1 problem 1

If f(x, y) is a computable function, we know that:

For all input x, if the program halts, it will return f(x).

if the program doesn't halt (this situation will never happen, because y=f(x) is a C program. The C program must halt on any input.)

This means for all input x, we can get y=f(x)(f(x,y)). Then for f(f(x,y),y), for all input x, it will return f(f(x,y)) which is f(y). So if f(x,y) is computable function, then so is f(f(x,y),y).

2 problem 2

f(x) is a totally computable function, this means that f(x) is defined for all x.

we know g is :

g(0) = 1

g(n+1) = f(g(n)).

From 0 to n. g(0)=1, g(1)=f(g(0))=f(1), g(2)=f(f(1)),...., g(n+1)=f(g(n)).

Since for all x, there is a concrete f(x) corresponding to it. For all n, there is a concrete g(n) corresponding to it. This means that g is also a totally computable function.

3 problem 3

L is a regular language, so there is an algorithm M recognize L. For all input string w, if w$\in$L, M says yes. If w$\notin$L, M says no. Enumerate all w$\in$L, from n=0 to n=N, check: if there is a double word that length$\geq$n, f(n)=1. else f(n)=0. This means f(n) is defined for all n. So f is a totally computable function.

4 problem 4

First I construct a FA M' run on w(which is double word ww). I run two copies , M1 and M2, of M, in parallel on w. M1 starts form initial state of M, M2 start form a guess state of M. At the end of w, I will make sure that M1 ends at the guess state and M2 ends at the accepting state of M.
Since L is a regular language, I have a FA M to accept it. From n=0 to n=N. For the n, we know that at each loop n is a fixed number. Iff $\exists$w, w$\in$L(M') and —w—$\geq$n/2, f(n)=1, else f(n)=0. So we know that f(n) is defined for all n. So f is a totally computable function.

5 problem 5

F is to change three numbers into three other numbers. The whole process is how three numbers become other three numbers. And the relationship between each step in

the transformation process is the same. So F terminates iff $\exists$n. G(n, 0, 0, 0; 1, 1, 1)—— iff M starts with x=0, y=0 from line 0 will reach line1 with x=1, y=1.

6 problem 6

I can simulate the LP problem as a Presburger problem. $\exists$K, for all x1,...,xn, y$\in$Z. (K$\geq$f(x1,...,xn)) $\vee$ $\exists$K, $\exists$x1,...,xn, y$\in$Z. (K=f(x1,...,xn))
Beacuse the Presburger formula is decidable. so the LP problem is decidable.

7 problem 7

For example $\exists$x. (3x-4y=7).

$\exists$x. (x=1/3(7+4y)).

I rename 1/3(7+4y) as C1.

Then we get $\exists$x. (x=C1).

For C1, I am going to translate the original formula with qualifier $\exists$x into one without it.

For this case, x have three possible positions.

x<C1— x=C1-1

x=C1— x=C1

x>C1— x=C1+1

(C1-1=C1) $\vee$ (C1=C1) $\vee$ (C1+1=C1)

Finally $\exists$x. (x=C1) is translate into: (C1-1=C1) $\vee$ (C1=C1) $\vee$ (C1+1=C1). For this formula we don't have

qualifier over x and variable x.
So for all integers x1,···, xn, P(x1,···,xn) holds iff F(x1,···, xn) holds

8 problem 8

If I can use 4 water tanks to simulate a Two-counter machine, whether G is terminating is undecidable. Firstly, I have four water tanks named T1, T2, T3, T4. I use T1 and T2 to simulate the two counters x and y. For x==0 and y==0, I can use the instruction of water-level-test Water-Level(i)=1? to simulate. This is very simple imitation. However, it is hard to use 4 water tanks to simulate the instructions like x++, x–, y++, y–. For instance, if I want to use 4 water tanks to simulate the instructions x++ and x++. Since the q is a guess number, It is very difficult to keep the same amount of water increased twice. Although this problem is difficult, I can use T3 and T4 to solve this problem. First I add a very small amount q to T3. Think of q as the "1" in the x+1 instruction. (Because q is very small, so don't worry about it exceeding the boundary) Then perform (in, stay, out, in) on the four tanks to simulate x++. And after the instruction ends, I do test T3=1? If T3=1 continue, otherwise crash. After this, I perform (in, stay, in, out) to simulate the second instruction x++. And do test T4=1? If T4=1 continue, else crash. This ensures that the amount of water added to T1 is constant each time. I can also use the same method to simulate x–,

y++ and y–. So the hypothesis of using four water tanks to simulate Two-counter machine is true. In other words, whether G is terminating is undecidable.

9 problem 9

First, construct an NFA from A1,...,Ar B accepts a string w iff wi is a subsequence of Ai, and Ai accepts wi. Then we construct an algorithm C that agrees with a string x, if it is accepted by A but not by B. C enumerates all symbols x and then simulates running A and B in parallel on x. To imitate B, C uses the "subset construction" technique and updates the subset as it processes the symbols of x. It knows that C finds that there is no acceptable state in the subset. If A says yes and B says no, C says yes. So if C says yes on x, the system is distributed-able.

10 problem 10

The T system can be regarded as a molecular delivery system. From configuration (S, θ) to (S', θ'), we can simply change from θ signal to θ'signal, and remove all a in configuration S, and then every time removed a in S, we add an object-b and an object-c. To determine whether a system T is bounded, we only need to determine whether T contains a bounded w-execution. w-execution is a sequence of T starting from (S0,θ0) to (Sn,θn), if all (Si,θi)

will not halting, then this w-execution is bounded. As for any configuration (S, θ), there are many configurations in its next step, and which configuration goes next step is non-deterministic. So I don't think we can decide whether T is bounded.