$\frac{100}{100} + 10 = \frac{110}{100}$

CS 3350 Automata, Computability, and Formal Languages
Fall 2016, Test 2

10/10

Name: [redacted]

1. Use the algorithm that we had in class to transform the following context-free grammar into Chomsky normal form. This grammar describes simple arithmetic expressions (E), combining numbers (N) 0 and 1 with + and -. The starting variable is E, the rules are: E --> N, E --> ~~N+S~~ N+E, N --> 0, N --> 1, E --> N−E.

$E \to N$ (1)

$E \to N+E$ (2)

$N \to 0$ ✓

$N \to 1$ ✓

$E \to N-E$ (2)

$S_0 \to E$ (1)

step 0

step 1

$E \to 0$ ✓

$E \to 1$ ✓

$S_0 \to N+E$ (2)

$S_0 \to N-E$ (2)

$S_0 \to 0$ ✓

$S_0 \to 1$ ✓

step 2

$V_0 \to 0$ ✓

$V_1 \to 1$ ✓

$V_+ \to +$ ✓

$V_- \to -$ ✓

$V_{N+} \to NV_+$ ✓

$E \to V_{N+}E$ ✓

$V_{N-} \to NV_-$ ✓

$E \to V_{N-}E$ ✓

$S_0 \to V_{N+}E$ ✓

$S_0 \to V_{N-}E$ ✓

Rules

$N \to 0$

$N \to 1$

$E \to 0$

$E \to 1$

$E \to V_{N+}E$

$E \to V_{N-}E$

$S_0 \to 0$

$S_0 \to 1$

$S_0 \to V_{N+}E$

$S_0 \to V_{N-}E$

$V_0 \to 0$

$V_1 \to 1$

$V_+ \to +$

$V_- \to -$

$V_{N+} \to NV_+$

$V_{N-} \to NV_-$

10/10

2. Use the general algorithm to design a (non-deterministic) pushdown automaton that recognizes exactly the context-free language described in Problem 1. Show, step-by-step, how a word 1+0 will be accepted by this automaton.

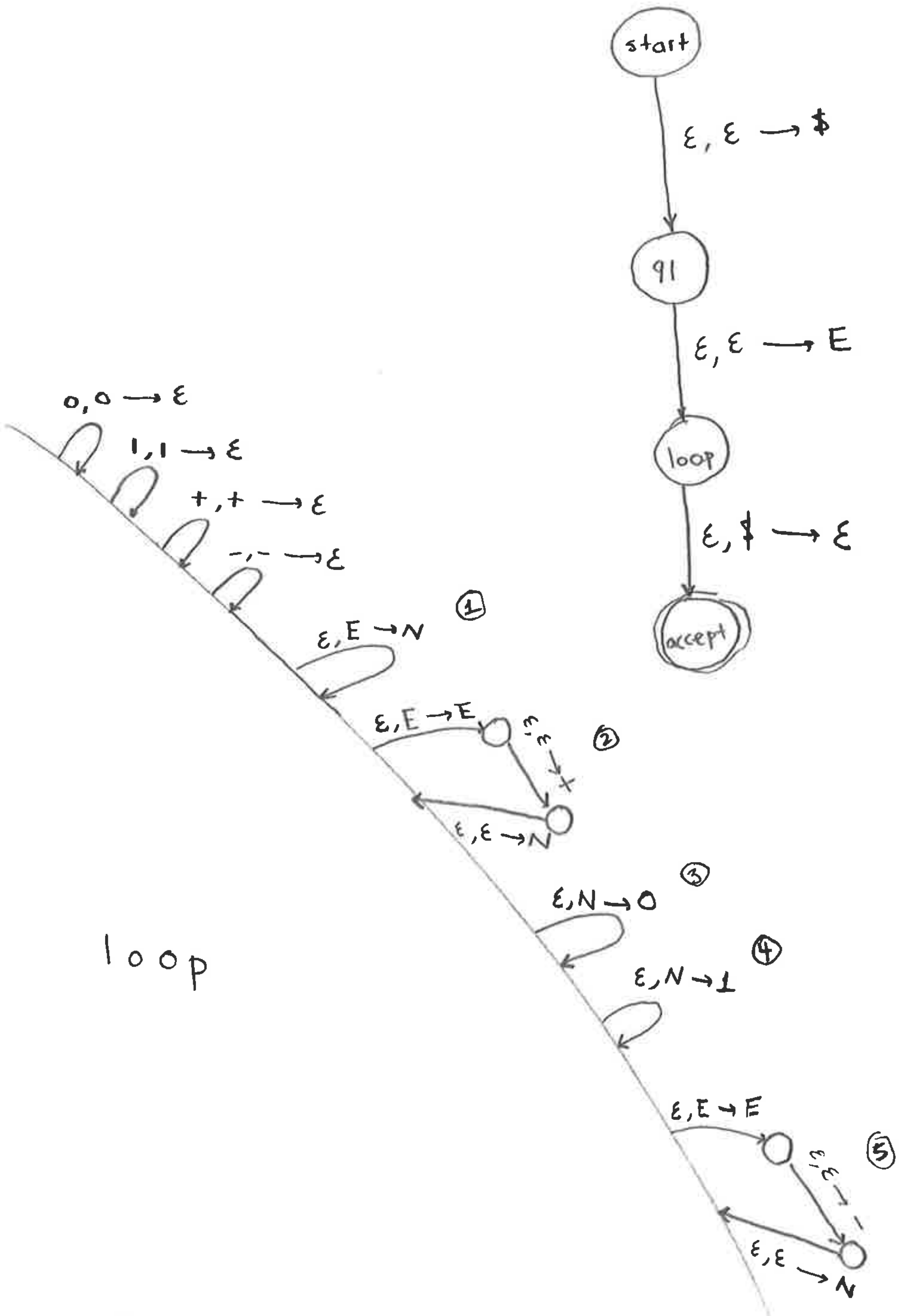
$E \to N$ ①

$E \to N + E$ ②

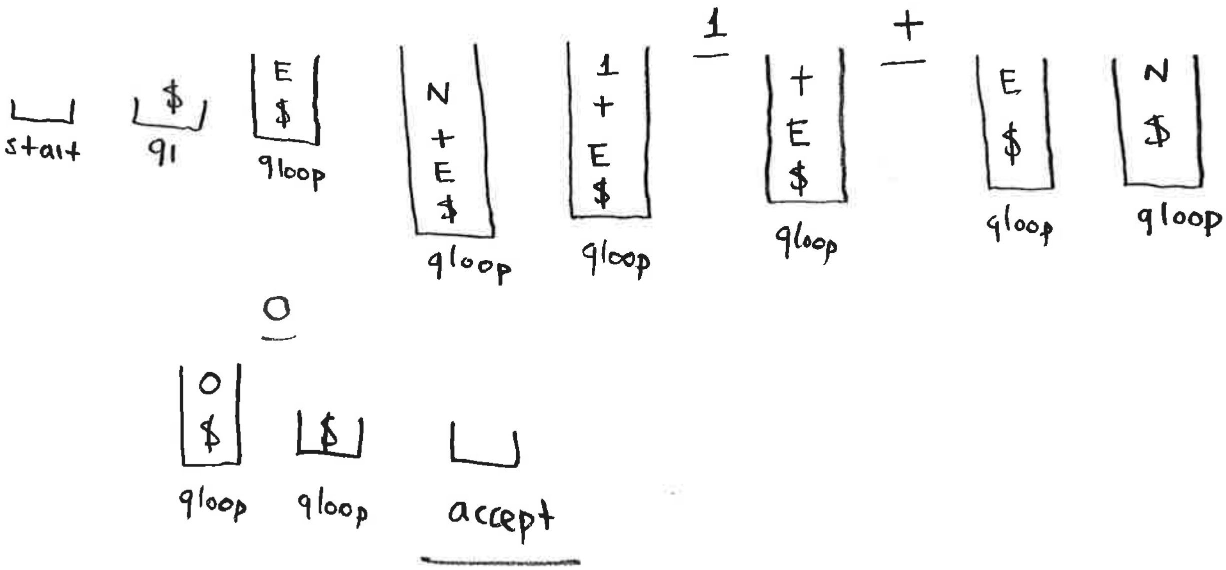
$N \to 0$ ③

$N \to 1$ ④

$E \to N - E$ ⑤

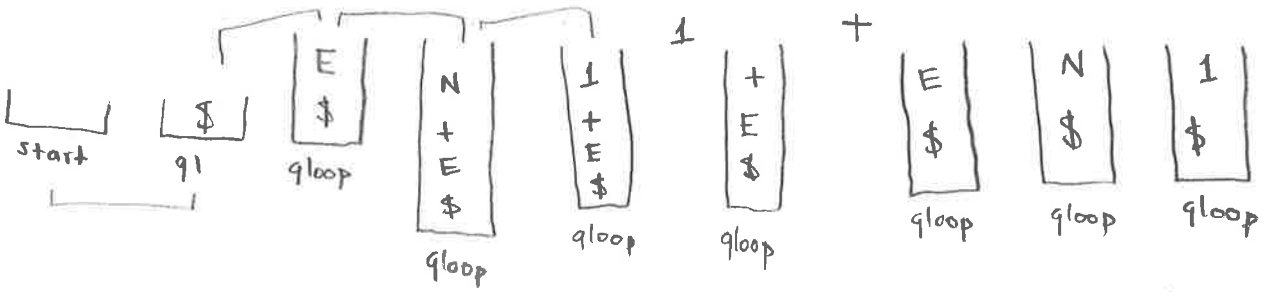


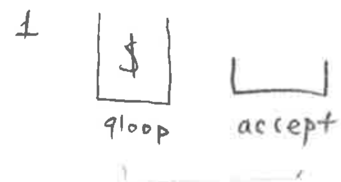
1 + 0


start
$
q1
E
$
qloop
N
+
E
$
qloop
1
+
E
$
qloop
1
+
E
$
qloop
+
E
$
qloop
N
$
qloop
0
0
$
qloop
$
qloop
accept

10/10

3. Apply, to pushdown automaton that you designed in Problem 2, the general algorithm of transforming a pushdown automaton into a context-free grammar, and show how this new grammar will generate the word 1+1. (If you are running out of time, just a few steps of this algorithm will be OK.)



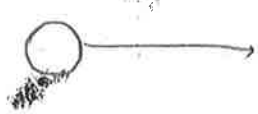


s → start

f → accept

ℓ → loop

$A_{sf} \longrightarrow \varepsilon A_{1\ell} \varepsilon \Rightarrow A_{sf} \longrightarrow A_{1\ell}$



General Rules

$A_{pp} \to \varepsilon$

$A_{pr} \to A_{pq} A_{qr}$

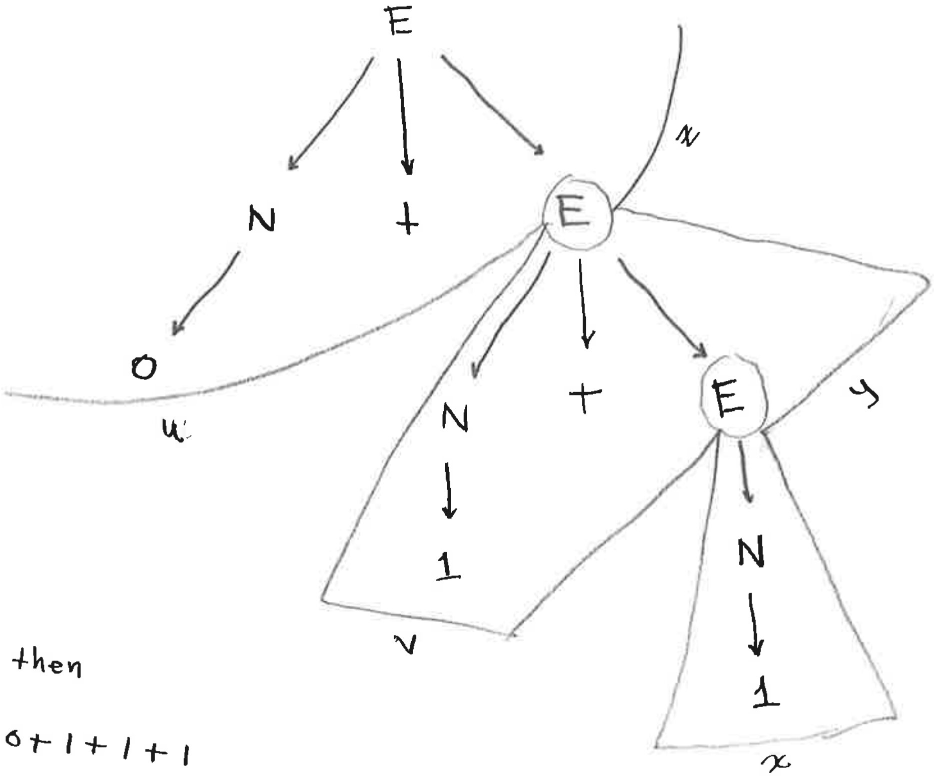
$A_{pq} \to a\, A_{r}\, b$

10/10

4. In the context-free grammar described in Problem 1, we can derive the sequence 0+1+1 as follows:

- first, we use the rule E --> N+E;
- then, we use the rules N --> 0 and E --> N+E;
- after that, we use the rules N --> 1 and E --> N;
- finally, we use the rule N --> 1.

Draw a tree that describes this derivation. Use this tree to determine the subdivision of this sequence into u, v, x, y, and z. Then, for i = 2, draw a tree explaining how the corresponding sequence uv^ixy^iz can be derived in this grammar.



$u = 0+$

$v = 1+$

$x = 1$

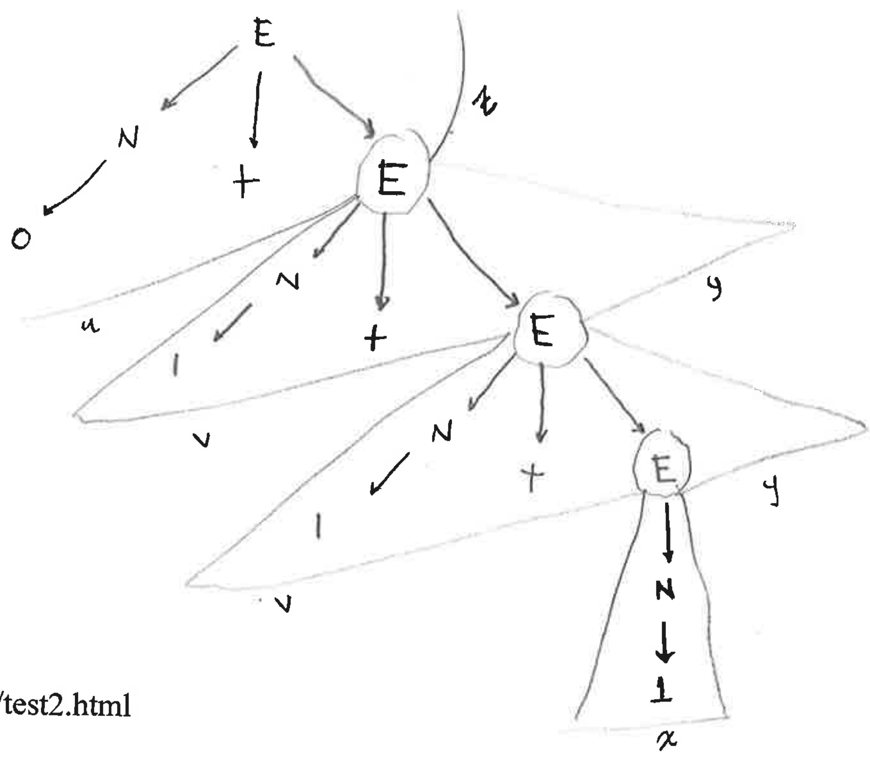
$y = \varepsilon$

$z = \varepsilon$

$uvxyz = 0+1+1$

if $i = 2$, then

$uvvxyyz = 0+1+1+1$



10|10

5. Use pumping lemma for pushdown automata to prove that the language consisting of all the words of the type $1^n0^n1^n0^n$ cannot be recognized by a pushdown automaton. Can you use the same lemma to prove that the language 1^n0^n cannot be generated by a context-free grammar?

Proof: Let's assume that L can be recognized by CFG
Then, by pumping lemma, there exists a p such that every word S from L whose length is at least p can be represented as $s = uvxyz$, where $len(vy) > 0$, $len(vxy) \le p$, and for any i $uv^ixy^iz \in L$

Let's take $S = 1^p0^p1^p0^p \in L$, here $len(s) = 4p \ge p$, So it can be represented as $uvxyz$ - by pumping lemma, let's considered all possible cases that hold $len(vxy) \le p$.
we take $s = uvvxyyz \in L$
but,

- If vxy is in first 1's, first 0's, second 1's or second 0's, means that when we pump we will add more of that symbol, disrupting the balance.
- If vxy is in first 1's and first 0's, then if we pump we will have more first 1's and first 0's than second 1's and second 0's. Simirlarly, if vxy is in first 0's and second 1's and in second 1's and second 0's

Any combination gives us a contradiction, since the word is not part of L. So, cannot be recognized by PDA. It is not a contex free

second question

$1^n 0^n$

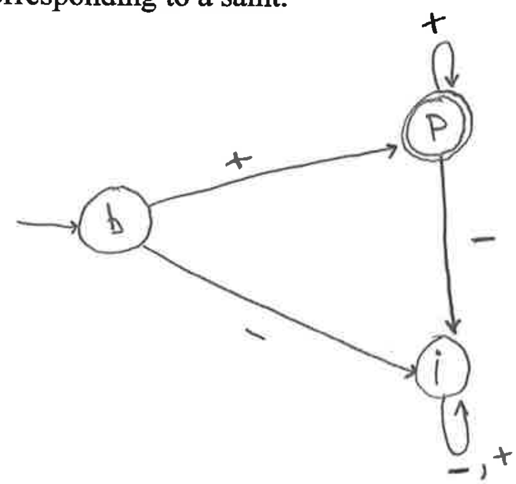
I think that $1^n 0^n$ can be recognized by a CFG since vxy can be between 1's and 0's.

10/10

6. Use the general algorithm to design a Turing machine that is equivalent to the following finite automaton. This automaton helps to check whether a person is a saint. The alphabet consists of two symbols: + ("good deed") and – ("bad deed"). A person is a saint if he or she performed at least one good deed and none bad deeds. The automaton has three states: b ("born", starting state), p ("perfect so far", final state), and i ("imperfect").

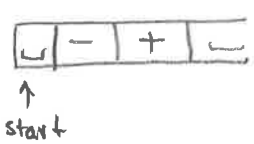
- From each state, – leads to the state i.
- The state i is a sink (once in i, we stay in i).
- From p, + stays in p.
- From b, + leads to p.

Show, step-by-step, how this Turing machine will reject a sequence consisting of a minus; and a plus as *not* corresponding to a saint.

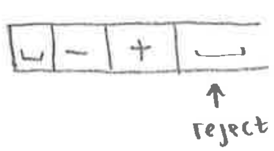


start, ⊔ ⟶ B, R
B, ⊔ ⟶ reject
B, – ⟶ i, R
B, + ⟶ P, R
P, ⊔ ⟶ accept
P, – ⟶ i, R
P, + ⟶ P, R
i, ⊔ ⟶ reject
i, – ⟶ i, R
i, + ⟶ i, R

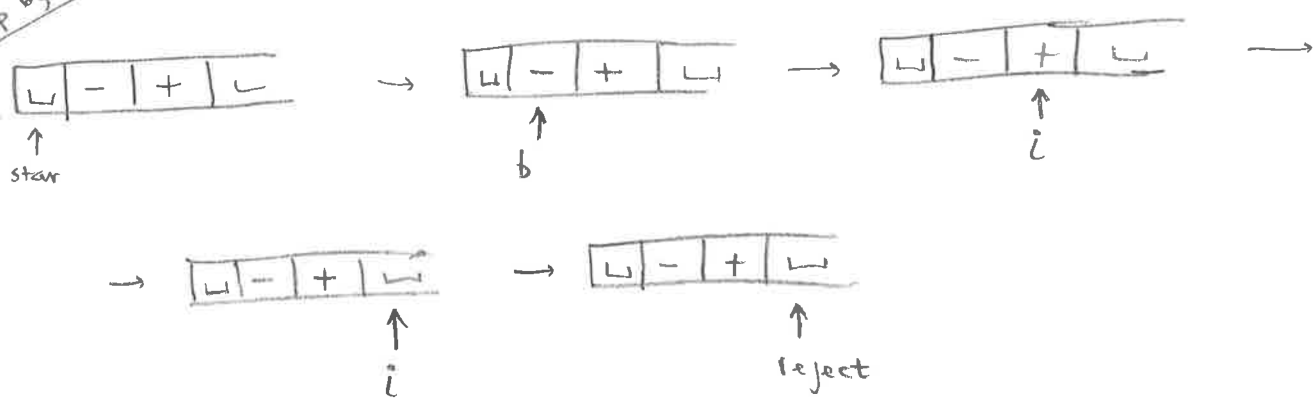
we have



we want



step by step



10/10

7. Use the general stack-based algorithms to show:

- how the compiler will transform the expression 2 + (3 – 4 * 5) into inverse Polish notation, and
- how it will compute the value of this expression.

2 + (3 - 4 * 5)

2 23 234 2345 2345 * – 2 3 4 5 * – +

	+	(	–	*	*	+	
		+	(	–	–		
			+	(	(		
				+	+		

3 - 20 = -17

2 3 4 5 * – +

	2	3	4	5	20	-17	-15	
		2	3	4	3	2		
			2	3	2			
				2				

Result = – 15

20/20

8-9. Design a Turing machine that computes the function n + 2 = n + 10_2 in binary. Trace it on the example of n = 1. Show, step-by-step, how the tape of the Turing machine can be represented as two stacks.

case 0

$$\begin{array}{r} 1011 \\ +\quad 10 \\ \hline 1101 \end{array}$$

case 1

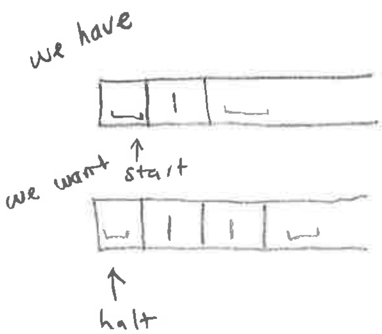
$$\begin{array}{r} 1111 \\ +\quad 10 \\ \hline 10001 \end{array}$$

case 2

$$\begin{array}{r} 1110 \\ +\quad 10 \\ \hline 10000 \end{array}$$

case 3

$$\begin{array}{r} 1100 \\ +\quad 10 \\ \hline 1110 \end{array}$$



start, ␣ ⟶ check0, R

check0, ␣ ⟶ 0, add2, R

add2, ␣ ⟶ 1, back, L

check0, 0 ⟶ add2, R

add2, 0 ⟶ 1, back, L

add2, 1 ⟶ 0, replace, R

replace, 1 ⟶ 0, replace, R

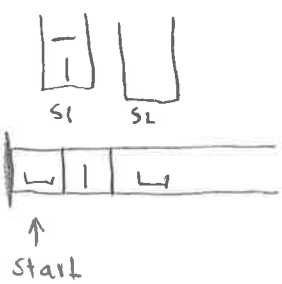
replace, 0 ⟶ 1, back, L

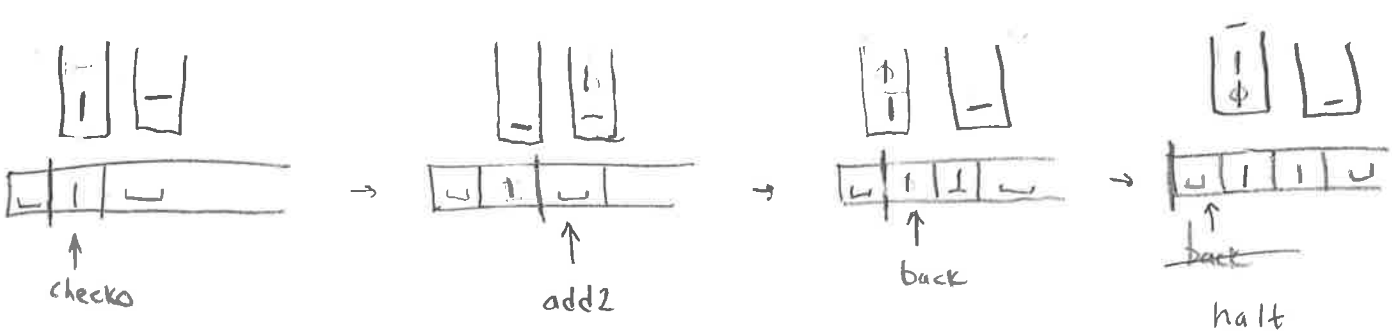
replace, ␣ ⟶ 1, back, L

back, 0 ⟶ back, L

back, 1 ⟶ back, L

back, ␣ ⟶ halt





10. Computability:

10/10

- Formulate Church-Turing thesis.
- Is it a mathematical theorem? A statement about the physical world?
- Formulate the halting problem. Is it computable?

① Church - Turing Thesis

Anything that can be computed on any physical device can also be computed on a Turing Machine (or Java Program)

② Church - Turing thesis is not a mathematical theorem
It is a statement about the physical world

③ Halting Problem

Th. No algorithm is possible that given a problem p and data d, checks whether p halts on d.

No, it is not computable.