Project 1

1.1.1 parent process P creates a single child process that generates a sequence of Fibonacci numbers

1.1.2 parent process P creates two child processes C1 & C2 where each process is connected to the other processes by pipes; i.e., P is connected to C1 by pipe fd0, C1 is connected to C2 by pipe fd1, and C2 is connected to P by fd2. Hints:

```c
#include <stdio.h>
#include <unistd.h>

int main(){
    pid_t ppid, pid;
    ppid = getpids();
    pid_t pid[2];
    pid[0] = ppid;
    while(pid[0] == ppid){
        pid[1] = fork();
        pid[2] = fork();
    }
    if(pid == 0){
        pid = getpid();
        if(pid==pid[1]){ //compute fib(n)}
        if(pid==pid[2]){ // compute ratios}
    }
    if(pid == 0){
        pid = getpid();
        if(pid==pid[1]){ //compute fib(n)}
        if(pid==pid[2]){ // compute ratios}
    }
    return 0;
}
```

Parent

```c
struct node
{int fib_seed;
 float ratio;
}result[20];

populate result[i].fib_seed with initial values ranging from 5 to 45
populate result[i].ratio with zeros

send result to C1 via pipe fd0
write(fd0[WRITE_END], result, sizeof(result));
close pipe fd0
wait();
receive result from C2 via pipe fd2
read(fd2[READ_END], result, sizeof(result));
analyze the content of the result data structure
print result & analysis
```

C1

```c
struct node
{int fib_seed;
 int fib_seq[50];
}transit[20];

receive result from Parent via pipe fd0
compute sequence fib_seq[i] using (result[i].fib_seed)
store fib_seq[i] and result[i].fib_seed in the data structure transit
send transit to C2 via pipe fd1
print result[i].fib_seed and sequence fib(result[i].fib_seed)
close pipes fd0 & fd1
```

C2

```c
struct node
{int fib_seed;
 int fib_seq[50];
}transit[20];

receive data structure transit from C1 via fd1
compute ratio and populate result data structure with fib_seed’s and ratio’s
send result data structure to Parent via fd2
close pipes fd1 & fd2
terminate
```
Rough Algorithm

P provides C1 with a number, e.g., n; C1 computes the sequence of \( \text{fib}(n) \) numbers; C1 sends the number n and the sequence of fib numbers to C2.

C2 computes the ratio of #prime numbers/#non-prime numbers in the sequence; C2 sends the number n and the ratio to P.

P accumulates the number n and the associated ratio in a data structure array; reports the results.

accumulate 20 sequences length varying between 5 and 45 numbers
repeat this process 20 times, i.e., goto

1.A.3
same as 1.A.2 except using shared memory

1.B.1 same as 1.A.1 except using threads

1.B.2
create two new threads t1 & t2
main thread provides t1 with number – same as Parent in 1.A.2
t1 computes \( \text{fib}(n) \) sequence; stores result in the common memory
t2 computes ratio; stores result in the common memory
main thread accumulates array of results