1. Finite automata and regular languages:

1a. Design a finite automaton for recognizing binary sequences that have even number of 0s and even number of 1s. Assume that the input strings contain only symbols 0 and 1. The easiest is to have 4 states:

- the desired (final) state ee in which we have even number of 0s and even number of 1s,
- the state eo, in which we have even number of 0s and odd number of 1s, and
- similarly defined states oe (odd-even) and oo (odd-odd).

You just need to describe transitions between these states, and which states are final. Show, step-by-step, how your automaton will accept the string 1100.

1b. On the example of this automaton, show how the word 1100 can be represented as xyz in accordance with the pumping lemma.

1c. Use a general algorithm to describe a regular expression corresponding to the finite automaton from the Problem 1a. (If you are running out of time, it is Ok not to finish, just eliminate the first state.)

1d-e. Some of the words from the corresponding language (but not all of them) The resulting language can be described by a regular expression \((11)^*(00)^*\). Use a general algorithm to transform this regular expression into a finite automaton: first a non-deterministic one, then a deterministic one.
2. Beyond finite automata: pushdown automata and context-free grammars:

2a. For a student to be on the College of Engineering's Dean's list, his or her GPA must be 3.5 or higher. In particular, this is possible if the student has only As and Bs and at least as many As as Bs: so the grades sequences AB, AABB, ABAA, etc. make a student eligible. Prove that the language consisting of all sequences of letter grades that lead to GPA of at least 3.5 is not regular.

2b. Use a general algorithm to transform the finite automaton from the Problem 1a into a context-free grammar (CFG). Show, step-by-step, how this CFG will generate the word 1100.

2c. For the context-free grammar from the Problem 2b, show how the word 1100 can be represented as uvxyz in accordance with the pumping lemma.

2d. Use a general algorithm to translate the CFG from 2b into Chomsky normal form.

2e. Use a general algorithm to translate the CFG from 2b into an appropriate push-down automaton. Explain, step-by-step, how this automaton will accept the word 1100.

2f. Use the general stack-based algorithms to show:

- how the compiler will transform the GPA-computing expression expression \((2 + 3 + 4) / 3\) into inverse Polish notation, and
- how it will compute the value of this expression.

2a. Theory \(L\) is not regular. Proof by contradiction:

Let's assume \(L\) is regular, then by pumping lemma, there exists \(p \in \mathbb{N}\) such that for all \(w \in L\) with \(|w| \geq p\), there exist \(x, y, z\) such that:

\[
|y| \geq 1, \quad |xy^iz| \in L \quad \forall i \in \mathbb{N}.
\]

Let's take a word \(w = B^p A^p \in L\). Then, since \(w = B^p A^p \in L\), let's consider the first \(p\) symbols in \(w\), \(y\) can only consist of \(B\)'s. By pumping lemma, \(xyyz\) should be in \(L\), but when we added an extra \(y\), which is some \(\#\) of \(B\)'s, we created a word with more \(B\)'s than \(A\)'s, which is not in the language \(L\). So \(L\) must not be regular.
2b. let's say...

- E is ee (both even)
- D is oo (both odd)
- Z is eo (zero is odd)
- N is oe (one is odd)

E → ON
E → Z
E → O
D → IN
D → OZ
Z → IE
Z → OD
N → OE
N → IZ

For 1100

2c.

- U = 11
- V = 00
- X = E
- Y = E
- Z = E

We pumping lemma to get U VWXYZZ
2a. Chomsky Normal Form

preliminary step

\[
\begin{align*}
E & \rightarrow \alpha N \beta \\
E & \rightarrow \alpha X \beta \\
E & \rightarrow \alpha I \beta \\
D & \rightarrow \alpha N \beta \\
D & \rightarrow \alpha Z \beta \\
Z & \rightarrow \alpha E \beta \\
Z & \rightarrow \alpha O \beta \\
N & \rightarrow \alpha E \beta \\
N & \rightarrow \alpha I \beta \\
S & \rightarrow \alpha E \\
E & \rightarrow V_0 \beta \\
E & \rightarrow V_1 \beta \\
D & \rightarrow V_0 \beta \\
D & \rightarrow V_1 \beta \\
Z & \rightarrow V_0 \beta \\
Z & \rightarrow V_1 \beta \\
V_0 & \rightarrow \alpha \\
V_0 & \rightarrow \beta \\
V_1 & \rightarrow \alpha \\
V_1 & \rightarrow \beta \\
\end{align*}
\]
For 1100

S 1 W 2 W W 5 W W 1 W W 7

word accepted
\[ \frac{(2 + 3 + 4)}{3} = \frac{2 + 3 + 4}{3} = 3 \]
3. Beyond pushdown automata: Turing machines

3a. In Houston, one third of the population is African-American (A), one third is Hispanic (H), and one third is White Non-Hispanic (W). To get a good representation, the city tries to make sure that every committee have equal number of representatives of each of the three groups. So, if a committee has WWAH it is perfectly balanced, but if it has WWHH, it is not. Prove that the set of all balanced committees is not context-free and therefore, cannot be recognized by a pushdown automaton.

3b-c. Use a general algorithm to design a Turing machine that accepts exactly all sequences accepted by a finite automaton from Problem 1a. Show, step-by-step, how this Turing machine will accept the word 1100. Describe, for each step, how the state of the tape can be represented in terms of states of two stacks.

3d-e. Design Turing machines for computing $a + 1$ in unary and in binary codes. Trace both for $a = 3$, i.e., for $a = 111$ in unary code and $a = 11$ in binary code.

3a. Let's assume that $L$ is context free, then by pumping lemma $A(G, E)$, $w, e \in L \Rightarrow uv_i x y_i z (w = u x y z \& \text{len}(vy) > 0 \& \text{len}(vxy) \leq p \& V_i (uw_i x y_i z))$.

Let's take a word $w = A^p H^p W^p$ so that its length is $3p^2 + p$, $w \in L$. We can represent it as $uvwxyz = A \ldots A H \ldots H W \ldots W$.

The fragment $vxy$ can't contain $A$'s and $H$'s and $W$'s because its length would be greater than $p$.

We have 5 options:

1) $vxy$ is in $A$'s
2) $vxy$ is in $A$'s and $H$'s
3) $vxy$ is in $H$'s
4) $vxy$ is in $H$'s and $W$'s
5) $vxy$ is in $W$'s
In case 1: We add A's but not H's or W's.
So $\forall x y^2 z$, we have more A's than H's or W's.
So $uv^2xy^2z \notin L$.

In case 2: We add A's and H's but not W's.
So $\forall x y^2 z$, we have more A's and H's than W's.
So $uv^2xy^2z \notin L$.

Very similar in case 3, 4, and 5...
But by pumping lemma $uv^2xy^2z \notin L$. So we have a contradiction and our assumption is false.

$L$ is not a CFG.

$$
\begin{array}{ccc}
E & & \text{for } ee \\
D & & \text{for } aa \\
Z & & \text{for } ea \\
N & & \text{for } oe \\
\end{array}
$$

$\text{start},_1 \rightarrow E, R$

$E, 1 \rightarrow Z, R$
$E, 0 \rightarrow N, R$
$E, \_ \rightarrow \text{accept}$

$D, 1 \rightarrow N, R$
$D, 0 \rightarrow Z, R$
$D, \_ \rightarrow \text{reject}$

$Z, 1 \rightarrow E, R$
$Z, 0 \rightarrow D, R$
$Z, \_ \rightarrow \text{reject}$

$N, 1 \rightarrow D, R$
$N, 0 \rightarrow Z, R$
$N, \_ \rightarrow \text{reject}$
3c.

```
start
push
pop
push
pop
push
pop
```

3d.

```
start, _ → mNum, R
mNum, 1 → R
mNum, _ → 1, back, L
back, 1 → 2
back, _ → halt
```

file:///C:/local.files/cs3350.18/final.html
3e. start, → moving, R
    moving, 1 → R
    moving, 0 → 1, back, L
    moving, → 1, back, L
    back, 1 → 0, L
    back, → halt

```
start

111

moving

111

moving

111

back
```
3. Beyond pushdown automata: Turing machines

3a. In Houston, one third of the population is African-American (A), one third is Hispanic (H), and one third is White Non-Hispanic (W). To get a good representation, the city tries to make sure that every committee has equal number of representatives of each of the three groups. So, if a committee has WWAHAW it is perfectly balanced, but if it has WWAHAH, it is not. Prove that the set of all balanced committees is not context-free and therefore, cannot be recognized by a pushdown automaton.

3b-c. Use a general algorithm to design a Turing machine that accepts exactly all sequences accepted by a finite automaton from Problem 1a. Show, step-by-step, how this Turing machine will accept the word 1100. Describe, for each step, how the state of the tape can be represented in terms of states of two stacks.

3d-e. Design Turing machines for computing a + 1 in unary and in binary codes. Trace both for a = 3, i.e., for a = 111 in unary code and a = 11 in binary code.

39 \[ \frac{1}{3} A \quad \frac{1}{3} H \quad \frac{1}{3} W \]

L = \( \{A^nH^nW^n\} \)

Proof by contradiction:

Let's assume L is CFG. Then by pumping lemma, there exists p such that any word W \( \in L \) (whose length \( \text{len}(w) \geq p \) can be represented as \( uvxy \), so that \( \text{len}(uvxy) \leq p \) for every \( u, uvxy \in L \)).

Let us take \( w = A^pH^pW^p = A_{\ldots}A_{\ldots}H_{\ldots}H_{\ldots}W_{\ldots}W_{\ldots} \) \( \text{len}(w) = 3p \geq p \), so we can represent it as \( uvxy \), such that \( u, v, x, y \in \Sigma^* \) and \( v \neq \epsilon \). Then \( \text{len}(w) = 3p \geq p \), so we can represent it as \( uvxy \).

So we have the following options:
1. \( vxy \) is in A,
2. \( vxy \) is in H's and A's,
3. \( vxy \) is in H's
4. \( vxy \) is in H's and W's
5. \( vxy \) is in W's.

When we consider \( uvvxwy \) in Case 1, we add A's but not H's or W's, so in \( uv^2xwy \) we have more A's than H's and W's, so \( uv^2xwy \notin L \).

In Case 2, we add A's and H's but not W's, so \( uv^2xwy \notin L \), because we have more A's and H's than W's.

In Case 3, we add H's but not A's or W's, so \( uv^2xwy \notin L \), because we have more H's than A's and W's.
In case 4 we add $H$'s and $W$'s but not $A$'s, so $uv^2xy^2z \notin L$ because we have more $H$'s and $W$'s than $A$'s.

In case 5, we add $W$'s but not $A$'s or $H$'s, so $uv^2xy^2z \notin L$ because we have more $W$'s than $H$'s and $A$'s.

But by pumping lemma $uv^2xy^2z \in L$. We have a contradiction. So our assumption is false, therefore $L$ is not CFG.
3b-c FA-7M

Word: 1100

2 stacks

pop from right stack, push into left stack

Accepted
3d-e. TM for computing \( a + 1 \) in unary is binary \( a = 3 \) in unary \( a = 11 \) in binary

**Unary**

\[
\begin{align*}
&\text{Start} \\
&\text{in} 1, 1 \rightarrow R \\
&\text{in} 1, - \rightarrow \text{in} 2, R \\
&\text{in} 2, 1 \rightarrow R \\
&\text{in} 2, - \rightarrow \text{Read, D, L} \\
&\text{Read, D, 1} \rightarrow \text{inmn, B} \\
&\text{inmn, B, 1} \rightarrow \text{L} \\
&\text{inmn, B, -} \rightarrow \text{inmn, B, L, L} \\
&\text{inmn, B, 1} \rightarrow \text{L} \\
&\text{inmn, B, -} \rightarrow \text{Halt}
\end{align*}
\]
Binary

\[ a = 11 = 3 \]

\[ \begin{align*}
A, 1, 0 & \rightarrow R \\
B, -, 1 & \rightarrow C, L \\
C, 0 & \rightarrow L \\
C, - & \rightarrow C
\end{align*} \]

Start:

\[ \begin{align*}
& \quad T \\
& \quad A
\end{align*} \]

100₂ = 4

halt
3. Beyond pushdown automata: Turing machines

3a. In Houston, one third of the population is African-American (A), one third is Hispanic (H), and one third is White Non-Hispanic (W). To get a good representation, the city tries to make sure that every committee have equal number of representatives of each of the three groups. So, if a committee has WWAHAH it is perfectly balanced, but if it has WWHAH, it is not. Prove that the set of all balanced committees is not context-free and therefore, cannot be recognized by a pushdown automaton.

3b-c. Use a general algorithm to design a Turing machine that accepts exactly all sequences accepted by a finite automaton from Problem 1a. Show, step-by-step, how this Turing machine will accept the word 1100. Describe, for each step, how the state of the tape can be represented in terms of states of two stacks.

3d-e. Design Turing machines for computing $a + 1$ in unary and in binary codes. Trace both for $a = 3$, i.e., for $a = 111$ in unary code and $a = 11$ in binary code.

3a. $L = \{ ^w_1H^w_1A^w_1 \}$

Assume $L$ is context free. Then by pumping lemma, $\exists p \forall w \in L$ : $\text{length}(w) \geq p$ and $\exists u v x y z \in L$ : $w = u v x y z$ and $\text{length}(vxy) > 0$ and $\text{length}(wxy) \geq p$ and $\forall i \in \mathbb{N}$ $(uvxy)_i \in L$.

Let's take $w = \text{WHHA}$. 3th length $(w) = 3p > p$ so we can represent it as $uvxyz$.

- We have several cases vxy can be:
  1. vxy contains only W, we add W but not H or A making them unequal
  2. vxy contains only H, "H "W or A ""
  3. vxy contains only A "A "W or H ""
  4. vxy contains W and H, we add W and H but not A making them unequal
  5. "W and A "W and A "H"
  6. "H and A "H and A "W"
  7. "W, H and A , since $\forall i \in \mathbb{N}$ $(uvxy)_i \in L$ we would add unequal amounts, making them unequal.

Since no case is possible, we have a contradiction, thus our assumption that the language was context free was false.
3b

\[
\begin{align*}
&\text{start: } \_, \_ \rightarrow D, R \\
&\text{D, 1} \rightarrow B, R \\
&\text{Q, 0} \rightarrow C, R \\
&\text{B, 1} \rightarrow Q, R \\
&\text{B, 0} \rightarrow A, R \\
&\text{C, 1} \rightarrow A, R \\
&\text{C, 0} \rightarrow D, R \\
&\text{A, 1} \rightarrow C, R \\
&\text{A, 0} \rightarrow B, R \\
&\text{D, } \_ \rightarrow \text{accept} \\
&\text{C, } \_ \rightarrow \text{reject} \\
&\text{B, } \_ \rightarrow \text{reject} \\
&\text{A, } \_ \rightarrow \text{reject}
\end{align*}
\]

3c

\[
\begin{align*}
&\text{push pop} \\
&\text{start} \\
&\text{D} \\
&\text{push pop} \\
&\text{B} \\
&\text{push pop} \\
&\text{C} \\
&\text{push pop} \\
&\text{D accept}
\end{align*}
\]

3d

\[
\begin{align*}
&\text{start: } \_, \_ \rightarrow \text{read}, R \\
&\text{read, 1} \rightarrow \text{read}, R \\
&\text{read, } \_ \rightarrow \text{add}, 1, L \\
&\text{add, 1} \rightarrow \text{back}, L \\
&\text{back, 1} \rightarrow \text{back}, L \\
&\text{back, } \_ \rightarrow \text{halt}
\end{align*}
\]

3e

\[
\begin{align*}
&\text{start: } \_, \_ \rightarrow \text{read}, R \\
&\text{read, 1} \rightarrow \text{read}, 0, R \\
&\text{read, 0} \rightarrow \text{back}, 1, L \\
&\text{read, } \_ \rightarrow \text{back}, 1, L \\
&\text{back, 1} \rightarrow \text{back}, L \\
&\text{back, 0} \rightarrow \text{back}, L \\
&\text{back, } \_ \rightarrow \text{halt}
\end{align*}
\]
4. Beyond Turing machines: computability

4a. Formulate Church-Turing thesis. Is it a mathematical theorem? Is it a statement about the physical world?

✓ 4b. Prove that the halting problem is not algorithmically solvable.

✓ 4c. Not all algorithms are feasible, but, unfortunately, we do not have a perfect definition is feasibility. Give a current formal definition of feasibility and give two examples:

- an example of an algorithm's running time which is feasible according to the current definition but not practically feasible, and
- an example of an algorithm's running time which is practically feasible but not feasible according to the current definition.

✓ 4d. Briefly describe what is P, what is NP, what is NP-hard, and what is NP-complete. Is P equal to NP?

✓ 4e. Give an example of an NP-complete problem: what is given, and what we want to find.

✓ 4f. Give definitions of a recursive (decidable) language and of a recursively enumerable (Turing-recognizable) language.

4a. Church-Turing Thesis: Anything that can be computed on any physical device can also be computed on a Turing Machine. It is not a mathematical theorem. It is a statement about the physical world.
4.6 **Halting Problem:** In a computer, a program $p$ of data $d$ is a sequence of 0's and 1's.

\[
\begin{align*}
01_2 \\ 001_2 \\ 110_2 \\ 101_2
\end{align*}
\]

these are different strings but the computer sees the same string.

-To avoid ambiguity, we can append 1 in front

\[
\begin{align*}
11_2 \\ 101_2
\end{align*}
\]

now they are different.

**Definition of Halting Problem:**

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

**Proof by contradiction:**

Let's assume there exists a halt checker, the number corresponding to this program is $P_0$.

Does $P_0$ halt on $P_0$?

- **Case 1:** $P_0$ halts on $P_0$, then $\text{halts}(P_0, P_0)$ is true, so $P_0$ won't halt on $P_0$.

- **Case 2:** $P_0$ doesn't halt on $P_0$, then $\text{halts}(P_0, P_0)$ is false, so $P_0$ halts on $P_0$.

So, we have a contradiction in both cases, and our assumption is false.
4C. Feasibility: \( \text{feasibility} = \text{poly time} \).

Definition: An algorithm \( A \) is called feasible if there exists a polynomial \( P \) such that for every input \( x \) of length \( n \), \( t_A(x) \leq P(n) \). (Time needed to solve this polynomial is less than \( P(n) \)).

- Example of feasible according to the definition,
  but not practically feasible:

  \[
  \begin{align*}
  10^{300} & \cdot n \\
  n^{1000}
  \end{align*}
  \]

- Example of not feasible according to the definition,
  but practically feasible:

  \[
  \begin{align*}
  10^{(-100 \cdot n)} \\
  e^{(10^{-1000} \cdot n)}
  \end{align*}
  \]
4d. _P_-class of languages decidable in polynomial time on a deterministic single tape Turing Machine.

_NP_- class of languages that have polynomial time verifiers.

_NP_-hard - A problem x is _NP_-hard if there is an _NP_-complete problem y such that y is reducible to x in polynomial time.

Ex: SATs

_NP_-complete - A language B is _NP_-complete if it satisfies 2 conditions:

1) B is in _NP_.

2) Every A in _NP_ is polynomial time reducible to B.

4e. Ex: Hamiltonian path - whether input graphs contain a path s to t that goes through every node exactly once.

_P_ ≠ _NP_ It is not known whether _P_ = _NP_, it is still an unsolved question.
4f. Recursively Decidable.

A set \( A \) is decidable if there is an algorithm that given a natural number \( n \), decides whether \( n \in A \).

\[
\begin{align*}
\text{public static boolean } & \text{ is}(\text{int} \ n) \\
& \{ \\
& \text{return false;} \\
& \} \\
& \{ \\
& \text{return true;} \\
& \}
\end{align*}
\]

\( 0 \) if \( n \in A \\
1 \) if \( n \notin A \)

\text{Th. 1} \ \emptyset \text{ is decidable}

\text{Th. 2} \ \mathbb{N} \text{ is decidable}

\text{Th. 3} \ \text{Every finite set is decidable}

\text{Th. 4} \ \text{If } A, B \text{ are decidable then } A \cup B \text{ is decidable.}

\text{Th. 5} \ \text{If } A \cap B \text{ are decidable then } A \cap B \text{ is decidable}

\text{Th. 6} \ \text{If } A \text{ is decidable then } -A \text{ is decidable.}

\text{v.e.}
Recursively enumerable (r.e) or Turing-recognizable (t.r)

\[ T \vdash \text{There is a Turing Machine that accepts all elements from this set and only them} \]

Set is r.e. \( \iff \) there is an algorithm that eventually prints all elements of this set and only them.

\[ \square \rightarrow \text{prints all } n \in A. \]

**Th 1** \( \emptyset \) is r.e.

**Th 2** \( \mathbb{N} \) is r.e.

**Th 3** Every finite set is r.e.

**Th 4** Every decidable set is r.e.

**Th 5** If \( A, B \) are r.e. then \( A \cup B \) is r.e.

**Th 6** If \( A, B \) are r.e. then \( A \cap B \) is r.e.

**Th 7** If \( A \) is r.e. and \( -A \) is r.e., then \( A \) is decidable.

**Th 8** The halting set \( H \) is not decidable but r.e.

\( H \) is r.e. Take all \( p \leq 5, d \leq 1 \) run all of them 4 times each.

\[ \begin{align*}
  p &\leq 5 \\
  d &\leq 1 \\
  &\text{if any of them halts, print } (p,d) \\
\end{align*} \]

- Take all \( p \leq 2, d \leq 2 \) run all of them 2 times each.
- Take all \( p \leq 3, d \leq 3 \) run all of them 3 times each.