Name: [redacted]
Student ID: [redacted]

CS181 Winter 2017 - Final

Due Friday, March 17, 11:55 PM

- This exam is open-book and open-notes, but any materials not used in this course are prohibited, including any material found on the internet. **Collaboration is prohibited.** Please avoid temptation by not working on the final while you are in the presence of any other student who has taken or is currently taking CS181. **Be extra careful if you live with or meet regularly with a student of this class.** If you have any questions about the exam, ask the TA or Professor Sahai by email or after class. **Do not ask other students.** You are allowed to use any theorem shown in class or in the textbook, as long as you clearly cite it. Please monitor Piazza for any clarifications. **Do not** post any questions on piazza.
- We suggest that you spend approximately 12 hours (not necessarily contiguous) to take this exam. Start early so that you have time to understand and think about the problems. **The solutions must be submitted on CCLE by 7 PM on Friday, March 17.**
- Place your name and UID on every page of your solutions. Retain this cover sheet and the next sheet with the table as the first pages of your solutions. **Please use separate pages for each question. All problems require clear and well-written explanations.**
- There are 5 questions and a total of 230 points and an extra credit question worth 50 points.
- For each part (except for the extra credit), if you describe a non-trivial approach that you tried using to solve the problem but realized it doesn't work and explain correctly why it doesn't work and then write "I don't know" you will get 20% points for that problem. You will **not** get 20% points for just writing "I don't know". Whether your stated approach was indeed non-trivial is solely at the discretion of the grader.

Please **handwrite** the following honor code agreement and sign and date in the spaces provided.
Honor Code Agreement: I promise and pledge my honor that during the exam period, I did not and will not talk to any person about CS 181 material except for the professor or the TA, nor will I refer to any material except for the class textbook and my own class notes. I will abide by the CS181 Honor Code.

Signature: _____

Date: _____

Question	**Points**	
1		
2		
3		
4		
5		
EC		
Total		

1. **Pushdown Automata (20 points)**

   (a) **(20 points)** Let L_1 and L_2 be languages. We define the operation

   $$L_1 \diamond L_2 = \{xy \mid x \in L_1, y \in L_2, \text{ and } |x| = 2\,|y|\}.$$

   Prove that the set of context-free languages is not closed under the $\diamond$-operation. I.e. find two context-free languages L_1 and L_2 such that $L_1 \diamond L_2$ is not context-free.

Solution:
Let $L_1 = \{0^n 1^n \mid n \geq 1\}$
and let $L_2 = \{2^n \mid n \geq 1\}$

Now, $L_1 \diamond L_2 = \{0^n 1^n 2^n \mid n \geq 1\}$

Now, it was shown in class that L_1 is context-free. Furthermore, L_2 is trivially context free since it can be trivially simulated using a DFA and all regular languages are context free.

Finally, as shown in class, the langauge $L_1 \diamond L_2 = \{0^n 1^n 2^n \mid n \geq 1\}$ is not context free. Hence, the set of context-free languages is not closed under $\diamond$ operation.

2. **Oracle Building Turing Machines (45 points)**
In this problem, we define a Turing machine variant called an Oracle Building Turing Machine (OBTM). An Oracle Building Turing Machine has a total of three tapes: in addition to its regular tape, it has a *machine tape* and an *input tape*. The OBTM can write to the three tapes in the ordinary way, but it has an extra trick up its sleeve. At any given moment, the OBTM can go into a special *query state* and perform an *oracle query*. Suppose an ordinary Turing Machine representation $\langle M \rangle$ is stored on the machine tape of an OBTM T and the input tape of T contains the string w. If T enters the query state, then immediately T learns whether $w \in L(M)$ or $w \notin L(M)$ (even if M does not halt on w, T will get to know that $w \notin L(M)$). In particular, if $w \in L(M)$, then the first cell of the input tape is set to 1, and if $w \notin L(M)$ then the first cell of the input tape is set to 0. We say that a language $L \subseteq \Sigma^*$ is oracle recognizable (respectively, oracle decidable) if there exists an oracle building Turing machine T recognizing (respectively, deciding) L.
Note: Throughout this problem, $\langle M \rangle$ denotes a description of an *ordinary* Turing machine.
Example: An oracle building Turing machine can decide the language $\mathsf{TM} = \{(\langle M \rangle, x) \mid M(x) \text{ accepts}\}$. Specify the OBTM O as follows. On input $(\langle M \rangle, x)$, O writes $\langle M \rangle$ to the machine tape and x to the input tape. Then O enters the query state and retrieves whether $x \in L(M)$ or $x \notin L(M)$. In the first case, O accepts and in the second case, O rejects. Since the query is immediate, O always terminates and thus is an oracle decider for TM.

(a) **(10 points).** Show that the language

$$\mathsf{HALT} = \{(\langle M \rangle, x) \mid M \text{ halts on input } x\}$$

is oracle decidable.

(b) **(15 points).** Show that the language

$$\mathsf{NEQ} = \{(\langle M_1 \rangle, \langle M_2 \rangle) \mid L(M_1) \neq L(M_2)\}$$

is oracle recognizable.

(c) **(20 points).** Show that there exists a language L which is not oracle recognizable.
Hint: Think about infinities.

Solution:
(a) Specify the OBTM O as follows:
On input $(\langle M \rangle, x)$, O writes $< M >$ to the machine tape and x to the input tape. Then O enters the query state and retrieves whether $x \in L(M)$ or $x \notin L(M)$. This result (available on the first cell of the input tape) is stored on O's main tape. Next, O interchanges the accepting states and rejecting states in $< M >$ to give $< M' >$. Note that M' rejects when M accepts and M' accepts when M rejects. Next, O places $< M' >$ on the machine tape and x on the input tape. Then O enters the query state and retrieves whether $x \in L(M')$ or $x \notin L(M')$. This result is again stored. Finally, O accepts if the results of the two query states are different (i.e. 01 or 10), and rejects if they are the same (i.e. 00).

Analysis:
When the query state of O returns a 1, $x \in L(M)$ and M accepts (halts). When the query state of O returns a 0, $x \notin L(M)$ and either M rejects (halts) or loops. Consider the following cases:
1. If O returns 1 on querying $< M >$, querying on $< M' >$ cannot return a 1 since when M accepts, M' rejects and vice-versa.
2. If O returns 1 on querying $< M >$, and 0 on querying $< M' >$. M halts on x since an accepting state is reached by M.
3. If O returns 0 on querying $< M >$, and 1 on querying $< M' >$. M halts on x since a rejecting state is reached by M since M' reached an accepting state.
4. If O returns 0 on querying $< M >$, and 0 on querying $< M' >$. M loops on x since neither machine reached an accepting state and both of them cannot simultaneously be in a rejecting state for the same

input x. Thus, both of them must loop on input x

(b) Specify the OBTM O as follows:

On input $(\langle M_1 \rangle, \langle M_2 \rangle)$, O starts an enumerator E that generates all strings in Σ^*. For each string x generated by the E, O writes $< M_1 >$ to the machine tape and x to the input tape. Then O enters the query state and retrieves whether $x \in L(M_1)$ or $x \notin L(M_1)$. This result (available on the first cell of the input tape) is stored on O's main tape. Then, O writes $< M_2 >$ to the machine tape and x to the input tape and O enters the query state and retrieves whether $x \in L(M_2)$ or $x \notin L(M_2)$. This result is again stored. Finally, O accepts if the results of the two query states are different (i.e. 01 or 10), and continues otherwise.

Analysis:
If $L(M_1) \neq L(M_2)$, there must be some x generated by the enumerator E so that x is accepted by only one of the machines (and rejected or looped on by the other). This corresponds to the case when the query results are different (i.e. 10 or 01). In such a case, the machine accepts. Since, O will eventually accept (and halt) if $(< M_1 >, < M_2 >) \in NEQ$, and loop otherwise, NEQ is oracle-recognizable.

(c) Consider the language $EQ = \{(\langle M_1 \rangle, \langle M_2 \rangle) \mid L(M_1) = L(M_2)\}$. Now, intuitively to recognize this language, the Oracle Turing Machine must check $\forall x \in \Sigma^*$ and ensure that $\forall x \in L(M_1), x \in L(M_2)$ and also $\forall x \notin L(M_1), x \notin L(M_2)$. Now, this problem will never accept since the machine will run forever going through all the strings in Σ^*.

Proof by contradiction:
Assume for contradiction that EQ is oracle recognizable and $\exists R$ which is a recognizer for the EQ.

Build M(y):

Let $x =< M >$
ACCEPT if $y = 0^n1^n$
Let N be a TM s.t. $L(N) = \{0^n1^n \mid n \geq 1\}$
Run $R(x, < N >)$:
ACCEPT:
ACCEPT $y = 0$
REJECT:
REJECT all y
LOOP:

Analysis:
If R ACCEPTS, this implies that R believes that $L(M) = L(N)$. However, $L(M) = 0 \cup \{0^n1^n \mid n \geq 1\}$ and $L(N) = \{0^n1^n \mid n \geq 1\}$. This is a contradiction.
If R REJECTS, this implies that R believes that $L(M) \neq L(N)$. However, $L(M) = \{0^n1^n \mid n \geq 1\}$ and $L(N) = \{0^n1^n \mid n \geq 1\}$. This is a contradiction.
If R LOOPS, this implies that R believes that $L(M) \neq L(N)$. However, $L(M) = \{0^n1^n \mid n \geq 1\}$ and $L(N) = \{0^n1^n \mid n \geq 1\}$. This is a contradiction.
.

Hence, a contradiction is reached in each case. Thus, EQ is not oracle-recognizable.

3. **Undecidability and Unrecognizability (65 points)** Throughout this problem, we let $\Sigma = \{0, 1\}$ and define a distance between words as follows. If $x, y \in \Sigma^*$ are words of equal length, $|x| = |y|$, then we define the *distance* between x and y, written $||x - y||$, to be the number of bits of x and y that are different. Define the language

$$\mathsf{CLOSEBY} = \{(\langle M_1 \rangle, \langle M_2 \rangle) \mid \forall x \in L(M_1)\, \exists y \in L(M_2) \colon\ ||x - y|| \leq 1\}\,.$$

(a) **(15 points).** Show that the language CLOSEBY is not decidable.

(b) **(20 points).** Show that the language CLOSEBY is not Turing recognizable.

(c) **(30 points).** Show that the language

$$\mathsf{LEQ\text{-}HALT} = \left\{(\langle M \rangle, \langle N \rangle) \mid \forall x \in \Sigma^* : \begin{smallmatrix} M(x) \text{ and } N(x) \text{ halt and} \\ M(x) \text{ halts in fewer steps than } N(x) \end{smallmatrix}\right\}$$

is unrecognizable. In solving this problem you may assume for simplicity that any Turing machine can obtain its own description in 0 steps and that any loading of data into memory (i.e. loading constant values of the Turing Machine's pseudocode) also takes 0 steps.

Solution:
(a) Assume for contradiction that $CLOSEBY$ is decidable and $\exists D$, a decider for $CLOSEBY$.
Build M(y):

Let $x =< M >$
ACCEPT if $y = 0^n \forall n \geq 1$
Let N be a TM s.t. $L(N) = \{0^n \mid n \geq 1\}$
Run $D(x, < N >)$:
    ACCEPT:
        ACCEPT $y = 11$
    REJECT:
        REJECT all y

Analysis:
1. If D ACCEPTS, this implies that D believes that $(< M >, < N >) \in CLOSEBY$. However, $L(M) = \{11\} \cup \{0^n \mid n \geq 1\}$ and $L(N) = \{0^n \mid n \geq 1\}$. Thus, for the string 11 in $L(M)$, the only string of equal length in $L(N)$ is 00. Clearly, $||11 - 00|| = 2$. Thus, $(< M >, < N >) \notin CLOSEBY$. This is a contradiction.
2. If D REJECTS, this implies that D believes that $(< M >, < N >) \notin CLOSEBY$. However, $L(M) = \{0^n \mid n \geq 1\}$ and $L(N) = \{0^n \mid n \geq 1\}$. Thus, $\forall x \in L(M) \exists y \in L(N)$ s.t. $y = x$. Thus, $||x - y|| = 0$. Thus, $(< M >, < N >) \in CLOSEBY$. This is a contradiction.

Hence, a contradiction is reached in each case. Thus, $CLOSEBY$ is not decidable.

(b) Assume for contradiction that $CLOSEBY$ is recognizable and $\exists R$, a recognizer for $CLOSEBY$.
Build M(y):

Let $x =< M >$
ACCEPT if $y = 0^n \forall n \geq 1$
Let N be a TM s.t. $L(N) = \{0^n \mid n \geq 1\}$
Run $D(x, < N >)$:
    ACCEPT:
        ACCEPT $y = 11$
    REJECT:
        REJECT all y
    LOOP:

Analysis:
1. If R ACCEPTS, this implies that R believes that $(< M >, < N >) \in CLOSEBY$. However, $L(M) = \{11\} \cup \{0^n \mid n \geq 1\}$ and $L(N) = \{0^n \mid n \geq 1\}$. Thus, for the string 11 in $L(M)$, the only string of equal length in $L(N)$ is 00. Clearly, $||11 - 00|| = 2$. Thus, $(< M >, < N >) \notin CLOSEBY$. This is a contradiction.
2. If R REJECTS, this implies that R believes that $(< M >, < N >) \notin CLOSEBY$. However, $L(M) = \{0^n \mid n \geq 1\}$ and $L(N) = \{0^n \mid n \geq 1\}$. Thus, $\forall x \in L(M) \exists y \in L(N)$ s.t. $y = x$. Thus, $||x - y|| = 0$. Thus, $(< M >, < N >) \in CLOSEBY$. This is a contradiction.
3. If R LOOPS, this implies that R believes that $(< M >, < N >) \notin CLOSEBY$. However, $L(M) = \{0^n \mid n \geq 1\}$ and $L(N) = \{0^n \mid n \geq 1\}$. Thus, $\forall x \in L(M) \exists y \in L(N)$ s.t. $y = x$ and $||x - y|| = 0$. Thus, $(< M >, < N >) \in CLOSEBY$. This is a contradiction.

Hence, a contradiction is reached in each case. Thus, $CLOSEBY$ is not recognizable.

(c) Assume for contradiction that $LEQ - HALT$ is recognizable and $\exists R$, a recognizer for $LEQ - HALT$. Build M(y):

Let $x =< M >$
Let $N(z)$ be a TM that counts upto $100|z|$ before accepting
Run $R(x, < N >)$ for $|y|steps$:
R ACCEPTS:
LOOP
R REJECTS:
ACCEPT all y
R did not halt:
ACCEPT all y

Analysis:
1. If R ACCEPTS, this implies that R believes that $(< M >, < N >) \in LEQ - HALT$. However, M LOOPS by design. Thus, $(< M >, < N >) \notin LEQ - HALT$. This is a contradiction.
2. If R REJECTS, this implies that R believes that $(< M >, < N >) \notin LEQ - HALT$. However, M halts immediately in fewer than $|y|$ steps while N halts in $100|y|$ steps. Thus, $(< M >, < N >) \in LEQ - HALT$. This is a contradiction.
3. If R does not halt, this implies that R believes that $(< M >, < N >) \notin LEQ - HALT$. However, M halts immediately in $|y|$ steps while N halts in $100|y|$ steps. Thus, $(< M >, < N >) \in LEQ - HALT$. This is a contradiction.

Hence, a contradiction is reached in each case. Thus, $LEQ - HALT$ is not recognizable.

4. **Alice in Turing Land (45 points)**
Alice is visiting her uncle over the weekend and by accident discovers a wondrous room. The room contains infinitely many light bulbs in a (long) row labelled in order with the numbers $1, 2, \ldots$. The room is dimly lit since only the first two light bulbs are turned on, and the rest are all turned off. Consequently, Alice does not see the Math Hatter. He suddenly jumps out from the semidarkness and shuts the door. "I have a riddle for you," he says with glee. "You shall only escape this room if you manage to make it so that only the first light bulb is on." Alice says, "Okay, I'll just turn off the second light bulb." The Math Hatter chuckles: "Oh, but were it only that easy! The lights can only be manipulated according to rules of my making." He specifies the following:

(i) A window size in the form of an integer $M \in \mathbb{N}$.

(ii) A set of substitution rules R. Each rule $r \in R$ is specified by an integer n (note that two rules of R can have different values for n) and a mapping of n light bulbs in states $(b_1, \ldots, b_n) \in \{\mathsf{ON}, \mathsf{OFF}\}^n$ to the states $(c_1, \ldots, c_n) \in \{\mathsf{ON}, \mathsf{OFF}\}^n$. I.e. given n light bulbs in a row, Alice can change them according to the rule

$$(b_1, \ldots, b_n) \mapsto (c_1, \ldots, c_n).$$

Note that any such change much be applied to all these light bulbs simultaneously.

(iii) A rule with value n may only applied to a consecutive sequence of light bulbs that are labeled by $k+1, k+2, \ldots, k+n$, where k is a multiple of M. For example, if a rule has $n = 3$ and $M = 10$, then it can be applied to the light bulbs labeled $(11, 12, 13)$, but not the light bulbs labeled $(4, 5, 6)$, since $4 - 1 = 3$ is not a multiple of $M = 10$.

We call a pair (M, R) as specified above *Alice-Escapable* if it is possible to start with only the first two light bulbs on and, by repeated substitution according to rules in R and the conditions above, reach a state where only the first light bulb is on.

Example: Say that the Math Hatter specifies $M = 2$ and three substitution rules

$$n_1 = 4 \text{ and } (\mathsf{ON}, \mathsf{ON}, \mathsf{OFF}, \mathsf{OFF}) \mapsto (\mathsf{ON}, \mathsf{ON}, \mathsf{ON}, \mathsf{OFF}) \tag{1}$$
$$n_2 = 2 \text{ and } (\mathsf{ON}, \mathsf{OFF}) \mapsto (\mathsf{ON}, \mathsf{ON}) \tag{2}$$
$$n_3 = 4 \text{ and } (\mathsf{ON}, \mathsf{ON}, \mathsf{ON}, \mathsf{ON}) \mapsto (\mathsf{ON}, \mathsf{OFF}, \mathsf{OFF}, \mathsf{OFF}). \tag{3}$$

Then Alice can achieve that only the first light bulb is on by applying the rules as follows. Let the row of light bulbs initially be

$$\mathsf{ON}, \mathsf{ON}, \mathsf{OFF}, \mathsf{OFF}, \mathsf{OFF}, \ldots$$

Then first using rule (1) to the first four lights, Alice gets

$$\mathsf{ON}, \mathsf{ON}, \mathsf{OFF}, \mathsf{OFF}, \mathsf{OFF}, \cdots \mapsto \mathsf{ON}, \mathsf{ON}, \mathsf{ON}, \mathsf{OFF}, \mathsf{OFF}, \ldots$$

Then using rule (2), she gets

$$\mathsf{ON}, \mathsf{ON}, \mathsf{ON}, \mathsf{OFF}, \mathsf{OFF}, \cdots \mapsto \mathsf{ON}, \mathsf{ON}, \mathsf{ON}, \mathsf{ON}, \mathsf{OFF}, \ldots,$$

Finally, she uses rule (3) to escape with

$$\mathsf{ON}, \mathsf{ON}, \mathsf{ON}, \mathsf{ON}, \mathsf{OFF}, \cdots \mapsto \mathsf{ON}, \mathsf{OFF}, \mathsf{OFF}, \mathsf{OFF}, \mathsf{OFF}, \ldots$$

Note that there is no shorter solution to this example, because the rules have to be applied according to condition (iii) above. **End of Example.** You may **not** assume that this example is the actual (M, R) given by the Math Hatter.

(a) **(45 points).** Show that the language

$$\mathsf{ALICE} = \{(M, R) \mid (M, R) \text{ is Alice-Escapable}\}$$

is undecidable.

Solution:
(a) Intuition: Let ONbe 1 and OFFbe 0. Suppose we are given $M \in \mathbb{N}$, then each cell on the tape will contain a string of length M consisting of 0's and 1's which denotes the condition of the light-bulbs. Each cell also keeps track of the state of the machine (i.e. the state is encoded in the tape), and the head position. When a substitution is performed, the entire string of length M is substituted. If the substitution had length $n < M$, the substitution just changes the bits corresponding to the substitution and copies over the rest of the bits. In this way, the substitution will always occur starting at k s.t. k is a multiple of M.

Consider the following tape alphabet T:
$T = \{\$\} \cup (\tau \times (Q \times \{\#\}))$
where Q is the set of states, τ are the set of n length binary strings, \$ denotes the start of the tape, and # denotes that the head is not at this position. For simplicity, # will not be shown in diagrams. Note that the set of binary strings contains 2^n strings corresponding to each configuration of 0's and 1's possible in a string of length n.

Now, consider the following rules:
rule for the starting state:

\$	q_0 1100...

rule for starting state for positions that don't have the head:

#
0000...

rule for shifting right:

q	#		#	q'	
a	b	->	a'	b	

rule for shifting left :

#	q		q'	#	
a	b	->	a	b'	

rule for copying cells that don'r have the head:

#	->	#
a		a

Proof by contradiction:
Assume for contradiction that $ALICE$ is decidable and let D be a decider for $ALICE$.

Build M(y):
Let $x =< M >$
convert x to $\alpha = (M, R)$
Run $D(\alpha)$:
D ACCEPTS:
HALT

D REJECTS:

Change first cell using the rule $1100... -> 1000...$

Analysis:
1. If D ACCEPTS, this implies that D believes that (M, R) is Alice-escapable. However, M HALTS by design. Thus, $(M, R) \notin ALICE$. This is a contradiction.
2. If D REJECTS, this implies that D believes that $(M, R) \notin ALICE$. However, M immediately changed the first cell to 10000... and Alice can escape. Thus, $(M, R) \in ALICE$. This is a contradiction.

The Infinite Weaver (55 points + 50 Extra Credit)
In Infinite Land, every day consists of one minute for every positive integer. So, a clock in Infinite Land does not have minutes 1 to 60 and hours 1 to 24, but instead just minutes $1, 2, \ldots$. In Infinite Land lives the Infinite Weaver. Every minute, the Infinite Weaver weaves a carpet of any size (even infinite) that can fit on the upper right quadrant of $\mathbb{R}^2$. Any carpet has a pattern consisting of a coloring of the unit squares of the carpet.

Now, the Infinite Customer enters the story. She has heard of the Infinite Weaver's wonderful carpets, and to decorate her Infinite House, she needs infinitely many carpets. She first goes to the weaver and orders all possible carpets of width 1, of finite length, and with every possible pattern containing the colors red and blue. In other words, for every $n \in \mathbb{N}$ the order contains all carpets in red and blue of size $1 \times n$.

(a) **(10 points).** Show that the Infinite Weaver can deliver the order in one (infinite) day.

Seeing as the Infinite Weaver so easily completed her first order, the Infinite Customer gets curious. How much can the weaver actually hope to deliver? Now, she asks the weaver to weave every possible carpet of finite width $w \in \mathbb{N}$, of finite length $n \in \mathbb{N}$, and with every possible pattern of red and blue colors.

(b) **(10 points).** Show that the Infinite Weaver can still deliver the order in one (infinite) day.

The Infinite Customer decides that she likes other colors besides blue and red. She creates infinitely many colors $\{c_i\}_{i\in\mathbb{N}}$. She requests that the Infinite Weaver produce every possible carpet of width 1, of finite length $n \in \mathbb{N}$, and with every possible sequence of n colors among the colors $\{c_i\}_{i\in\mathbb{N}}$.

(c) **(20 points).** Show that once again the Infinite Weaver can deliver the order in one (infinite) day.

The Infinite Weaver, enjoying her successful business, hires infinitely many weavers $\{W_i\}_{i\in\mathbb{N}}$. Each employee W_i has the same capabilities as the Infinite Weaver.

(d) **(15 points).** Show that, in fact, every order that the Infinite Weaver can fulfill in one day with the help of her infinite employees, she could have also fulfilled on her own in one day.

Solution:
(a) Let 1 and 0 represent the colors blue and red, respectively. Now, a carpet of size $1 \times n$ can be represented as a binary string of length n. Let this binary representation be called B.

Now, consider the following mapping function that maps a given representation B to a natural number k:
$2(2^{|B|-1} - 1) + B + 1 = k$
Here B's value in decimal representation is being added.

Now, every string B of length n can have 2^n different combinations. Thus, to give each carpet a unique number, all the previous carpets (carpets of smaller size) must be given a number first. Number of carpets (i.e. natural numbers) already taken for size $|B|-1 = 2(2^{|B|-1}-1)$ using the geometric sequence summation formula. Hence, now the numbers available for size $|B|$ are $2(2^{|B|-1} - 1) + 1$, $2(2^{|B|-1} - 1) + 2$, $2(2^{|B|-1} - 1) + 3 \ldots$. In this way, each carpet can be given a unique natural number such that all natural numbers are used in sequence. Hence, the sizes of the sets of one (Infinite) day and the total number of $1 \times n$ carpets is the same and the Weaver can make the delivery.

(b) Consider the following scheme of numbering the carpets:
Start with a square of dimensions 1×1 in the upper-right quadrant of $\mathbb{R}^2$ such that the sqaure touches the x-axis and the y-axis and has one vertex at Origin, enumerate all the carpets (including a different carpet for different configurations of 1's and 0's) that are present in this square and give them the next available natural number (the first number assigned is 1). Next, move to a square of dimensions 2×2 and repeat the process; however, this time exclude the carpets already enumerated by the smaller square. Continue for larger and larger squares each time incrementing the side of the square by 1. Note that the number of squares numbered for each square area is finite.

Now, if the above scheme is followed, each every possible carpet of dimensions $m \times n$ s.t. $m \geq n$ will be reached and enumerated when we reach the square with side m. Hence, each carpet can be mapped to a natural number (a minute). Thus, the Weaver can make a delivery in one (Infinite) day.

(c) Intuition: Since there are an infinite number of colors, even a 1×1 carpet has infinite permutations each corresponding to a different color. Thus, if we start enumerating just 1×1 carpets, we will never get to higher carpets. Hence, an approach that repeatedly moves to bigger carpets but revisits smaller carpets is required.

Consider the following visual representation of the problem:
Let the carpet of length i be represented along the y-axis starting at $x = i$. Thus, starting at the coordinates (i,0), there is a carpet extending to (i,i) in the xy-plane. Let the carpets of all lengths $i \in \mathbb{N}$ be represented similarly. Hence, when the xy-plane is viewed from above, we will see columns of increasing height as we move towards the right (positive x-axis).
Now, represent the infinite number of color combinations on the z-axis.

AN ASIDE THAT IS NECESSARY
Claim: Infinite color combinations for a carpet of dimensions $1 \times n$ can be represented along a single axis. In other words, all the combinations are countably infinite.
Proof: Consider the trivial case of carpet of dimensions 1×1. Here the color combinations are just the set of infinite colors C_i. Hence, they have a direct mapping C_i to i.
Next, consider the case of carpets of dimension 1×2. Here, the color combinations are the set $C_i \times C_j$. Represent C_i along x-axis and C_j along the y-axis. Draw diagonal lines of slope -1 cutting the plane starting at y-intercept of 2, then 3, then 4, and so on and number the points in the plane encountered. In this way, the point with coordinates (m, n) which lies on the line $y = -x + (m + n)$ will be reached when the line with y-intercept $m + n$ is reached. Hence, this set is also countably infinite.
For higher dimension cases, the case of $C_i \times C_j \times C_k$ can be represented in the xyz-space and all points can be reached by a cube (with one vertex on the Origin, and three sides along the x-axis, the y-axis, and the z-axis) of increasing length of each side. Similarly, the 4D case can be solved using enumeration along a hypercube and so on.
END OF ASIDE

Now, we have the carpets along the x-axis (with their lengths along the y-axis) and the color combinations along the z-axis. Consider the view of the xz-plane with the positive y-axis going into the plane (away from the viewer). Any point (a, b) in this plane (the xz-plane) can be reached by drawing a diagonal line of the form: $z = -x + (a + b)$ following the same argument as presented above in *AN ASIDE THAT IS NECESSARY*. Thus, we have shown that any carpet of length b and a color combination a can be reached and numbered using a natural number. Thus, the Weaver can make the delivery.

(d) Consider the following visual representation for the problem:
Let the employees W_i be represented on the x-axis with employee W_i at position $x = i$. Let the work of an employee be represented on the y-axis with the $j'th$ carpet at position $y = j$. Thus, each carpet j made by the employee W_i is represented at the coordinates (i, j). Now, since there are infinite employees and an infinite number of carpets for each employee, this representation covers the entire upper-right quadrant of $\mathbb{R}^2$.

Now, if we can show that the Infinite Weaver can enumerate all these carpets (the upper-right quadrant of $\mathbb{R}^2$), then we would have shown that the Infinite Weaver can make these on her own since we will find a 1:1 mapping for each carpet to a minute.

Consider the following numbering scheme:

1. Start the numbering from 1 and number the carpet at (1,1).
2. Next, number (1,2) and (2,1).
3. Next, number (1,3), (2,2), and (3,1).
4. Continue in the same pattern.

This numbering scheme essentially numbers the quadrant along diagonal lines cutting the plane starting from a point on the y-axis down to a point on the x-axis such that the points have the same value (i.e. the slope of the line is -1).

Claim: This numbering scheme reaches all the points in the upper-right quadrant of $\mathbb{R}^2$.
Proof: Consider an arbitrary point with the coordinates (m, n). Now, this point lies on the line $y = -x + (m + n)$, which is a line with slope -1 and y-intercept $m + n$. Thus, this point will be enumerated by the diagonal drawn starting at $y = m + n$, which will be reached at step number $m + n - 1$ as per the numbering scheme described above. Thus, eventually every point will be numbered.

Hence, the Infinite Weaver can fulfil every order on her own in a day that she can fulfil in a day with all her employees combined.

EXTRA CREDIT
Assume for contradiction that the Infinite Weaver can produce every infinite carpet, each the size of upper right-quadrant of $\mathbb{R}_2$ and each with infinite colors.

Let the $i'th$ carpet produced by the Infinite Weaver be represented by $Carp_i$. Now, consider the carpet constructed by the following steps:
For each carpet $Carp_i$, copy the entire $i'th$ row but change the unit square at the position (i, i) to any other arbitrary color different from the present color. Thus, row 1 of the new carpet is identical to that of $Carp_1$ but different at position 1 (i.e. (1,1) in the new carpet). Row 2 of the carpet is identical to row 2 of $Carp_2$ but different at position 2 (i.e. (2,2) in the new carpet). Hence, the new carpet is different from the $i'th$ carpet at position (i, i).

Hence, our new carpet could not have been one of those produced by the Weaver. This is a contradiction.

Thus, the Weaver cannot produce every infinite carpet the size of the upper-right quadrant of $\mathbb{R}^2$ with infinite colors.