Part 3 (36 points) – Theoretical Questions

Question 1

The following code is run by 3 threads:

```
static int x = 0;
void *thread_func(void *p)
{ x++;
}
```

What is the minimal value of x at the end of the run?

The minimal value is 1. The situation: Thread A is run, load zero to the register, increments it, and then a context switch to thread B (before A can store 1 into k). Thread B loads 0, increments it, and then a context switch to thread C (before B can store 1 into k). Thread C loads 0, increments it, and then stores 1 into K. Context switch back to thread B, which finished its process by storing 1 into k and same story for thread A.

Question 2

Consider the following code:

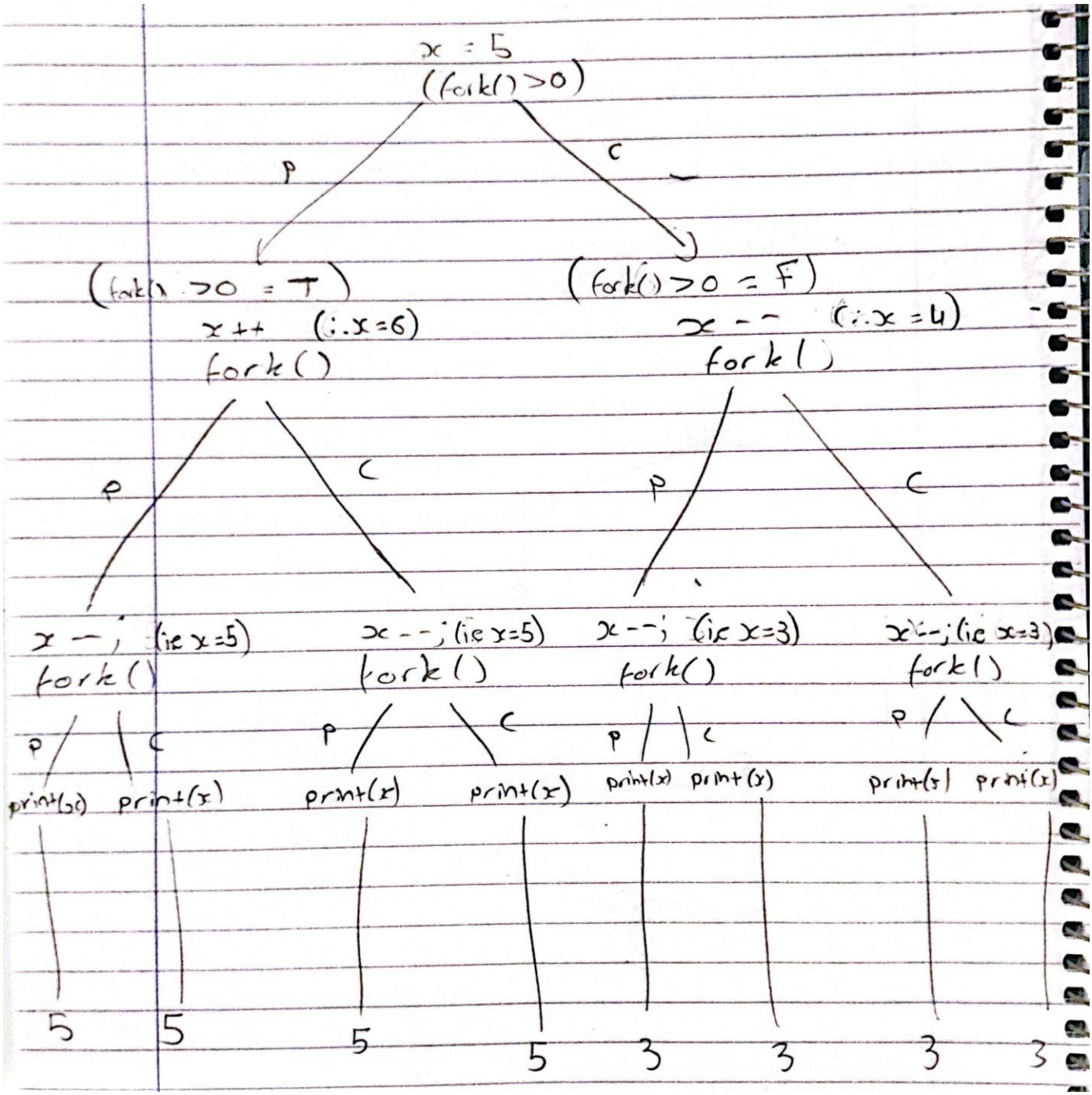
```
int main ()
{
   int x = 5;
   if ( fork () > 0){
      x ++;
      fork ();
   } else {
      x - -;
      fork ();
   }
   x - -;
   fork ();
   printf ( %d\ n , x );
   return 0;
}
```

What will be the output if we run the given code? (the order does not matter)

5
5
5
5
3

3
3
3

Explanation:



Question 3:

When executing the following code, how many times will the line “forked” appear in the output?

```c
int main()
{
   pid_t wpid;
   int status=0;
   for ( int i=0; i<10; i++){
       if( i % 3 == 0){
       fork();
       }
   }
   printf("forked\n");
   // wait for all processes
   while((wpid=wait(&status))>0);
   return 0;
}
```

It will be printed 15 time.

Explanation in the diagram: