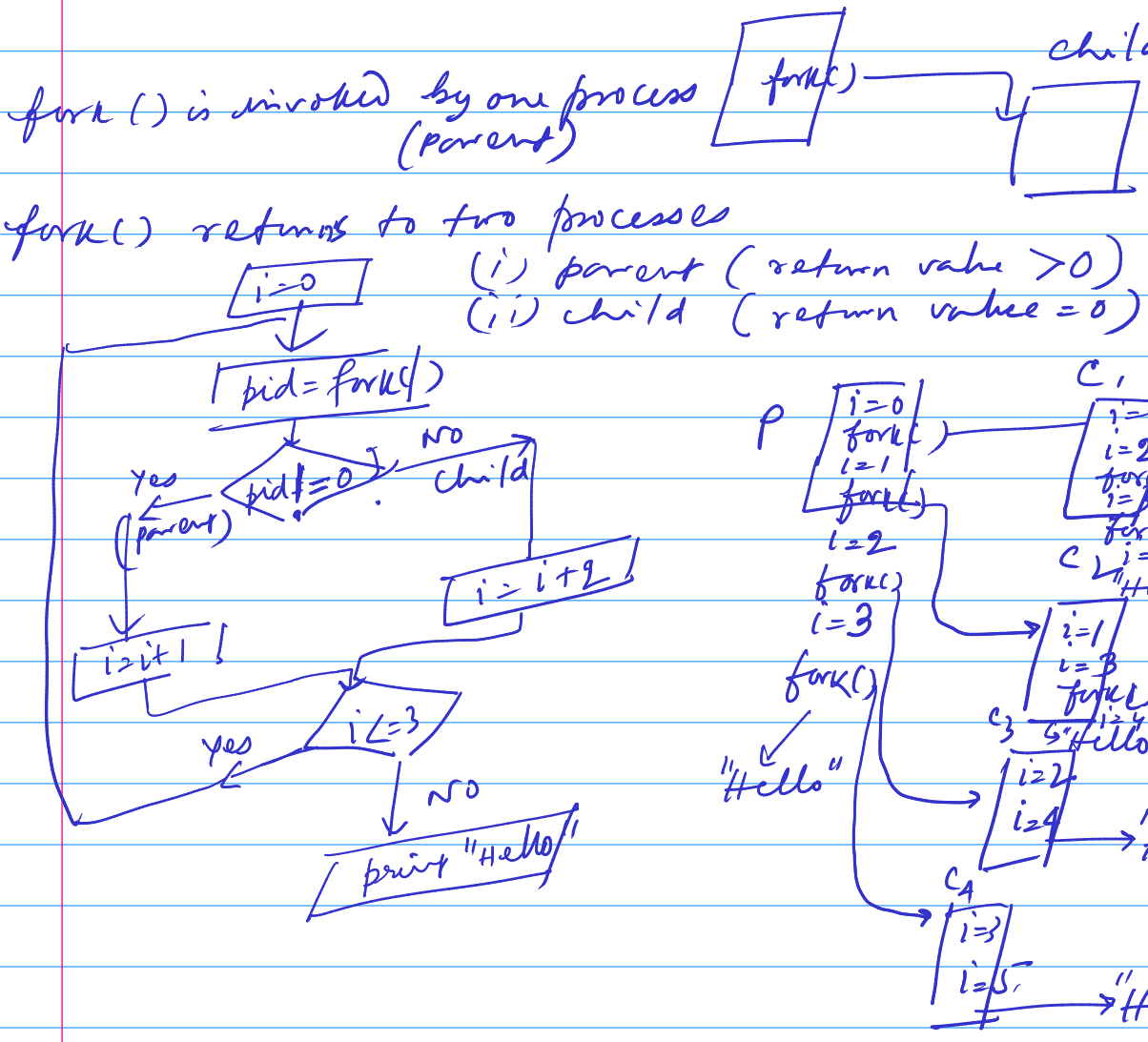
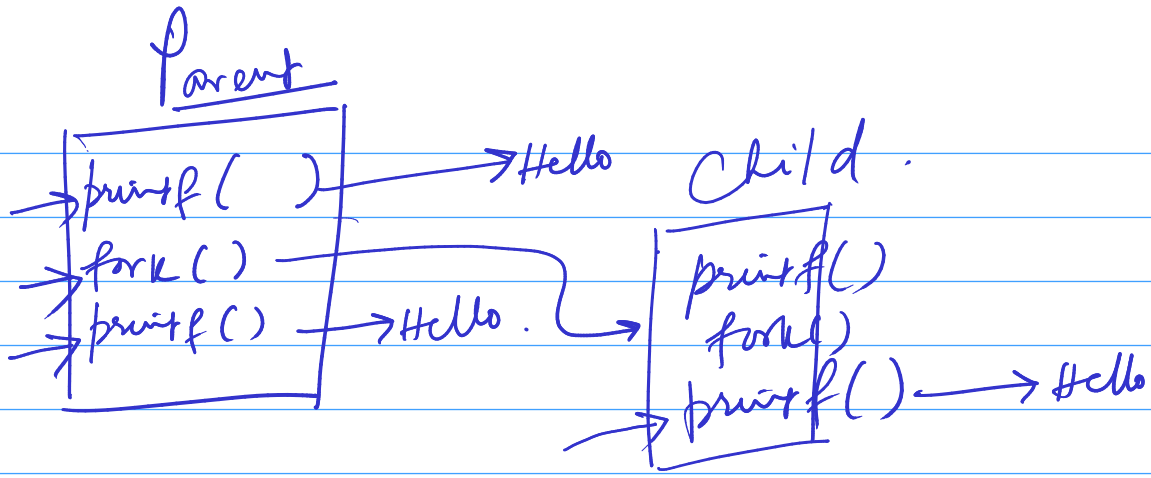
For the following code segment determine how many times "Hello" will be printed. Provide justifications for your answer.

```
int i=0;
int pid;
do{
        pid=fork();
        if(pid!=0);
                i++;
        else
                i+=2;
}while(i<=3);
printf("Hello");
```

After fork()

Child starts from the next instruction after that fork() for which it was created. Child will have same memory image as that of parent.



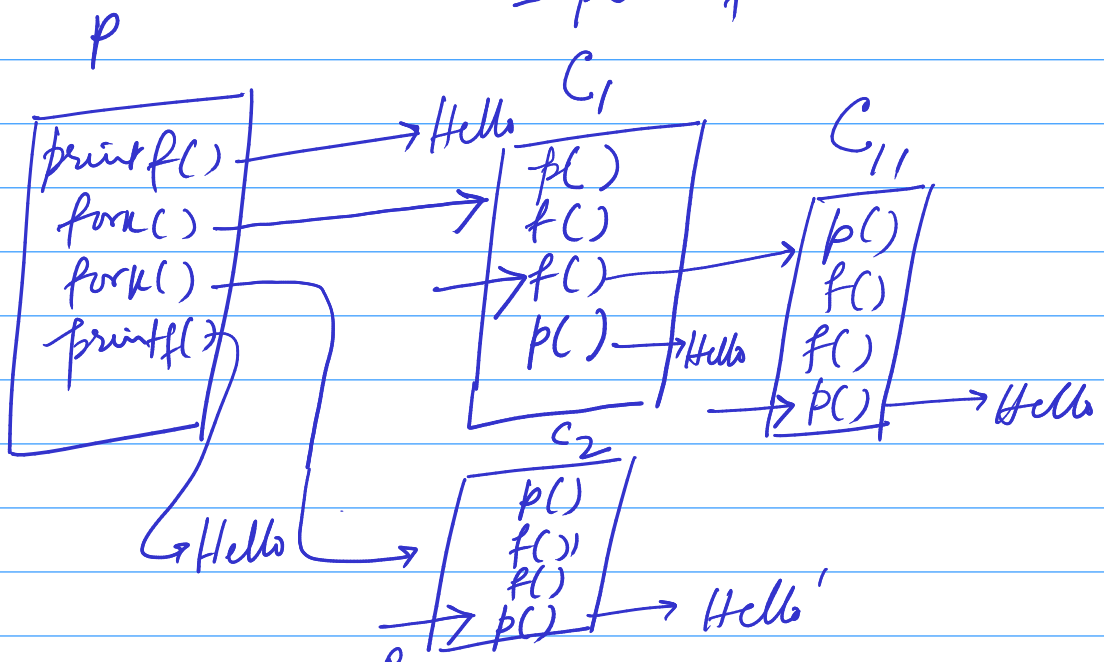


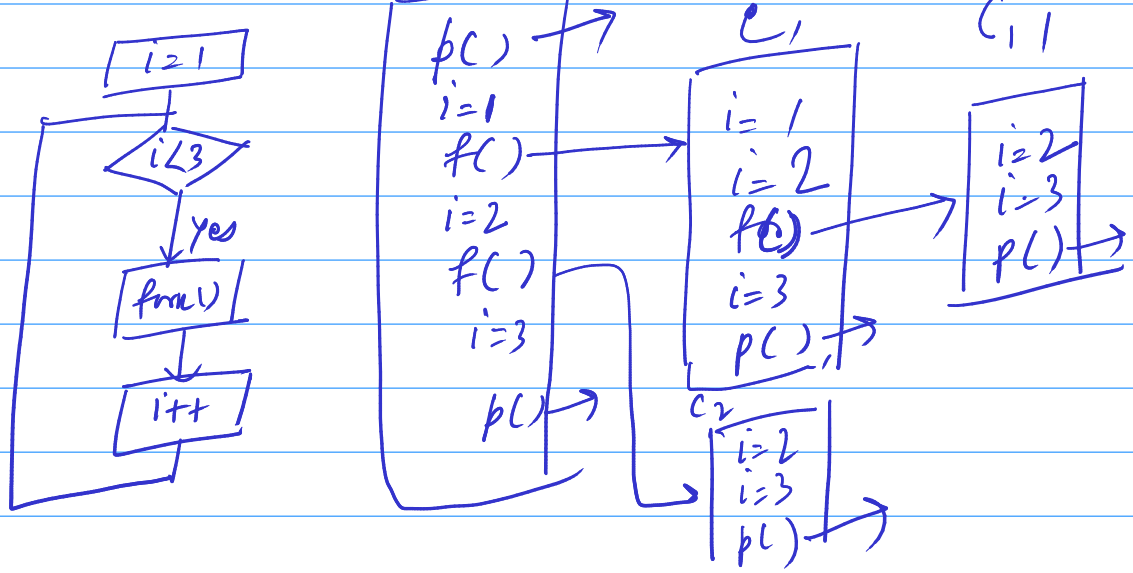
Return Value of fork()

= 0 to the child

≠ 0 to the parent

= pid of the child





P_1, P_2, and P_3 are three processes executing their respective tasks. They should synchronize among themselves using semaphores such that the string "ABCACB" is printed infinite times. Determine, minimum number of semaphores required and their initial values. Also identify places where operations on those semaphore should be inserted in the code of P_1, P_2, and P_3. Describe how your solution works.

P_1	P_2	P_3

while(true){
print("A");
}
	while(true){
	print("B");
	}
		while(true){
		print("C");
		}

X=1 Y=0 Z=0

P1 turn=1
while(T){
wait(X)
print(A);
if(turn==1){
signal(Y)
turn=2
}else{signal(Z)
turn=1
}
}

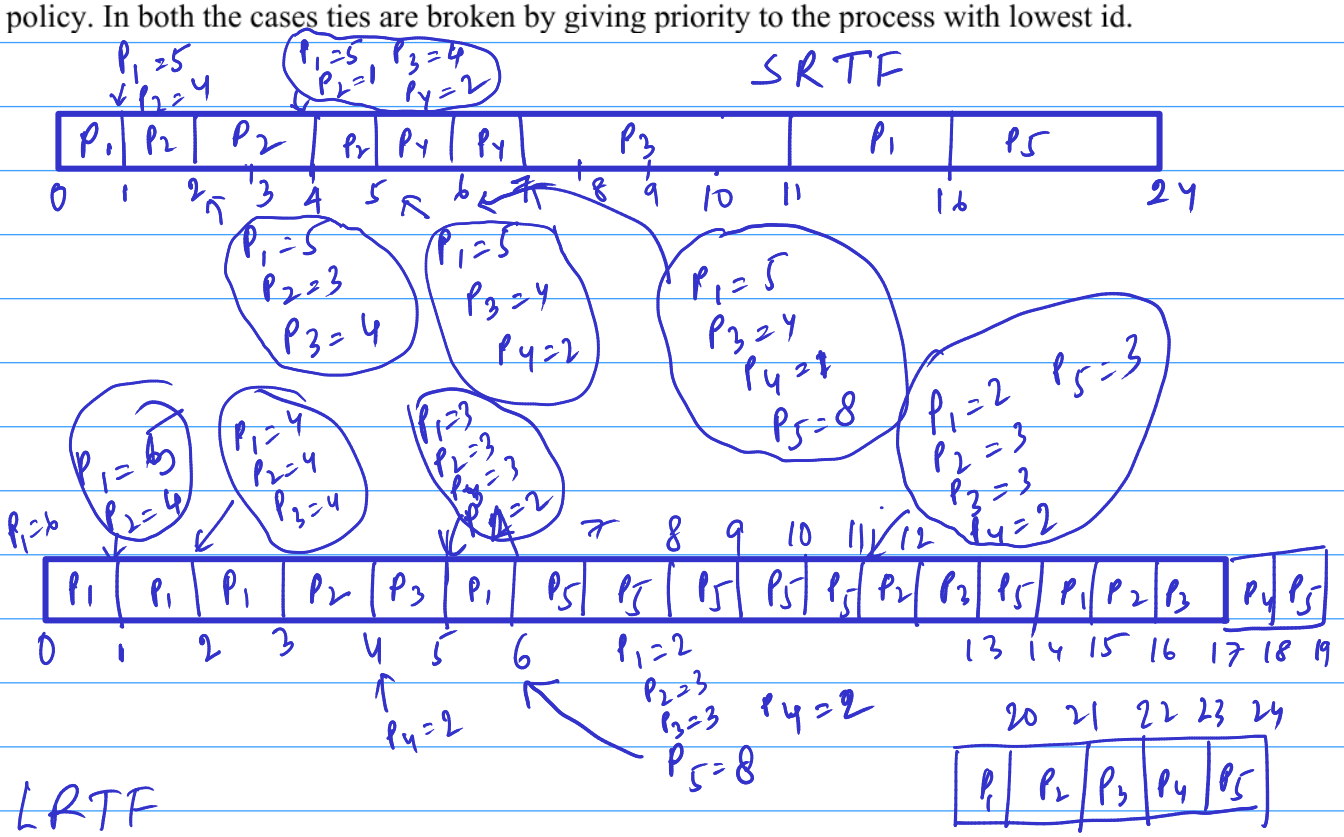
P2 turn=1
while(T){
wait(Y)
print(B)
if(turn==1){
signal(Z)
turn=2
}
else{
signal(X)
turn=1
}
}

P3 turn=1
while(T){
wait(Z)
print(C)
if(turn==1){
signal(X)
turn=2
}
else{
signal(Y)
turn=1
}
}

Consider the following set of processes with the arrival times and the CPU burst times given in milliseconds

Process	Arrival Time	Burst Time
P_1	0	6
P_2	1	4
P_3	2	4
P_4	4	2
P_5	6	8

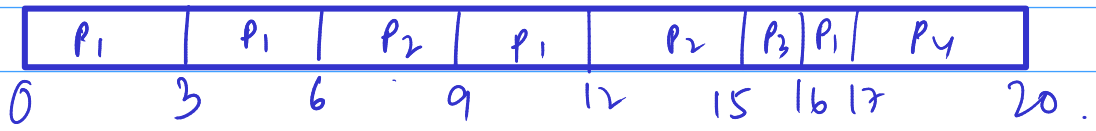
Determine the turnaround time and waiting time for all the processes using Shortest Remaining Time First (SRTF) and Longest Remaining Time First (LRTF) scheduling policy. In both the cases ties are broken by giving priority to the process with lowest id.



Consider the following processes, with the arrival time and the length of the CPU burst. Determine the waiting time, turn-around time of these processes for both the scheduling algorithms (i) preemptive shortest remaining-time first and (ii) round robin (time quanta = 3).

Process	Arrival Time	Burst Time
P_1	0	10
P_2	3	6
P_3	7	1
P_4	8	3

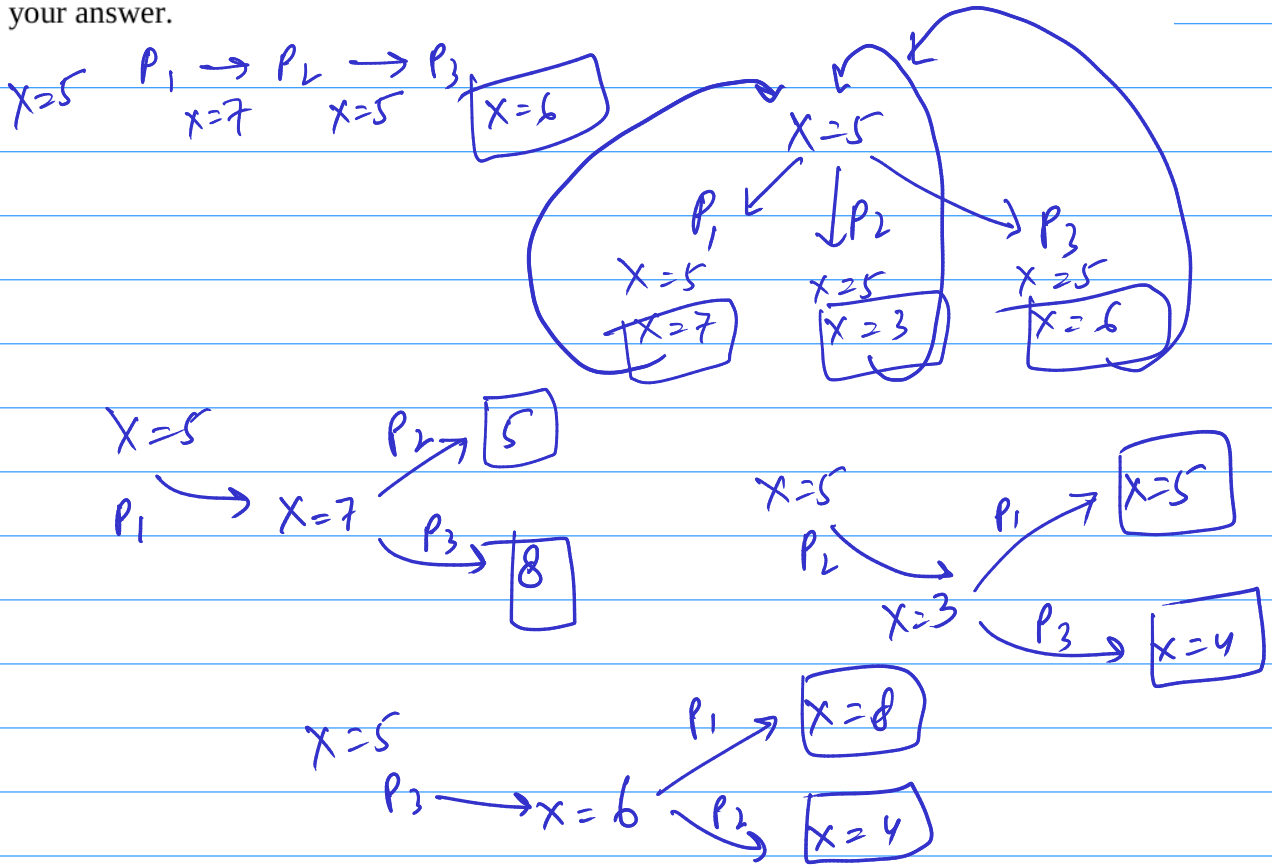
Round Robin



Three processes P_1, P_2 and P_3 access and modify a shared variable x in the following manner. Initial value of shared variable x is 5. Processes do not use any mechanism to impose mutually exclusive access to p.

P_1	P_2	P_3
...	...	...
...	...	...
x=x+2	x=x-2	x=x+1
...	...	...
...	...	...

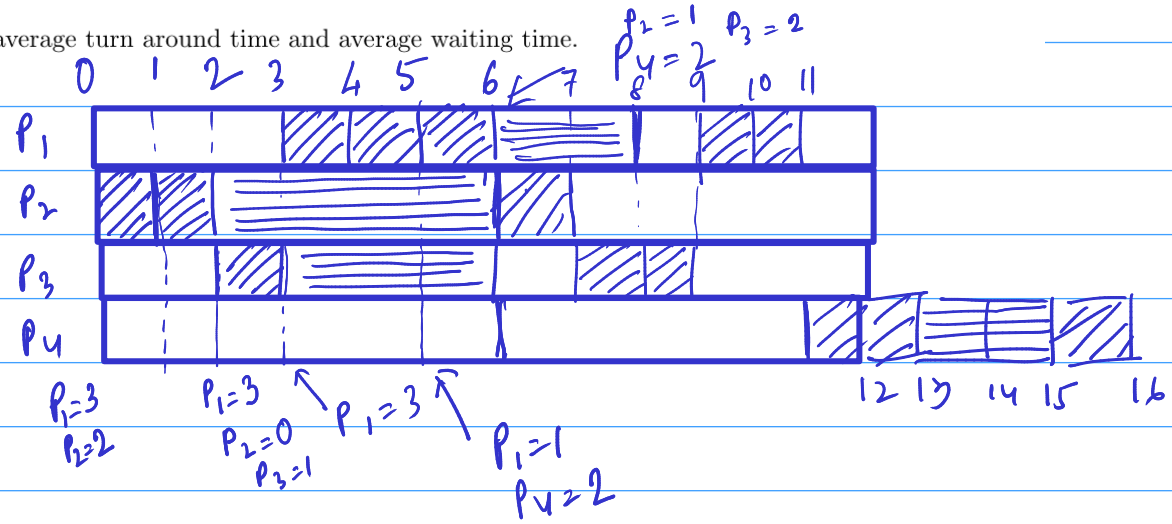
The processes execute on a uni processor system running a time shared operating system. What are the possible values of x after all the three processes finish execution? Justify your answer.



An operating system uses a shortest remaining compute time first scheduling algorithm and schedules a new process either when the running process gets blocked on I/O or when the running process finishes its compute burst. Assume that all I/O operations can be overlapped as much as possible. Consider the set of 4 processes whose arrival times and burst times are given below:

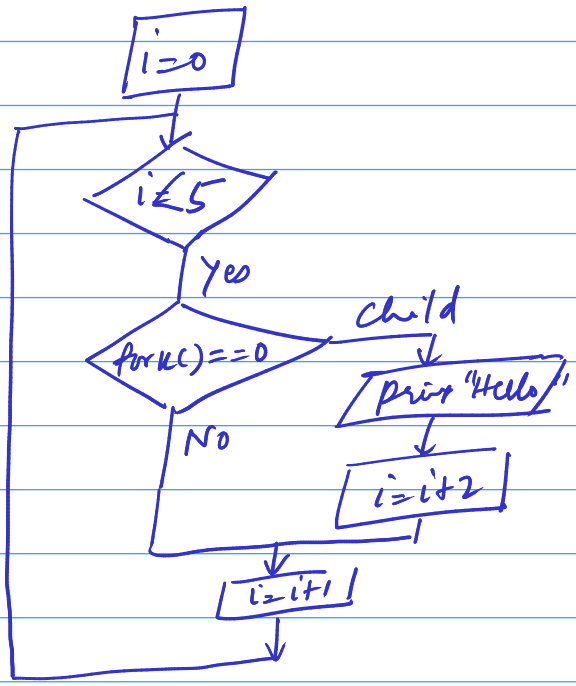
Process	Arrival Time	CPU Burst	I/O Burst	CPU Burst
P_1	0	3	2	2
P_2	0	2	4	1
P_3	2	1	3	2
P_4	5	2	2	1

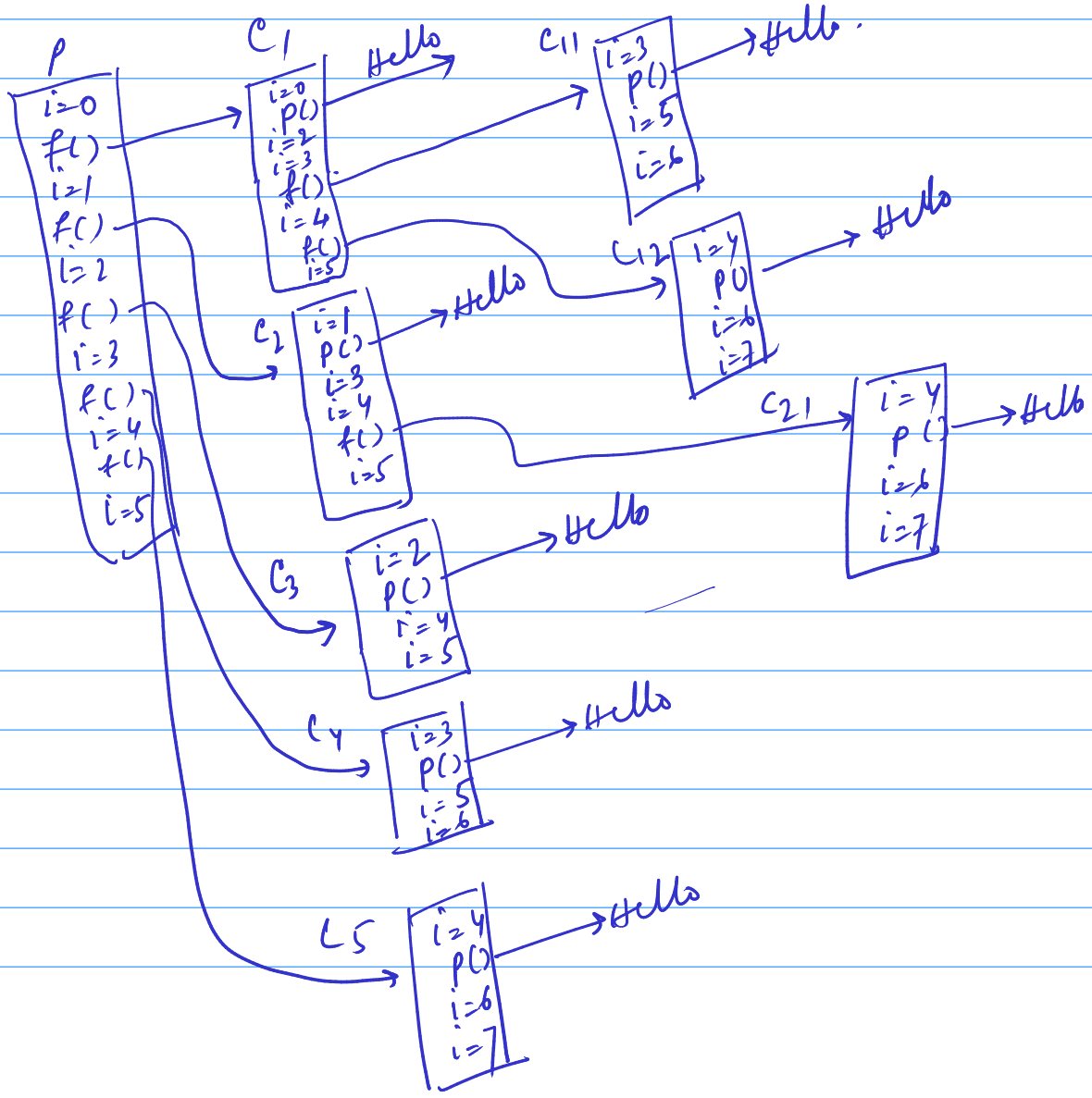
Determine average turn around time and average waiting time.



For the following code segment determine how many times "Hello" will be printed. Provide necessary Justifications.

```
for(i=0;i<5;i++){
    if(fork()==0){
        printf("Hello\n");
        i+=2;
    }
}
```





Hello, pid=7454, ppid=7453
Hello, pid=7455, ppid=7454
Hello, pid=7468, ppid=7453
Hello, pid=7469, ppid=7454
Hello, pid=7470, ppid=7468
Hello, pid=7495, ppid=7453
Hello, pid=7510, ppid=7453
Hello, pid=7511, ppid=7453