Ejercicio 1. Responder breve y claramente los siguientes incisos:

1. ¿Qué es un problema computacional de decisión? ¿Es el tipo de problema más general que se puede formular?
2. ¿Qué cadenas integran el lenguaje aceptado por una MT?
3. En la clase teórica 1 se hace referencia al problema de satisfactibilidad de las fórmulas booleanas. Formular las tres formas del problema, teniendo en cuenta las tres visiones de MT consideradas: calculadora, aceptadora o reconocedora, y generadora.
4. ¿Qué postula la Tesis de Church-Turing?
5. ¿Cuándo dos MT son equivalentes? ¿Cuándo dos modelos de MT son equivalentes?
6. ¿En qué difieren los lenguajes recursivos, recursivamente numerables no recursivos, y no recursivamente numerables?
7. Probar que $R \subseteq RE \subseteq \mathfrak{L}$.
8. ¿Todo lenguaje de la clase CO-RE tiene una MT que lo acepta?
9. Justificar por qué los lenguajes Σ^* y $\emptyset$ son recursivos.
10. Si $L \subseteq \Sigma^*$, ¿se cumple que $L \in R$?
11. Justificar por qué un lenguaje finito es recursivo.
12. Justificar por qué si $L_1 \in$ CO-RE y $L_2 \in$ CO-RE, entonces $(L_1 \cap L_2) \in$ CO-RE.
13. Dados $\Sigma = \{a, b, c\}$ y $L = \{a^n b^n c^n \mid n \geq 0\}$, obtener $\Sigma^* \cap L$, $\Sigma^* \cup L$, y el complemento de L respecto de Σ^*.

1. Un problema computacional de decisión es un tipo de problema en el cual se da una respuesta por sí o no. Específicamente, se determina si el string que se recibió, pertenece al conjunto de palabras que la MT reconoce.
   No, el más general es el problema computacional de búsqueda, en el que, dado un input, nos devuelve el procesamiento de este como un output.
2. Las cadenas que integran el lenguaje aceptado por una MT son las que la máquina reconoce (esto se hace cuando paran en qA)
3. Una maquina aceptadora es aquella que utiliza una MT, que para un string dado, determine si existe una combinación de valores en sus variables en la que obtengamos como resultado verdadero (que sea satisfactible). Entonces, tenemos que construir una máquina que pruebe todas las combinaciones de valores de verdad en la fórmula. En caso de que alguna combinación satisfaga la fórmula, la máquina se para en qA, si ninguna lo hace, para en qR. La máquina siempre para porque a lo sumo las combinaciones de valores de verdad van a ser 2^n (n es el número de variables de la fórmula)

   Una maquina calculadora es aquella que utiliza una MT, que para un string dado, determine si hay una combinación de valores en sus variables en la que obtengamos como resultado verdadero e imprima la combinación en la cinta que actúa como output. Cuando se encuentre una cadena que pueda ser satisfecha, escribe en la cinta de output la asignación de valores que satisfacen la cadena, si no existe una combinación que la satisfaga se deja la cinta en blanco.

   Una maquina generadora es aquella que utiliza una MT que genere todas las fórmulas booleanas satisfactibles, entonces se necesita crear todas las cadenas booleanas posibles, y en cada una evaluar lo siguiente:

Si la fórmula es rechazada, se genera la próxima y se vuelve a analizar.
Si la fórmula es aceptada, la escribe en la cadena de output y después se pone un símbolo para diferenciarla del siguiente string. Se debería seguir un orden a la hora de generarlas para asegurar de que no se repitan las formulas analizadas.

4. La Tesis de Church-Turing plantea que todo lo computable puede ser resuelto por una MT
5. Dos MT son equivalentes cuando reconocen el mismo lenguaje. Dos modelos de MT son equivalentes cuando para cada MT de un modelo se puede encontrar una MT' equivalente del otro.
6. Los lenguajes recursivos son aquellos en los que la MT va a parar siempre, en los recursivamente numerables la MT puede parar o quedarse loopeando, los no recursivos son los que no se pueden decidir con una MT que siempre pare, y los no recursivamente numerables son lenguajes que no tienen una MT que lo acepte.
7. $R \subseteq RE$ se demuestra por definición, la definición de RE es igual que la de R pero con una restricción menos (la que permite a la MT quedarse loopeando).
   $RE \subseteq \mathfrak{L}$ es demostrado por definición, ya que $\mathfrak{L}$ es el conjunto de todos los lenguajes, por ende cualquier conjunto de lenguajes está incluido en él, entre otros RE.
8. No, solo se podrían aceptar los lenguajes que pertenecen a R dentro del CO-RE
9. Son recursivos porque se puede crear una MT que los reconozca y acepte siempre. Para Σ* hay que hacer una MT que acepte cualquier string, y para ∅ hay que hacer una MT que rechace siempre
10. Esto es falso, porque todos los lenguajes $L \subseteq \Sigma^*$ pero no todos pertenecen a R (como el Halting Problem)
11. Un lenguaje finito es recursivo porque se puede hacer una MT que pare siempre, ya que, para cualquier lenguaje finito, se podría hacer una MT con un estado por carácter y con eso se pueden reconocer todos los strings.
12. Siendo que: $L1 \in CO\text{-}RE$ y $L2 \in CO\text{-}RE$
    Por la definición de CO-RE $L1^C \in RE$ y $L2^C \in RE$ y la segunda cláusula de RE (si ambos están en RE, su unión está en RE)
    $L1^C \cup L2^C \in RE$ por las leyes de Morgan $L1^C \cup L2^C \in RE$ entonces $(L1 \cap L2)^C \in RE$ y nuevamente por definición de CO-RE si $(L1 \cap L2)^C \in RE$ entonces $L1 \cap L2 \in CO\text{-}RE$.
13. $\Sigma^* \cap L = L$
    $\Sigma^* \cup L = \Sigma^*$
    El complemento de L respecto a Σ* es L^C, o se podría ver como Σ* - L.

Ejercicio 2. Construir una MT, con cualquier cantidad de cintas, que acepte de la manera más eficiente posible el lenguaje $L = \{a^n b^n c^n \mid n \geq 0\}$. Plantear primero la idea general.

Idea general:
Dada una MT ML que acepta L y para siempre (hipótesis)

Construcción:
M tiene 2 cintas. Con el input w en la cinta 1, M hace:

1. Escribe en unario la cantidad de “a” que hay en el input w sobre la cinta 2.
2. Recorre la cinta 2 controlando que haya igual cantidad de “b” que de unos, en caso de no cumplirse para en qR.
3. Repite el paso anterior controlando las “c”, en caso de cumplirse para en qA y en caso contrario en qR

Ejercicio 3. Explicar (informal pero claramente) cómo simular una MT por otra que en un paso no pueda simultáneamente modificar un símbolo y moverse.

Habría que hacerlo en 2 pasos, en el primero modificar el símbolo y en el siguiente hacer el movimiento requerido

Un ejemplo sería:
$\delta(q0, 0) = (q1, 1, L)$

Pasaría a:
$\delta(q0, 0) = (q1, 1, S)$
$\delta(q1, 1) = (q1, 1, L)$

Ejercicio 4. Sean L_1 y L_2 dos lenguajes recursivamente numerables de números naturales representados en notación unaria (por ejemplo, el número 5 se representa con 11111). Probar que también es recursivamente numerable el lenguaje $L = \{x \mid x$ es un número natural representado en notación unaria, y existen y, z, tales que $y + z = x$, con $y \in L_1, z \in L_2\}$.
Ayuda: la prueba es similar a la vista en clase de la clausura de RE respecto de la concatenación.

Al estar los números representados en notación unaria, se puede pensar la suma como una concatenación por ejemplo, 11 + 111 = 11111 y 11 . 111 = 11111, entonces se puede pensar en que x = yz.
En este ejemplo la forma de probarlo es con la clausura de RE con w tomando el valor de x, w2 el de z y w1 el de y.
Si L1 $\in$ RE y L2 $\in$ RE, entonces L1 . L2 $\in$ RE, siendo L1 . L2 el lenguaje concatenación (o producto) de L1 con L2 , es decir: L1 . L2 = {w | w = w1w2 , con w1$\in$ L1 y w2 $\in$ L2 }

Ejercicio 5. Dada una MT M_1 con $\Sigma = \{0, 1\}$:
1. Construir una MT M_2 que determine si $L(M_1)$ tiene al menos una cadena.
2. ¿Se puede construir además una MT M_3 para determinar si $L(M_1)$ tiene a lo sumo una cadena? Justificar.

Ayuda para la parte (1): Si $L(M_1)$ tiene al menos una cadena, entonces existe al menos una cadena w de unos y ceros, de tamaño n, tal que M_1 a partir de w acepta en k pasos. Teniendo en cuenta esto, pensar cómo M_2 podría simular M_1 considerando todas las cadenas de unos y ceros hasta encontrar eventualmente una que M_1 acepte (¡cuidándose de los posibles loops de M_1!).

1. Los pasos para la construcción de esta máquina serían:
    1) Hacer i := 1.
    2) Copiar i cadenas en una cinta y hacer i pasos en cada una, si encuentra la cadena para en qA.
    3) i:=i+1

4) Vuelve al paso 2

De esta manera la maquina buscara de manera indefinida hasta que se determine si L(M1) tiene al menos una cadena.

2. No se puede porque para saber si acepta solo 1, hay que probar con el conjunto de todos los strings posibles (infinito)