106/100

CS 3350 Automata, Computability, and Formal Languages
Spring 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: S --> BB, S --> BSb, S --> bA, A --> aB, A --> ε, B --> ε.

Starting state S

$S \to BB$
$S \to BSb$
$S \to bA$
$A \to aB$
~~$A \to \varepsilon$~~
~~$B \to \varepsilon$~~
$S \to \varepsilon$

STEP1

~~$S \to B$~~
$S \to Sb$
$A \to a$
$S \to b$
$S \to BB$
$S \to BSb$
$S \to bA$
$A \to aB$
$S \to \varepsilon$

STEP2

$U_b \to b$
$S \to SU_b$
$A \to a$
$S \to b$
$S \to U_bA$
$A \to U_aB$
$S \to BB$
$S \to A_1U_b$

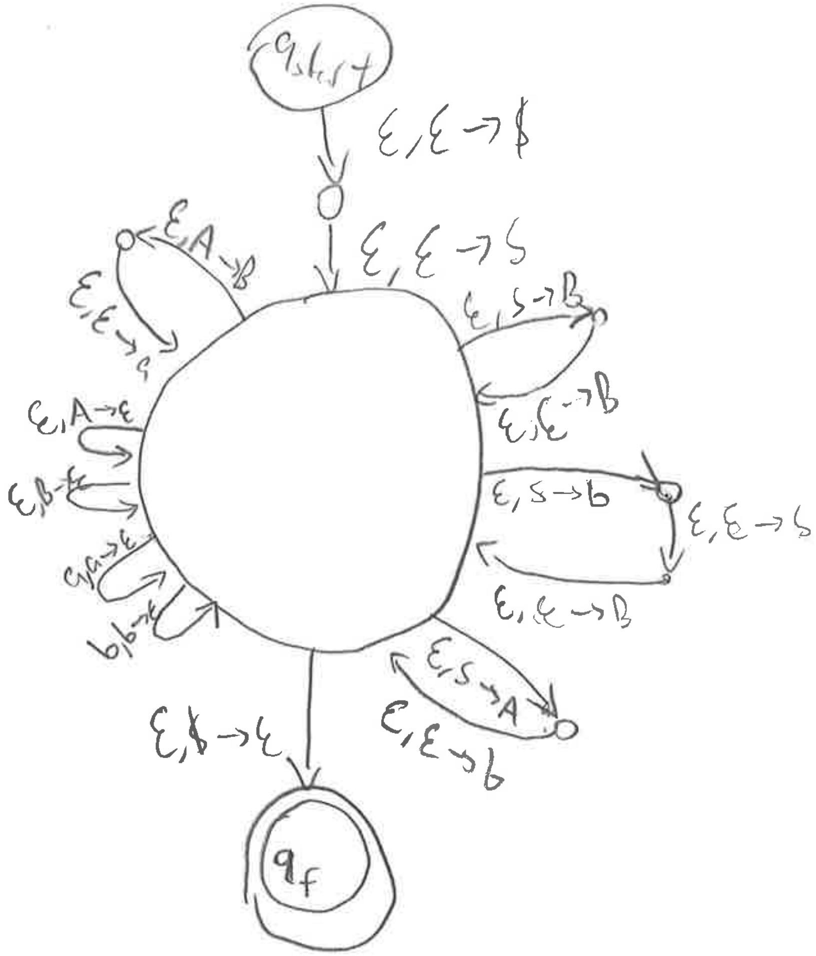
$U_a \to a$
$A_1 \to BS$
$S \to \varepsilon$

STEP3

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.

S → BB
S → BSb
S → bA
A → aB
A → ε
b → ε

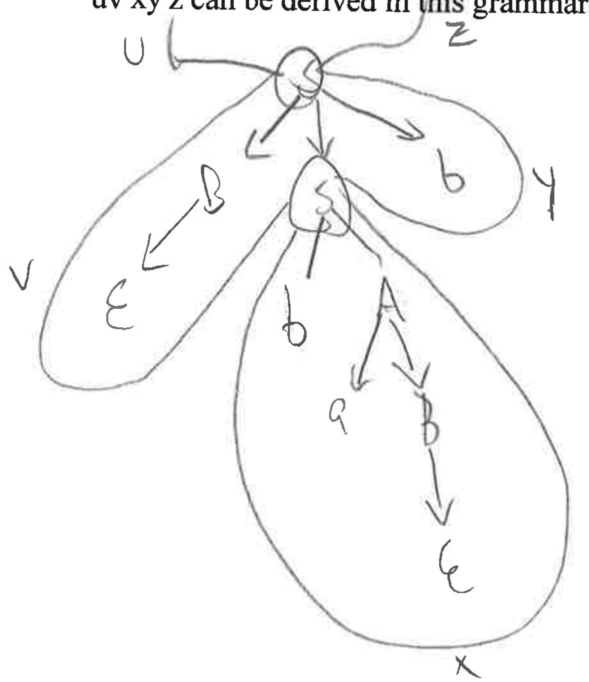


10/10

3. In the context-free grammar described in Problem 1, we can derive the sequence bab as follows:

- first, we use the rule S --> BSb;
- then, we use the rules B --> ε and S --> bA;
- after that, we use the rule A --> aB;
- finally, we use the rule B --> ε.

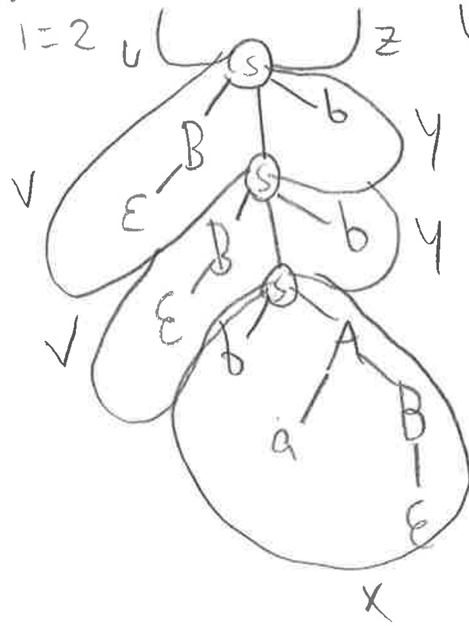
Draw a tree that describes this derivation. Use this tree to determine the subdivision of this sequence into u, v, x, y, and z. Then, for i = 0 and i = 2, draw trees explaining how the corresponding sequences uv^ixy^iz can be derived in this grammar.



$u = \varepsilon$
$v = \varepsilon$
$x = ba$
$y = b$
$z = \varepsilon$

$i = 0$ $uv^0xy^0z = ba$

$i = 2$ $uv^2xy^2z = babb$



10|10

4. Use pumping lemma for pushdown automata to prove that the language consisting of all the words of the type $a^{3n}b^{2n}c^{n}$ cannot be recognized by a pushdown automaton.

$L = \{a^{3n} b^{2n} c^{n} : n = 0, 1, 2, \ldots\}$

Prove L is not CFG i.e. cannot be recognize by PDA

Proof Assume L is a context free grammar then by pumping lemma there exists a p such that for $s \in L$ and $len(s) \geq p$ then there exist $uvxyz = s$ where $len(vy) > 0$, $len(vxy) \leq p$ and for $i = 0, 1, \ldots$ $uv^ixy^iz \in L$.
Let $s = a^{3p}b^{2p}c^{p}$, then by pumping lemma $uvxyz = s$ since $len(s) = 6p \geq p$
Now take uv^ixy^iz for $i = 2$ then $uv^ixy^iz \in L$ but

- if vxy is in a's and we pump only a's are added disrupting balance. The case is the same for vxy in b's or c's.
- if vxy is in a's and b's then only a's and b's will be added when we pump and balance will disrupted
- if vxy is in b's and c's then only b's and c's will be added when we pump and balance will be disrupted

Since $len(vxy) \leq p$ it cannot be in all three letter groups and since in all other cases, balance was disrupted uv^ixy^iz is not in L which is contradiction

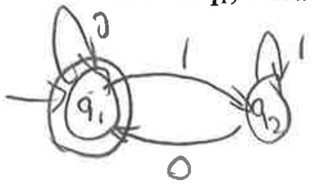
So L is not CFG//

10/10

5. Use the general algorithm to design a context-free grammar that is equivalent to the following finite automaton. This automaton has two states:

- a state q_1 which is both a start state and a final state, and
- a state q_2.

In the state q_1, 1 leads to q_2, and 0 leads to q_1. In the state q_2, 1 leads to q_2, and 0 leads to q_1.



$S \to Q_1$

$Q_1 \to 1 Q_2$

$Q_2 \to 1 Q_2$

$Q_1 \to 0 Q_1$

$Q_2 \to 0 Q_1$

$Q_1 \to \varepsilon$

10/10

6. Use the general algorithm to design a Turing machine that is equivalent to a finite automaton from Problem 5.

(start, #) → (Q1, R)

(Q1, 1) → (Q2, R)

(Q1, 0) → (Q1, R)

(Q2, 1) → (Q2, R)

(Q2, 0) → (Q1, R)

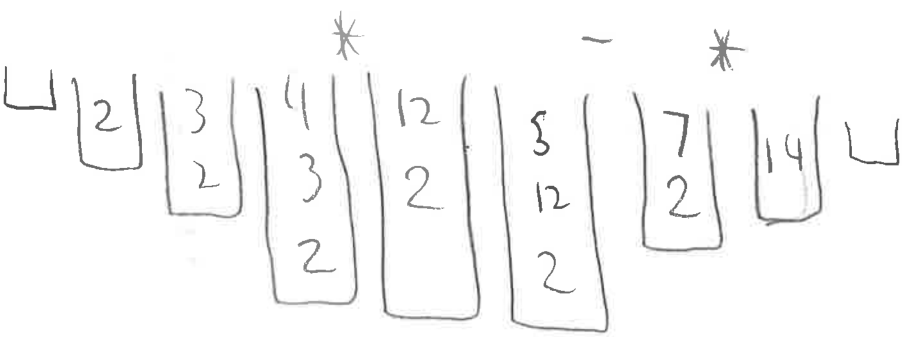
(Q2, #) → reject

(Q1, #) → accept

7. Use the general stack-based algorithms to show: 10/10

- 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 - *



20/20

8-9. Design a Turing machine that computes the function n + 2: 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.

(start, #) → (inNum, R)

(inNum, #) → (add1, 1, R)

(inNum, 1) → (inNum, R)

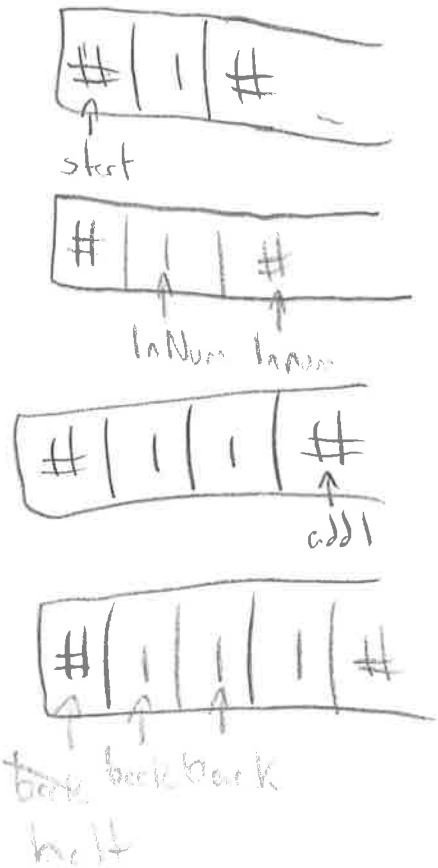
(add1, #) → (back, 1, L)

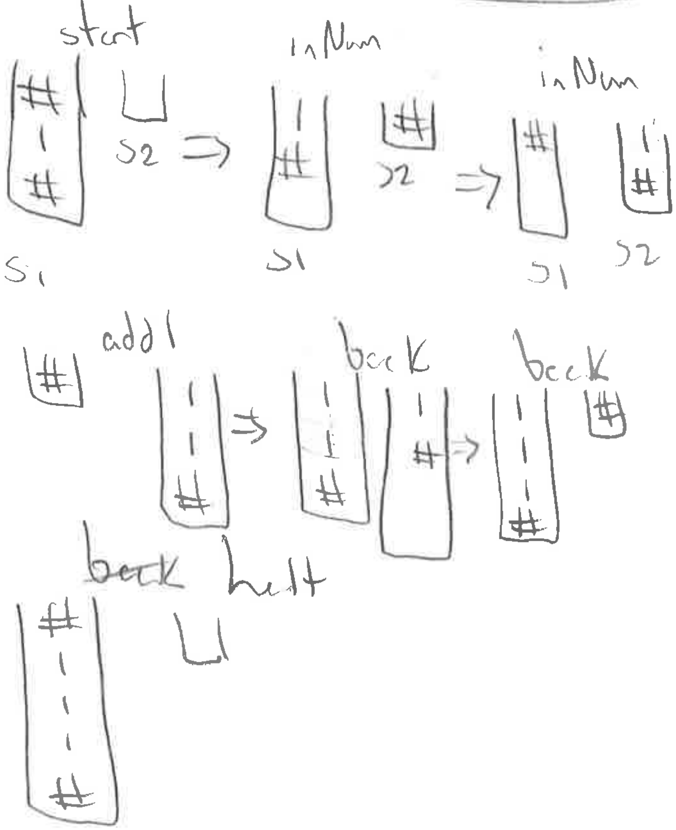
(back, 1) → (back, L)

(back, #) → halt

Unary

n = 1





10. Computability and feasibility:

$\frac{10+3}{10}$

- Formulate Church-Turing thesis.
- Is it a mathematical theorem? A statement about the physical world?
- Formulate the current definition of a feasible algorithm.
- *For extra credit:* Explain why this definition is not perfect.

Church-Turing thesis

Anything that is computable on a physical device can be computated on a turing machine

Is a statement about physical world

Feasible algorithm

An algorithm is feasible if there exist a polynomial $P(n)$ that $F_A(x) \leq p(len(x)$

Not a perfect definition because there are algorithms that are not feasible in practice such as $F_A(n) = 10^{301} n$ and there are algorithms that are not feasible by definition but are feasible in practice such as $F_A(n) = e^{10^{-101} n}$