Part1:

The Church-Turing thesis is a thesis in computer science named after mathematicians Alonzo Church and Alan Turing. The most basic view of the thesis states that all computations or algorithms can be performed by a Turing machine. A computer program written in any conventional programming language can be translated into a Turing machine, and conversely, any Turing machine can be translated into a large program in most programming languages, so this thesis is equivalent to the following statement: A conventional programming language can suffice Efficient to express any algorithm. This thesis is generally assumed to be true and is also known as Church's thesis or Church's conjecture and Turing's thesis. Turing's concern was more general than Church's, because Church only considered functions of positive integers, whereas Turing described his work as a computable function involving "integer variables or a real or computable variable, computable predicate, etc.

Strong Church-Turing thesis believes that a Turing machine can compute anything that any computer can compute. It can solve all problems that can be represented as computations (far beyond computable functions). In fact, the Strong Church-Turing thesis is incorrect. The equivalence between the Strong Church-Turing Thesis and the original paper on Turing machines is a myth, and the Strong Church-Turing thesis argues that all computable problems can be solved using integers It can be expressed as a function of integers, so it can be captured by a Turing machine. All computable problems can be described by algorithms. Algorithms are what computers do.

Goldin, D., & Wegner, P. (2008). The interactive nature of computing: Refuting the strong Church–Turing thesis. *Minds and Machines*, *18*(1), 17-38.

Abramson, F. G. (1971, October). Effective computation over the real numbers. In *SWAT (FOCS)* (pp. 33-37).

Turing, A. M. (1937). Computability and λ-definability. *The Journal of Symbolic Logic*, *2*(4), 153-163.

0,0,R

X,X,R

Part2:

0,X,R

C,C,R

0,0,L

0,X,L

C,C,L

C,C,R

X,X,R

0,0,L

X,B,R

a,a,R

b,b,R

Part3:

B:blank

B,B,L

a,B,R

a,B,R

a,B,R

b,B,R

B,B,R

B,B,R

b,B,L

a,a,L

b,b,L

Part4:

use the closure properties:

L1= {a,b} is context free language : S-> a|b

L2 = {,n >= 0} is context free language : S->aS |

So L1L2 is context free language ,so L is CFL.