888
Child PID: 889
swayam@Swayam:Q1$
```

```
swayam@Swayam:~$ ps -ef | grep prog2
swayam       864     365  0 06:18 pts/0    00:00:00 ./prog2
swayam       865     864 99 06:18 pts/0    00:00:46 ./prog2
swayam       886     872  0 06:19 pts/2    00:00:00 grep --color=auto prog2
swayam@Swayam:~$ kill 889
swayam@Swayam:~$ ps -ef | grep prog2
swayam       891     872  0 06:21 pts/2    00:00:00 grep --color=auto prog2
swayam@Swayam:~$
```

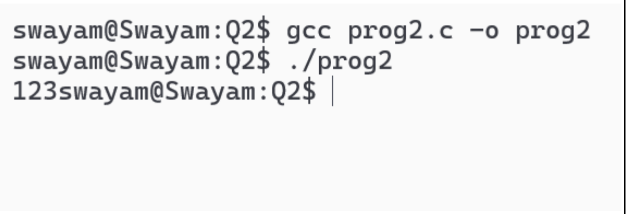
2. Trace the output of the following codes:

a)
```
int main( )
 {
    if(fork()==0)
       printf("1");
    else
       printf("2");
    printf("3");
return 0;
}
```



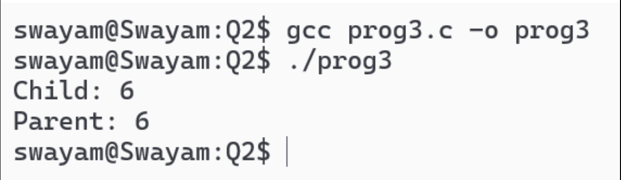
```
swayam@Swayam:Q2$ gcc prog1.c -o prog1
swayam@Swayam:Q2$ ./prog1
2313swayam@Swayam:Q2$
```

b)
```
int main( )
 {
    if(vfork()==0)
    {
          printf("1");
          _exit(0);

    }
    else
      printf("2");
      printf("3");
  }
```



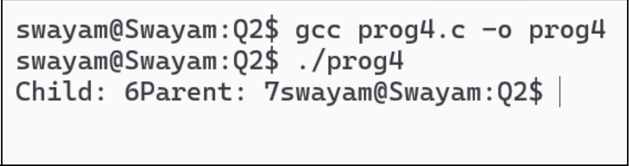
```
swayam@Swayam:Q2$ gcc prog2.c -o prog2
swayam@Swayam:Q2$ ./prog2
123swayam@Swayam:Q2$
```

c)
```
int main( )
{
  pid_t pid; int
 i=5;
 pid=fork();
 i=i+1; if(pid=
 =0)
 {
      printf("Child: %d",i);
  }
  else

  {
      wait(NULL);
      printf("Parent: %d",i);
   }
   return 0;
}
```



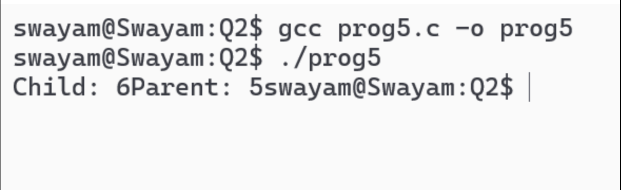
```
swayam@Swayam:Q2$ gcc prog3.c -o prog3
swayam@Swayam:Q2$ ./prog3
Child: 6
Parent: 6
swayam@Swayam:Q2$
```

d)
```
int main( )
{
  pid_t pid; int
  i=5;
  pid=vfork();
  i=i+1;
  if(pid==0)
 {
      printf("Child: %d",i);
      _exit(0);
 }

 else
 {
      printf("Parent: %d",i);
 }
 return 0;
}
```



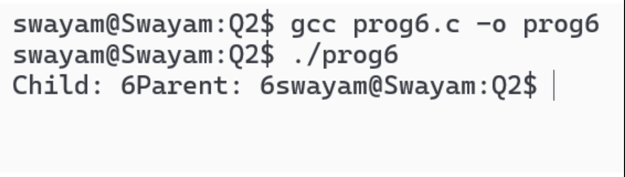
```
swayam@Swayam:Q2$ gcc prog4.c -o prog4
swayam@Swayam:Q2$ ./prog4
Child: 6Parent: 7swayam@Swayam:Q2$
```

e)
```
int main( )
{
 pid_t pid;
 int i=5;
 pid=fork();
 if(pid= =0)
 {
   i=i+1;
   printf("Child: %d",i);
 }
  else
  {
   wait(NULL);
   printf("Parent: %d",i);
  }
  return 0;
 }
```



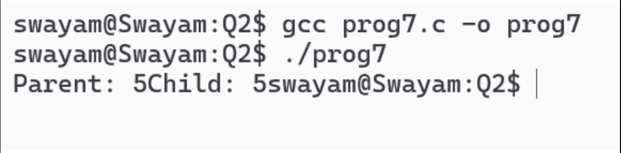
```
swayam@Swayam:Q2$ gcc prog5.c -o prog5
swayam@Swayam:Q2$ ./prog5
Child: 6Parent: 5swayam@Swayam:Q2$
```

f)
```
int main( )
{
  pid_t pid;
  int i=5;
   pid=vfork();
   if(pid==0)
   {
     i=i+1;
     printf("Child: %d",i);
     _exit(0);
   }
   else
   {
     printf("Parent: %d",i);
   }
   return 0;
  }
```



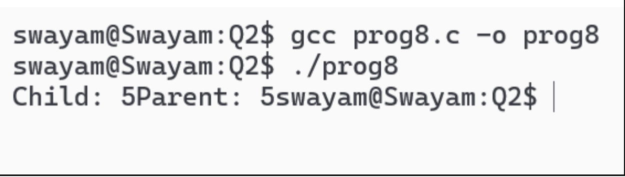
```
swayam@Swayam:Q2$ gcc prog6.c -o prog6
swayam@Swayam:Q2$ ./prog6
Child: 6Parent: 6swayam@Swayam:Q2$
```

g)
```
int main( )
{
 int i=5;
 if(fork( )==0)
 {
      printf("Child: %d",i);
 }
 else
 {
     printf("Parent: %d",i);
 }
 return 0;
}
```

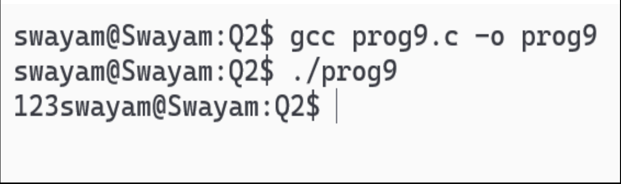


```
swayam@Swayam:Q2$ gcc prog7.c -o prog7
swayam@Swayam:Q2$ ./prog7
Parent: 5Child: 5swayam@Swayam:Q2$
```

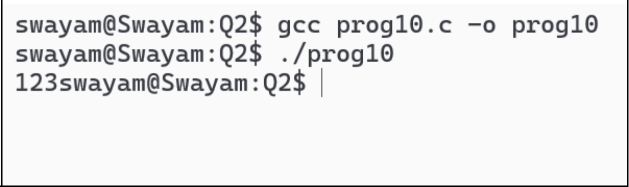
h)
```
int main( )
{
  int i=5;
  if(vfork( )==0)
  {
      printf("Child: %d",i);
      _exit(0);
   }
   else
   {
      printf("Parent: %d",i);
   }
    return 0;
 }
```



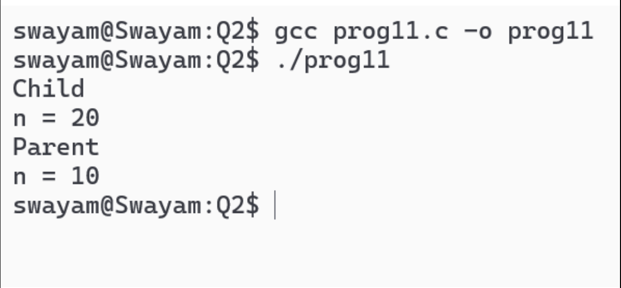
```
swayam@Swayam:Q2$ gcc prog8.c -o prog8
swayam@Swayam:Q2$ ./prog8
Child: 5Parent: 5swayam@Swayam:Q2$
```

i)
```
int main( )
{
 if(fork( )==0)
 {
     printf("1");
  }
  else
  {
      wait(NULL);
      printf("2");
      printf("3");
   }
   return 0;
  }
```


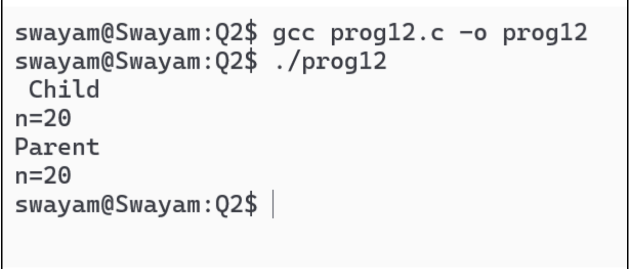
```
swayam@Swayam:Q2$ gcc prog9.c -o prog9
swayam@Swayam:Q2$ ./prog9
123swayam@Swayam:Q2$
```

j)
```
int main( )
{
 if(vfork( )==0)
 {
     printf("1");
     _exit(0);
 }
  else
 {
     printf("2");
    printf("3");
  }
  return 0;
}
```


```
swayam@Swayam:Q2$ gcc prog10.c -o prog10
swayam@Swayam:Q2$ ./prog10
123swayam@Swayam:Q2$
```

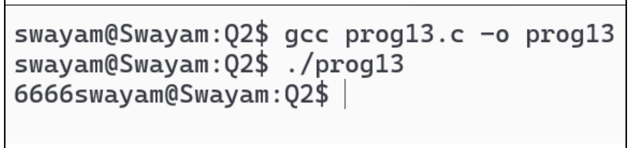
k)
```
int main( )
{
pid_t c1;
int n=10;
c1=fork( );
if(c1==0)
{
     printf(" Child\n");
     n=20;
     printf("n=%d \n",n);
 }
 else
{
      wait(NULL);
      printf("Parent\n");
      printf("n=%d \n",n);
  }
  return 0;
}
```


```
swayam@Swayam:Q2$ gcc prog11.c -o prog11
swayam@Swayam:Q2$ ./prog11
Child
n = 20
Parent
n = 10
swayam@Swayam:Q2$
```

l)
```
int main( )
{
 pid_t c1;
 int n=10;
 c1=vfork( );
 if(c1==0)
 {
      printf(" Child\n");
      n=20;
      printf("n=%d \n",n);
      _exit(0);
  }
  else
  {
       printf("Parent\n");
       printf("n=%d \n",n);
  }
  return 0;
 }
```


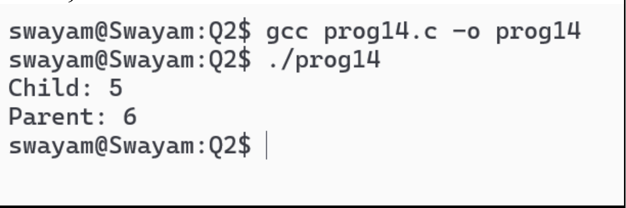
```
swayam@Swayam:Q2$ gcc prog12.c -o prog12
swayam@Swayam:Q2$ ./prog12
 Child
n=20
Parent
n=20
swayam@Swayam:Q2$
```

m)
```
int main( )
{
 int i=5;
 fork();
 i=i+1;
 fork();
 printf ( "% d",i);
 return 0;
}
```



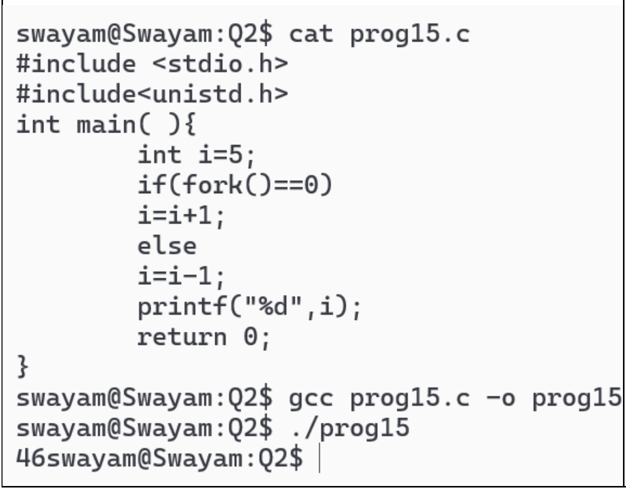
```
swayam@Swayam:Q2$ gcc prog13.c -o prog13
swayam@Swayam:Q2$ ./prog13
6666swayam@Swayam:Q2$
```

n)
```
int main( )
{
 pid_t pid;
 int i=5;
 pid=vfork();
 if(pid==0)
 {
      printf("Child: %d",i);
      _exit(0);
 }
  else
  {
         i=i+1;
         printf("Parent: %d",i);
  }
  return 0;
}
```



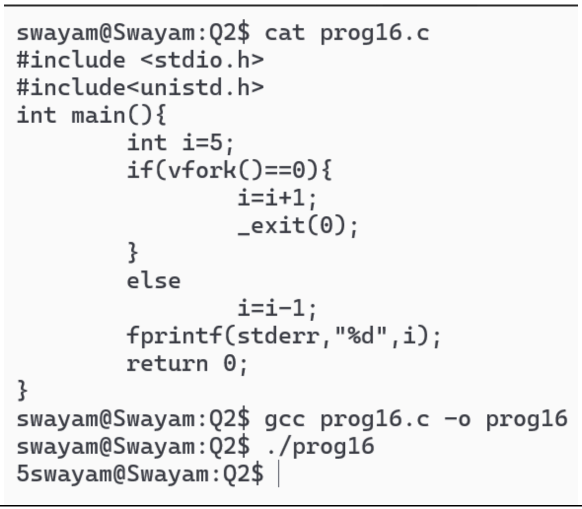
```
swayam@Swayam:Q2$ gcc prog14.c -o prog14
swayam@Swayam:Q2$ ./prog14
Child: 5
Parent: 6
swayam@Swayam:Q2$
```

o)
```
int main( )
{
 int i=5;
 if(fork()==0)
 i=i+1;
 else
 i=i-1;
 printf("%d",i);
 return 0;
}
```



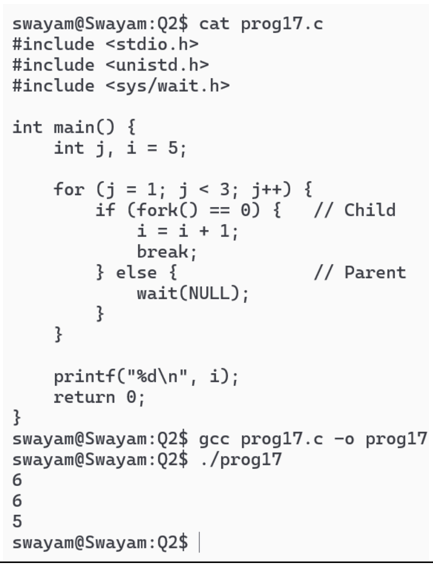
```
swayam@Swayam:Q2$ cat prog15.c
#include <stdio.h>
#include<unistd.h>
int main( ){
        int i=5;
        if(fork()==0)
        i=i+1;
        else
        i=i-1;
        printf("%d",i);
        return 0;
}
swayam@Swayam:Q2$ gcc prog15.c -o prog15
swayam@Swayam:Q2$ ./prog15
46swayam@Swayam:Q2$
```

p)
```
int main( )
{
 int i=5;
 if(vfork()==0)
 {
      i=i+1;
      _exit(0);
  }
  else
       i=i-1;
   fprintf(stderr,"%d",i);
   return 0;
   }
```



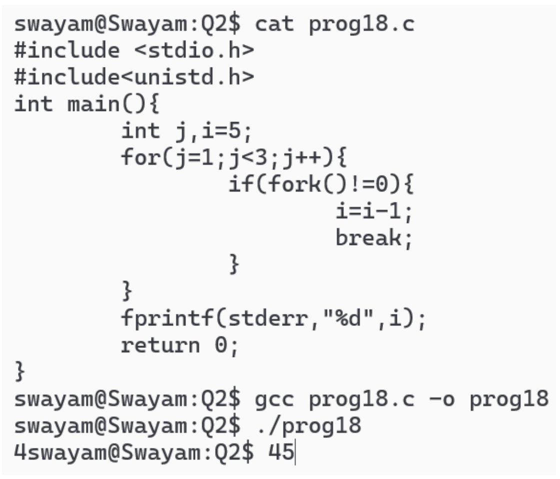
```
swayam@Swayam:Q2$ cat prog16.c
#include <stdio.h>
#include<unistd.h>
int main(){
        int i=5;
        if(vfork()==0){
                i=i+1;
                _exit(0);
        }
        else
                i=i-1;
        fprintf(stderr,"%d",i);
        return 0;
}
swayam@Swayam:Q2$ gcc prog16.c -o prog16
swayam@Swayam:Q2$ ./prog16
5swayam@Swayam:Q2$
```

q)
```
    int main( )
   {
   int j,i=5;
   for(j=1;j<3;j++)
   {
        if(fork()==0)
        {
              i=i+1;
              break;
         }
  else
     wait(NULL);
   }
   printf("%d",i);
   return 0;
   }
```



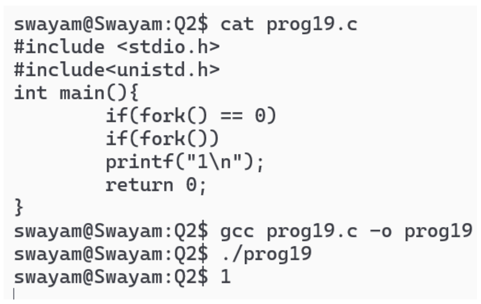
```
swayam@Swayam:Q2$ cat prog17.c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main() {
    int j, i = 5;

    for (j = 1; j < 3; j++) {
        if (fork() == 0) {   // Child
            i = i + 1;
            break;
        } else {             // Parent
            wait(NULL);
        }
    }

    printf("%d\n", i);
    return 0;
}
swayam@Swayam:Q2$ gcc prog17.c -o prog17
swayam@Swayam:Q2$ ./prog17
6
6
5
swayam@Swayam:Q2$
```

r)
```
       int main( )
      {
       int j,i=5;
       for(j=1;j<3;j++)
      {
           if(fork()!=0)
           {
               i=i-1;
               break;
            }

        }
        fprintf(stderr,"%d",i);
        return 0;
       }
```



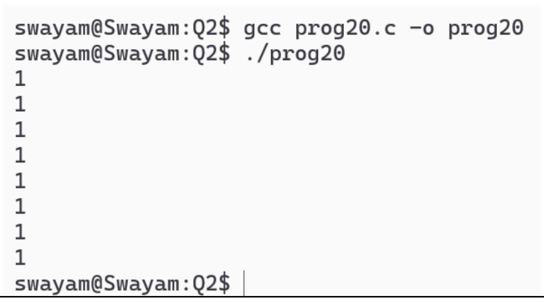
```
swayam@Swayam:Q2$ cat prog18.c
#include <stdio.h>
#include<unistd.h>
int main(){
        int j,i=5;
        for(j=1;j<3;j++){
                if(fork()!=0){
                        i=i-1;
                        break;
                }
        }
        fprintf(stderr,"%d",i);
        return 0;
}
swayam@Swayam:Q2$ gcc prog18.c -o prog18
swayam@Swayam:Q2$ ./prog18
4swayam@Swayam:Q2$ 45
```

s)
```
     int main( )
    {
     if(fork() == 0)
     if(fork())
     printf("1\n");

     return 0;
    }
```



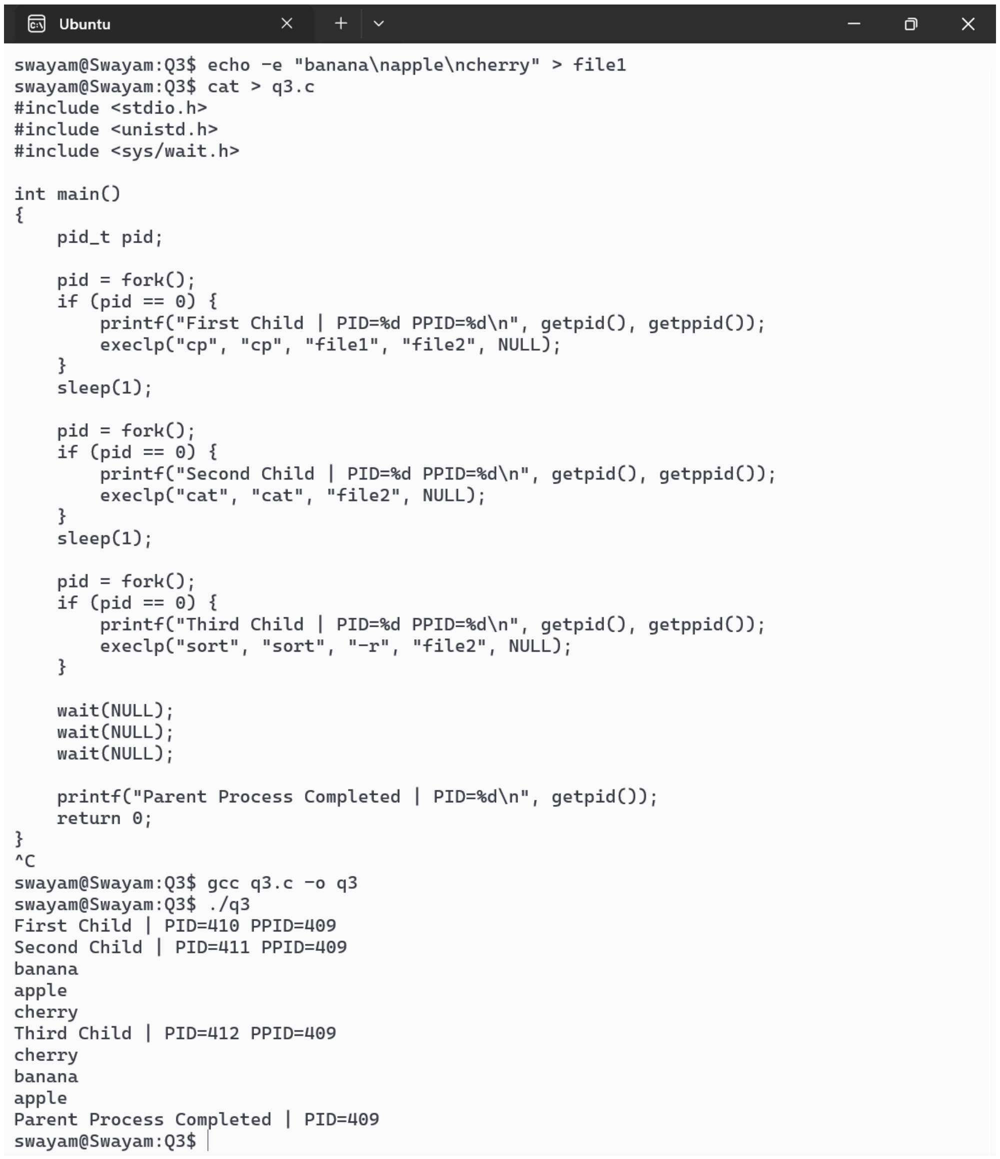
```
swayam@Swayam:Q2$ cat prog19.c
#include <stdio.h>
#include<unistd.h>
int main(){
        if(fork() == 0)
        if(fork())
        printf("1\n");
        return 0;
}
swayam@Swayam:Q2$ gcc prog19.c -o prog19
swayam@Swayam:Q2$ ./prog19
swayam@Swayam:Q2$ 1
```

t)
```
      void fun1(){
       fork();
       fork();
       printf("1\n");
       }
        int main() {
        fun1();
        printf("1\n");
        return 0;
        }
```



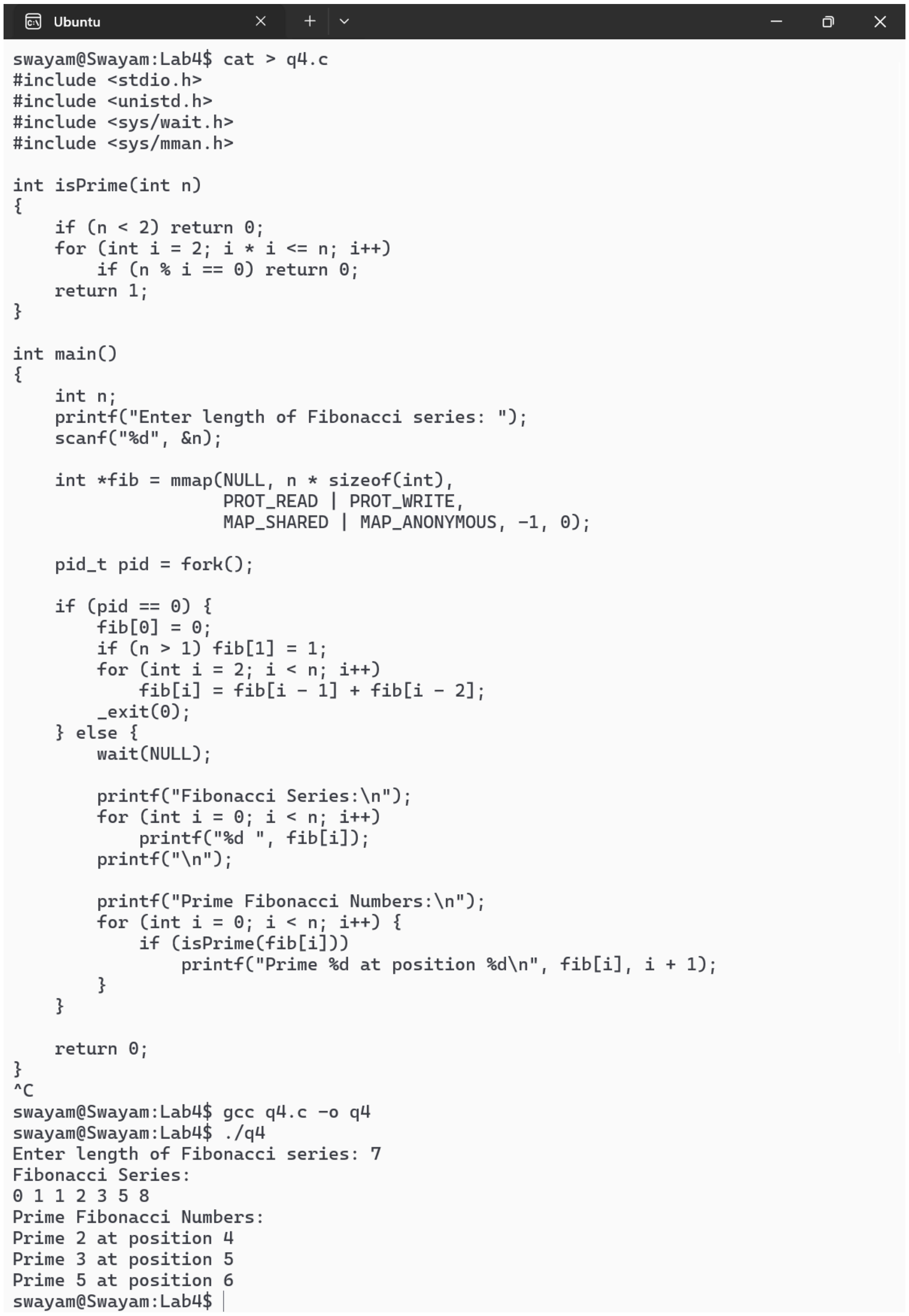
```
swayam@Swayam:Q2$ gcc prog20.c -o prog20
swayam@Swayam:Q2$ ./prog20
1
1
1
1
1
1
1
1
swayam@Swayam:Q2$
```

3. Write a C program that will create three child process to perform the following operations respectively:
   - First child will copy the content of file1 to file2
   - Second child will display the content of file2
   - Third child will display the sorted content of file2 in reverse order.
   - Each child process being created will display its id and its parent process id with appropriate message.
   - The parent process will be delayed for 1 second after creation of each child process. It will display appropriate message with its id after completion of all the child processes.



```
swayam@Swayam:Q3$ echo -e "banana\napple\ncherry" > file1
swayam@Swayam:Q3$ cat > q3.c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>

int main()
{
    pid_t pid;

    pid = fork();
    if (pid == 0) {
        printf("First Child | PID=%d PPID=%d\n", getpid(), getppid());
        execlp("cp", "cp", "file1", "file2", NULL);
    }
    sleep(1);

    pid = fork();
    if (pid == 0) {
        printf("Second Child | PID=%d PPID=%d\n", getpid(), getppid());
        execlp("cat", "cat", "file2", NULL);
    }
    sleep(1);

    pid = fork();
    if (pid == 0) {
        printf("Third Child | PID=%d PPID=%d\n", getpid(), getppid());
        execlp("sort", "sort", "-r", "file2", NULL);
    }

    wait(NULL);
    wait(NULL);
    wait(NULL);

    printf("Parent Process Completed | PID=%d\n", getpid());
    return 0;
}
^C
swayam@Swayam:Q3$ gcc q3.c -o q3
swayam@Swayam:Q3$ ./q3
First Child | PID=410 PPID=409
Second Child | PID=411 PPID=409
banana
apple
cherry
Third Child | PID=412 PPID=409
cherry
banana
apple
Parent Process Completed | PID=409
swayam@Swayam:Q3$
```

4. Write a C program that will create a child process to generate a Fibonacci series of specified length and store it in an array. The parent process will wait for the child to complete its task and then display the Fibonacci series and then display the prime Fibonacci number in the series along with its position with appropriate message.



```
swayam@Swayam:Lab4$ cat > q4.c
#include <stdio.h>
#include <unistd.h>
#include <sys/wait.h>
#include <sys/mman.h>

int isPrime(int n)
{
    if (n < 2) return 0;
    for (int i = 2; i * i <= n; i++)
        if (n % i == 0) return 0;
    return 1;
}

int main()
{
    int n;
    printf("Enter length of Fibonacci series: ");
    scanf("%d", &n);

    int *fib = mmap(NULL, n * sizeof(int),
                    PROT_READ | PROT_WRITE,
                    MAP_SHARED | MAP_ANONYMOUS, -1, 0);

    pid_t pid = fork();

    if (pid == 0) {
        fib[0] = 0;
        if (n > 1) fib[1] = 1;
        for (int i = 2; i < n; i++)
            fib[i] = fib[i - 1] + fib[i - 2];
        _exit(0);
    } else {
        wait(NULL);

        printf("Fibonacci Series:\n");
        for (int i = 0; i < n; i++)
            printf("%d ", fib[i]);
        printf("\n");

        printf("Prime Fibonacci Numbers:\n");
        for (int i = 0; i < n; i++) {
            if (isPrime(fib[i]))
                printf("Prime %d at position %d\n", fib[i], i + 1);
        }
    }

    return 0;
}
^C
swayam@Swayam:Lab4$ gcc q4.c -o q4
swayam@Swayam:Lab4$ ./q4
Enter length of Fibonacci series: 7
Fibonacci Series:
0 1 1 2 3 5 8
Prime Fibonacci Numbers:
Prime 2 at position 4
Prime 3 at position 5
Prime 5 at position 6
swayam@Swayam:Lab4$
```

Laboratory Assignments

Subject: Design Principles of Operating Systems

Subject code: CSE 3249

Assignment 5: Implementation of synchronization using semaphore:

Objective of this Assignment:

- To implement the concept of multi-threading in a process.
- To learn the use of semaphore i.e., to control access to shared resources.

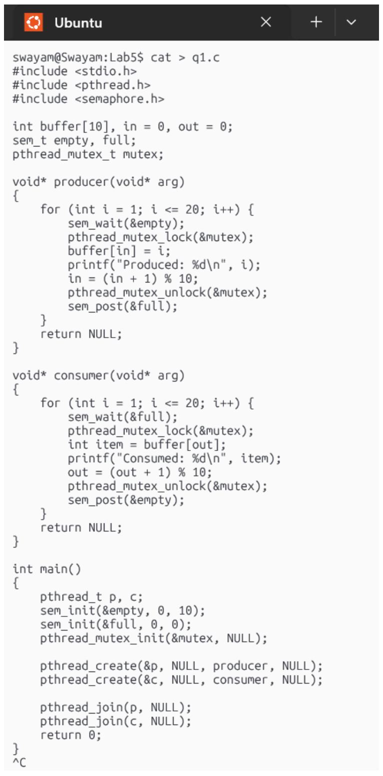
1. Producer-Consumer problem

Problem: Write a C program to implement the producer-consumer program where:

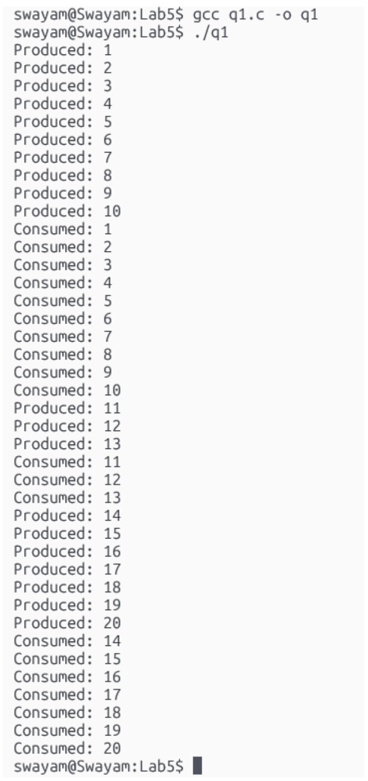
- Producer generates integers from 1 to 50.
- Consumer processes the numbers.

Requirements:

- Use a shared buffer with a maximum size of 10.
- Use semaphores and mutex to ensure thread-safe access to the buffer.
- Print the number that producer is producing and consumer is consuming.
- Both producer and consumer will continue for 20 iterations



```
swayam@Swayam:Lab5$ cat > q1.c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>

int buffer[10], in = 0, out = 0;
sem_t empty, full;
pthread_mutex_t mutex;

void* producer(void* arg)
{
    for (int i = 1; i <= 20; i++) {
        sem_wait(&empty);
        pthread_mutex_lock(&mutex);
        buffer[in] = i;
        printf("Produced: %d\n", i);
        in = (in + 1) % 10;
        pthread_mutex_unlock(&mutex);
        sem_post(&full);
    }
    return NULL;
}

void* consumer(void* arg)
{
    for (int i = 1; i <= 20; i++) {
        sem_wait(&full);
        pthread_mutex_lock(&mutex);
        int item = buffer[out];
        printf("Consumed: %d\n", item);
        out = (out + 1) % 10;
        pthread_mutex_unlock(&mutex);
        sem_post(&empty);
    }
    return NULL;
}

int main()
{
    pthread_t p, c;
    sem_init(&empty, 0, 10);
    sem_init(&full, 0, 0);
    pthread_mutex_init(&mutex, NULL);

    pthread_create(&p, NULL, producer, NULL);
    pthread_create(&c, NULL, consumer, NULL);

    pthread_join(p, NULL);
    pthread_join(c, NULL);
    return 0;
}
^C
```



```
swayam@Swayam:Lab5$ gcc q1.c -o q1
swayam@Swayam:Lab5$ ./q1
Produced: 1
Produced: 2
Produced: 3
Produced: 4
Produced: 5
Produced: 6
Produced: 7
Produced: 8
Produced: 9
Produced: 10
Consumed: 1
Consumed: 2
Consumed: 3
Consumed: 4
Consumed: 5
Consumed: 6
Consumed: 7
Consumed: 8
Consumed: 9
Consumed: 10
Produced: 11
Produced: 12
Produced: 13
Consumed: 11
Consumed: 12
Consumed: 13
Produced: 14
Produced: 15
Produced: 16
Produced: 17
Produced: 18
Produced: 19
Produced: 20
Consumed: 14
Consumed: 15
Consumed: 16
Consumed: 17
Consumed: 18
Consumed: 19
Consumed: 20
swayam@Swayam:Lab5$
```

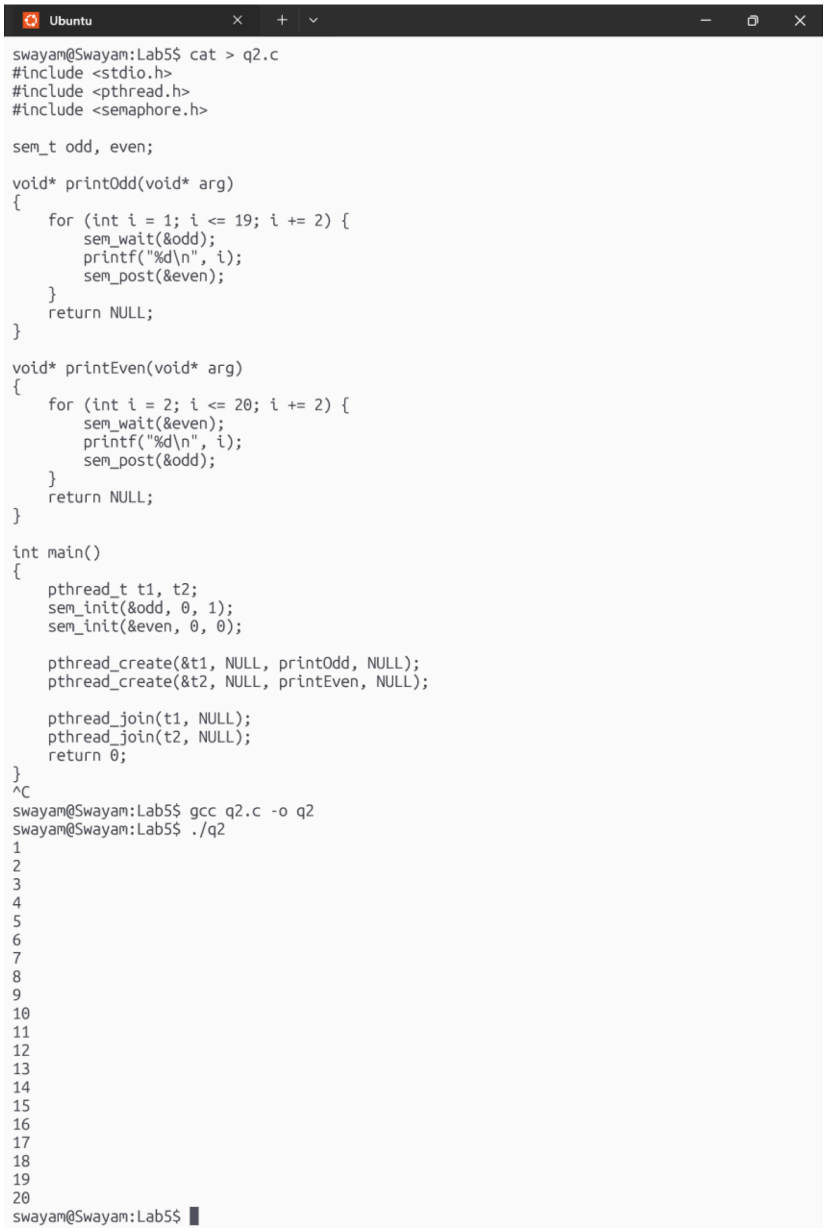
2. Alternating Numbers with Two Threads

Problem: Write a program to print 1, 2, 3 … upto 20. Create threads where two threads print numbers alternately.

- **Thread A** prints odd numbers: 1, 3, 5 ...
- **Thread B** prints even numbers: 2, 4, 6 ...

Requirements:

- Use semaphores to control the order of execution of the threads.
- Ensure no race conditions occur.



```
swayam@Swayam:Lab5$ cat > q2.c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>

sem_t odd, even;

void* printOdd(void* arg)
{
    for (int i = 1; i <= 19; i += 2) {
        sem_wait(&odd);
        printf("%d\n", i);
        sem_post(&even);
    }
    return NULL;
}

void* printEven(void* arg)
{
    for (int i = 2; i <= 20; i += 2) {
        sem_wait(&even);
        printf("%d\n", i);
        sem_post(&odd);
    }
    return NULL;
}

int main()
{
    pthread_t t1, t2;
    sem_init(&odd, 0, 1);
    sem_init(&even, 0, 0);

    pthread_create(&t1, NULL, printOdd, NULL);
    pthread_create(&t2, NULL, printEven, NULL);

    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    return 0;
}
^C
swayam@Swayam:Lab5$ gcc q2.c -o q2
swayam@Swayam:Lab5$ ./q2
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
swayam@Swayam:Lab5$
```

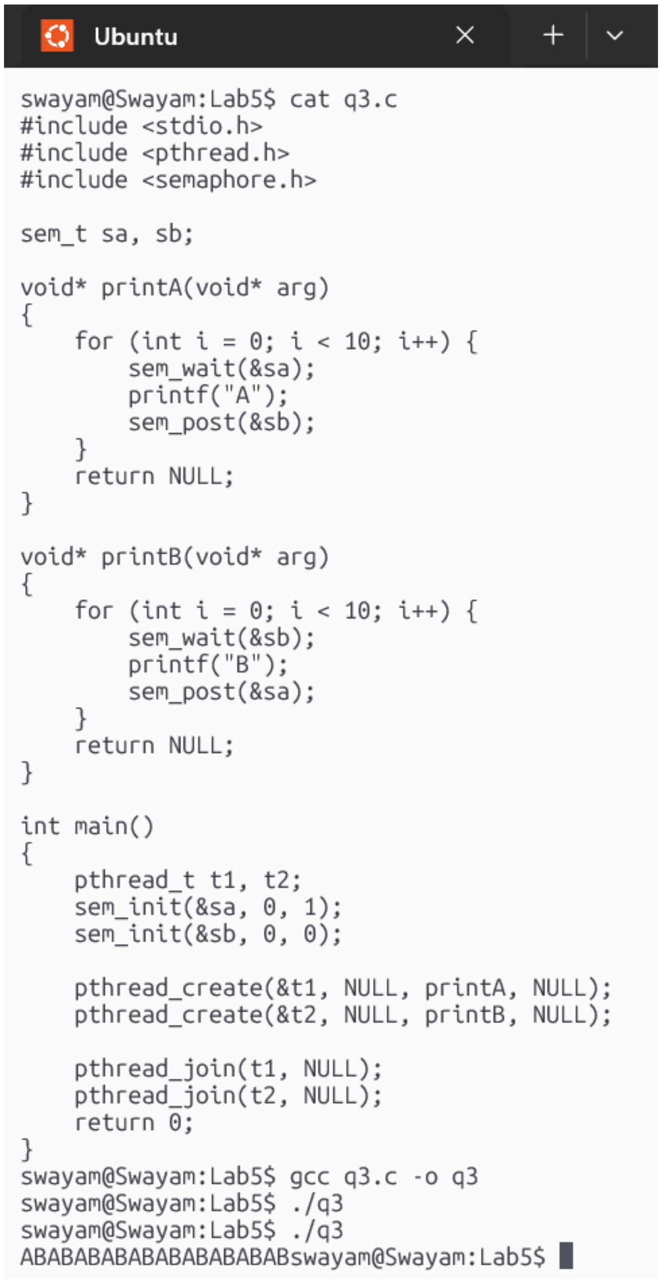
3. Alternating Characters

Problem: Write a program to create two threads that print characters (A and B) alternately such as ABABABABA…. upto 20. Use semaphores to synchronize the threads.

- **Thread A** prints A.
- **Thread B** prints B.

Requirements:

- Use semaphores to control the order of execution of the threads.
- Ensure no race conditions occur.



```
swayam@Swayam:Lab5$ cat q3.c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>

sem_t sa, sb;

void* printA(void* arg)
{
    for (int i = 0; i < 10; i++) {
        sem_wait(&sa);
        printf("A");
        sem_post(&sb);
    }
    return NULL;
}

void* printB(void* arg)
{
    for (int i = 0; i < 10; i++) {
        sem_wait(&sb);
        printf("B");
        sem_post(&sa);
    }
    return NULL;
}

int main()
{
    pthread_t t1, t2;
    sem_init(&sa, 0, 1);
    sem_init(&sb, 0, 0);

    pthread_create(&t1, NULL, printA, NULL);
    pthread_create(&t2, NULL, printB, NULL);

    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    return 0;
}
swayam@Swayam:Lab5$ gcc q3.c -o q3
swayam@Swayam:Lab5$ ./q3
swayam@Swayam:Lab5$ ./q3
ABABABABABABABABABABswayam@Swayam:Lab5$
```

4. Countdown and Countup

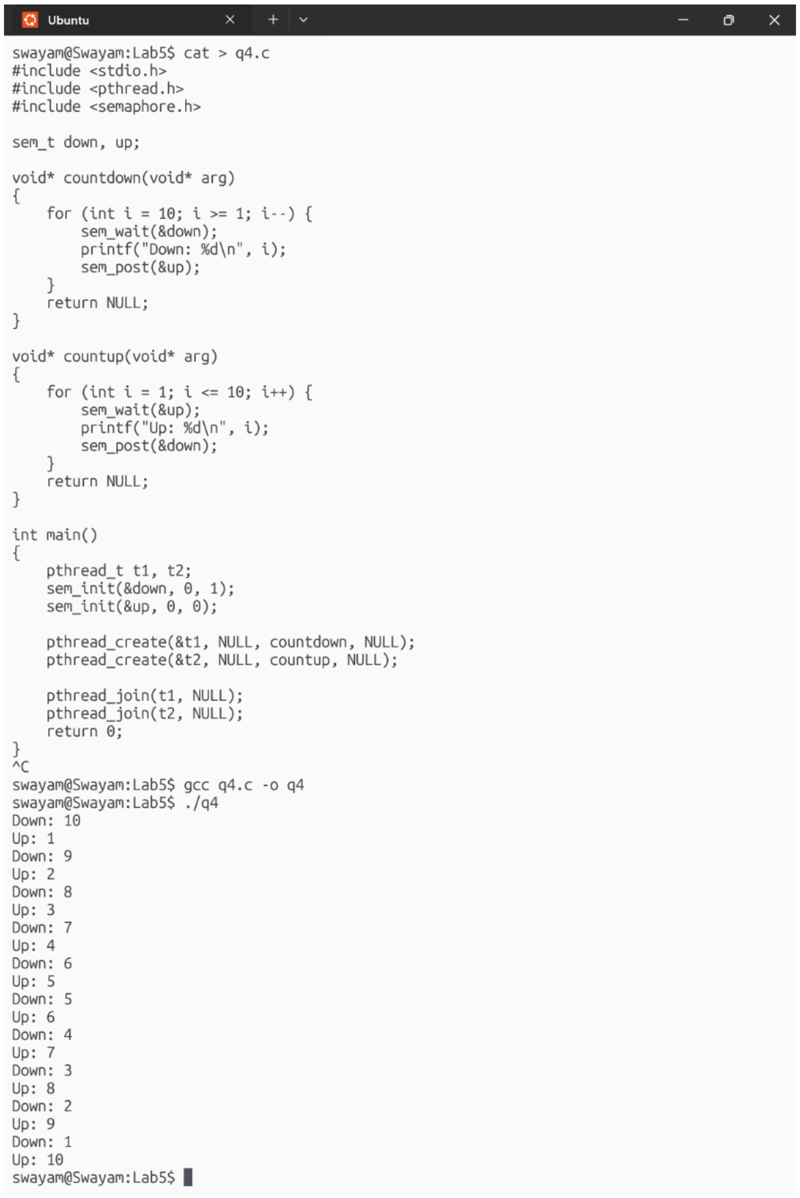
Problem: Write a program create two threads where:

- **Thread A** counts down from 10 to 1.
- **Thread B** counts up from 1 to 10.

Both threads should alternate execution.

Requirements:

- Use semaphores to control the order of execution of the threads.
- Ensure no race conditions occur.



```
swayam@Swayam:Lab5$ cat > q4.c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>

sem_t down, up;

void* countdown(void* arg)
{
    for (int i = 10; i >= 1; i--) {
        sem_wait(&down);
        printf("Down: %d\n", i);
        sem_post(&up);
    }
    return NULL;
}

void* countup(void* arg)
{
    for (int i = 1; i <= 10; i++) {
        sem_wait(&up);
        printf("Up: %d\n", i);
        sem_post(&down);
    }
    return NULL;
}

int main()
{
    pthread_t t1, t2;
    sem_init(&down, 0, 1);
    sem_init(&up, 0, 0);

    pthread_create(&t1, NULL, countdown, NULL);
    pthread_create(&t2, NULL, countup, NULL);

    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    return 0;
}
^C
swayam@Swayam:Lab5$ gcc q4.c -o q4
swayam@Swayam:Lab5$ ./q4
Down: 10
Up: 1
Down: 9
Up: 2
Down: 8
Up: 3
Down: 7
Up: 4
Down: 6
Up: 5
Down: 5
Up: 6
Down: 4
Up: 7
Down: 3
Up: 8
Down: 2
Up: 9
Down: 1
Up: 10
swayam@Swayam:Lab5$ 
```

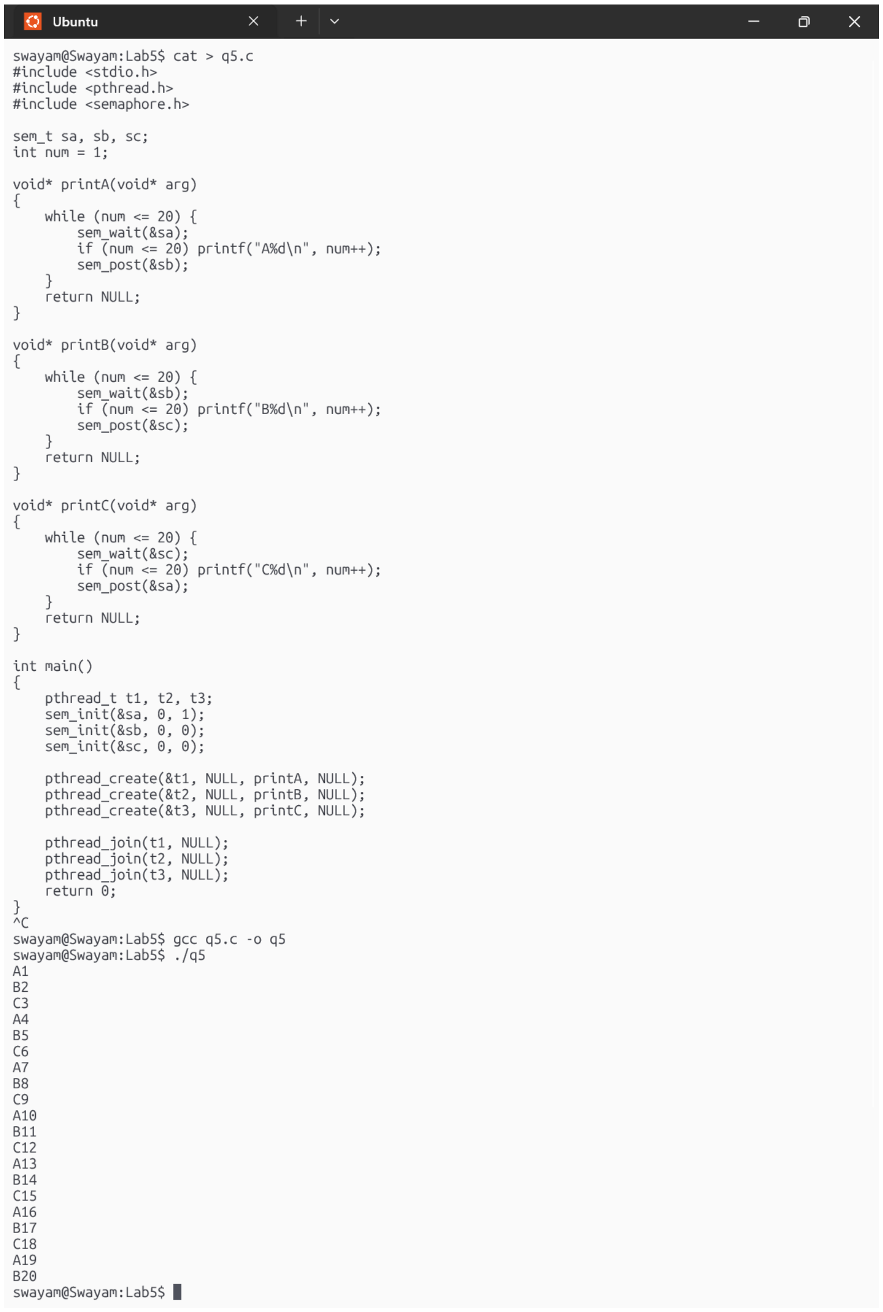
5. Sequence Printing using Threads

Problem: Write a program that creates three threads: Thread A, Thread B, and Thread C. The threads must print numbers in the following sequence: A1, B2, C3, A4, B5, C6 … upto 20 numbers.

- **Thread A** prints A1, A4, A7, …
- **Thread B** prints B2, B5, B8, …
- **Thread C** prints C3, C6, C9, ...

Requirements:

- Use semaphores to control the order of execution of the threads.
- Ensure no race conditions occur



```
swayam@Swayam:Lab5$ cat > q5.c
#include <stdio.h>
#include <pthread.h>
#include <semaphore.h>

sem_t sa, sb, sc;
int num = 1;

void* printA(void* arg)
{
    while (num <= 20) {
        sem_wait(&sa);
        if (num <= 20) printf("A%d\n", num++);
        sem_post(&sb);
    }
    return NULL;
}

void* printB(void* arg)
{
    while (num <= 20) {
        sem_wait(&sb);
        if (num <= 20) printf("B%d\n", num++);
        sem_post(&sc);
    }
    return NULL;
}

void* printC(void* arg)
{
    while (num <= 20) {
        sem_wait(&sc);
        if (num <= 20) printf("C%d\n", num++);
        sem_post(&sa);
    }
    return NULL;
}

int main()
{
    pthread_t t1, t2, t3;
    sem_init(&sa, 0, 1);
    sem_init(&sb, 0, 0);
    sem_init(&sc, 0, 0);

    pthread_create(&t1, NULL, printA, NULL);
    pthread_create(&t2, NULL, printB, NULL);
    pthread_create(&t3, NULL, printC, NULL);

    pthread_join(t1, NULL);
    pthread_join(t2, NULL);
    pthread_join(t3, NULL);
    return 0;
}
^C
swayam@Swayam:Lab5$ gcc q5.c -o q5
swayam@Swayam:Lab5$ ./q5
A1
B2
C3
A4
B5
C6
A7
B8
C9
A10
B11
C12
A13
B14
C15
A16
B17
C18
A19
B20
swayam@Swayam:Lab5$
```