IMPLEMENTATION

```
/*
Roll No         : B21CSB69
Name            : Sreelal V
Experiment No : 2.1
*/

#include<stdio.h>
#include<sys/wait.h>
#include<stdbool.h>
#include<unistd.h>

void fibonacci(int n ){
        printf("Fibnocci number \n") ;
        int i = 0 ,j = 1  , count = 0;
        if( n == 1)
                printf("%d ", i) ;
        else if ( n == 2 )
                printf("%d %d ",i, j ) ;
        else{
                int k = i + j ;
                printf("%d %d ",i, j) ;
                for ( int x = 3 ; x <= n ; x++ ){
                        printf("%d ",k) ;
                        i = j ;
                        j = k ;
                        k = i+j ;
                }
        }
}

bool isPrime(int n){
        if ( n == 0 || n== 1 )
                return false ;
        else
                for ( int i= 2 ; i<= n/2 ; i++ )
                        if ( n % i == 0 )
                                return false ;
        return true ;
}

void prime(int n){
        printf("Prime Numbers\n") ;
        int count = 0 ;
        for ( int i = 0 ; count < n ; i++)
                if ( isPrime(i) ){
                        printf("%d ",i) ;
                        count++ ;
                }
}
```

```
void forks(int n){
        int p = fork() ;
        if ( p == -1 )
                printf("Error in creating process\n") ;
        else if ( p== 0 ){
                fibonacci(n) ;
        }else if ( p > 0){
                wait(NULL) ;
                prime(n) ;
        }
}

int main(){
        int n ;
        printf("Enter a number : " ) ;
        scanf("%d", &n ) ;
        forks(n) ;
        printf("\n") ;
        return 0 ;
}
```

output:

```
Enter a number : 10
Fibnocci number
0 1 1 2 3 5 8 13 21 34
Prime Numbers
2 3 5 7 11 13 17 19 23 29
```

IMPLEMENTATION

```
/*
Roll No         : B21CSB69
Name            : Sreelal V
Experiment No : 2.2
*/

#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
#include<stdlib.h>

void printProcess(char process, int pid,int ppid){
        printf("\nPROCESS : %c \n",process) ;
        printf("Process ID: %d\n",pid);
        printf("Parent Process ID: %d\n",ppid);

}
void printError(){
        printf("Error in process creation\n") ;
}
int main(){

        int a = fork() ; //PROCESS A
        wait(NULL) ;
        if ( a == 0 ){
                printProcess('A',getpid(),getppid());

                int b = fork() ;//PROCESS B
                wait(NULL) ;
                if( b == 0){
                        printProcess('B',getpid(),getppid());

                        int d = fork() ; //PROCESS D
                        wait(NULL) ;
                        if ( d == 0 ){
                                printProcess('D',getpid(),getppid());

                                int h = fork() ; //PROCESS H
                                wait(NULL) ;
                                if ( h == 0 ){
                                        printProcess('H',getpid(),getppid());

                                        int i = fork() ; //PROCESS I
                                        wait(NULL) ;
                                        if ( i == 0 ){
                                                printProcess('I',getpid(),getppid());
                                        }else if ( i > 0 ){
                                                exit(0) ;
                                        }else
                                                printError() ; //END PROCESS I
```

```
                }else if ( h > 0 ){
                        exit(0) ;
                }else
                        printError() ; //END PROCESS H

        }else if ( d > 0 ){
                int e = fork() ; //PROCESS E
                wait(NULL) ;
                if ( e == 0 ){
                        printProcess('E',getpid(),getppid());
                }else if ( e > 0 ){
                        int f = fork() ; //PROCESS F
                        wait(NULL) ;
                        if ( f == 0 ){
                                printProcess('F',getpid(),getppid());
                        }else if ( f > 0 ){
                                exit(0) ;
                        }else
                                printError() ; //END PROCESS F
                }else
                        printError() ; //END PROCESS E


        }else
                printError() ; //END PROCESS D

    }else if (b > 0 ){
            int c = fork() ; //PROCESS C
            wait(NULL) ;
            if ( c == 0 ){
                    printProcess('C',getpid(),getppid());

                    int g = fork() ; //PROCESS G
                    wait(NULL) ;
                    if ( g == 0 ){
                            printProcess('G',getpid(),getppid());
                    }else if ( g > 0 ){
                            exit(0) ;
                    }else
                            printError() ; //END PROCESS G
            }else if ( c > 0 ){
                    exit(0) ;
            }else
                    printError() ; //END PROCESS C
    }else
            printError() ; //END PROCESS B


}else if ( a > 0 ){
    exit(0) ;
}else
```

```
                printError() ; //END PROCESS A
        exit(0) ;
        return 0 ;
}
```

output:

```
PROCESS : A
Process ID: 7233
Parent Process ID: 7232

PROCESS : B
Process ID: 7234
Parent Process ID: 7233

PROCESS : D
Process ID: 7235
Parent Process ID: 7234

PROCESS : H
Process ID: 7236
Parent Process ID: 7235

PROCESS : I
Process ID: 7237
Parent Process ID: 7236

PROCESS : E
Process ID: 7238
Parent Process ID: 7234

PROCESS : F
Process ID: 7239
Parent Process ID: 7234

PROCESS : C
Process ID: 7240
Parent Process ID: 7233

PROCESS : G
Process ID: 7241
Parent Process ID: 7240
```

IMPLEMENTATION

```
/*
Roll No         : B21CSB69
Name            : Sreelal V
Experiment No : 2.3
*/

#include<stdio.h>
#include<unistd.h>
#include<sys/wait.h>
#include<stdlib.h>

void  printError(){
        printf("Error in process creation\n") ;
}

void main(){
        int f1 = fork() ;
        if ( f1 == -1 )
                printError() ;
        else if ( f1 == 0 ){
                sleep(2) ;
                printf("First child : %d \n",getpid() )  ;
        }else {
                int f2 = fork() ;
                if ( f2 == -1 )
                        printError() ;
                else if ( f2 == 0 ){
                        sleep(1) ;
                        printf("Second child : %d \n",getpid()) ;
                }else{
                        int f3 = fork() ;
                        if ( f3 == -1 )
                                printError() ;
                        else if ( f3 == 0 )
                                printf("Third child : %d \n",getpid()) ;
                        else{
                                sleep(3) ;
                                printf("Parent Process : %d \n",getpid()) ;
                        }
                }
        }
}
```

output:

```
Third child : 7307
Second child : 7306
First child : 7305
Parent Process : 7304
```