1. Which of the following statements are correct. Write True/False only. Proof not required - Negative 2 for each incorrect answer, \((5 \times 3)\)

(a) \(L = \bigcup_{i=1}^{\infty} L_i\) is regular where \(L_i\) is regular.

False. Otherwise \(\bigcup_{i=1}^{\infty} a^i \cdot b^i\) will be regular.

(b) \(H_k = \{a_1a_2 \ldots a_k, k \geq 2 | (a_i a_{i+1}) \in R\}\) is regular where \(R \subset \Sigma \times \Sigma\).

True. The DFA could be built on states in \(\Sigma \times \Sigma\) that represents the last two symbols and the final states correspond to the relation \(R\).

(c) All strings over \(\{a,b,c,d\}\) that do not contain equal number of \(a,b,c,d\)'s, is context free language. Note that \(aabcc\) is in the language but \(abccadbd\) is not.

True. Your PDA can guess an unbalanced pair and verify if the counts are different.

(d) \(L^R\) is CFL where \(L\) is any CFL and \(L^R = \{rev(w) | w \in L\}\) and \(rev(w) = \) reverse of the string \(w\), for example, \(rev(0101) = 1010\).

True. Consider a CNF of the grammar and flip the two variables on the RHS for all productions.

(e) Let \(L = \{w \cdot w | w \in (0 + 1)^*\}\). Then \(\bar{L}\) is CFL.

True. All odd length strings are accepted. For even length strings consider the following grammar. \(S \Rightarrow AB | BA\)
\(A \Rightarrow CAC | a\) \hspace{1cm} \(B \Rightarrow CBC | b\).
\(C \Rightarrow a | b\).

2. Is the following problem decidable - For a given regular expression \(r\), does the language \(L(r)\) contain exactly 100 strings ? Justify. \((10\) )

From pumping Lemma of regular languages, we know that if there is a string of length \(\geq n\) accepted by the corresponding DFA, then there are infinite number of strings in the language. It also follows from P.L. that for all strings \(\geq n\), we can get a strictly smaller string \(u \cdot v^0 \cdot w\) shorter by at least length \(n\). So, we can determine if the r.e. generates infinite strings by trying all strings between lengths \(n\) and \(2n\) in the equivalent DFA. If any of the strings is accepted, the answer is No. Otherwise, we try all strings up to length \(n\) to check if exactly 100 strings are accepted.

Comment For the answers that did not use PL, it was expected that the algorithm for checking out "if there are exactly 100 paths from \(q_0\) to some accepting states" is described as an explicit algorithm. It is not a standard problem, so marks were deducted for lack of preciseness in terms of finite termination of the algorithm or ambiguity. The same policy was followed for answers trying to deal directly with r.e.

3. A shuffle of two strings \(x, y \in \Sigma^*\) denoted by \(x||y\) is the set of strings that can be obtained by interleaving the strings \(x\) and \(y\) in any manner. For example \(ab||cd = \{abcd, acbd, acdb, cabd, cadb, cdab\}\).
(The strings need not be of the same length.) For two sets of strings \(A, B\), the shuffle is defined as \(A||B = \bigcup_{x \in A, y \in B} x||y\). \((8 \times 8)\)
4. Is the following problem decidable -
   Given an undirected graph $G = (V,E)$ a dominating set is a subset $D \subset V$ such that, every for $v \in V$, either $v \in D$ or $w \in D$ where $(v,w) \in E$. That is either the vertex is in $D$ or one of its neighbours is in $D$. For an integer $k \leq |V|$, we want to determine if there is a dominating set of

(a) If $A, B$ are CFL, is $A \parallel B$ a CFL ?

   False. Consider the two CFL’s $a^i \cdot b^j$ and $c^j \cdot d^j$. The shuffle of the languages consists of strings that have equal number of $a$ and $b$’s and equal number of $c$’s and $d$’s. Consider the shuffle $a^na^jb^nd^n$ for a large $n$ according to PL of CFL. Clearly we can create imbalance between $a$ and $b$’s or $c$ and $d$’s since they are apart by $n$. So it can’yt be CFL.

(b) If $A, B$ are regular is $A \parallel B$ regular ?

   True. Consider $M_1$ and $M_2$ as the two DFAs for $A$ and $B$ respectively. Let $M$ denote the product of the two machines where the final states $(p_f, q_f)$ correspond to final states $p_f \in F_1, q_f \in F_2$. The transition function $\delta((p_i, q_j), a) = \{\delta_1(p_i, a), \delta_2(q_j, a)\}$, i.e. it is a non-deterministic machine that either makes the transition according to $M_1$ or $M_2$ (not both). So, the machine accepts a string $w_1, w_2, \ldots w_k$ iff there is a partition $a_1, a_2 \ldots a_r$ and $b_1, b_2 \ldots b_s$, $a_i, b_i \in \{w_1, w_2, \ldots w_k\}$ and $\delta_1(p_0, a) \in F_1$ and $\delta_2(q_0, b) \in F_2$.

4. Is the following problem decidable -
   Given a TM $M$ and $w$, does $M$ use a finite number of tape cells for input $w$ ? (12)

   Not decidable. We will reduce $L_u$ to $L_f (= <M, w> | M$ uses a finite number of cells on $w)$.

   The reduction function $f(M, w) = <M', w>$ where $M'$ behaves as follows
   Simulate $M$ on $w$ until
   (i) $M$ accepts then stop else
   (ii) If $M$ rejects $w$ or if any of the previous ID repeats (it stores all IDs on a separate tape)
   then keep moving the head rightwards for ever.

   Claim: $<M, w> \in L_u$ iff $<M', w> \in L_f$.
   If $M$ accepts $w$ then clearly it uses finite number of cells.
   If $M$ doesn’t accept $w$ it either halts and rejects or it goes into an infinite loop (ID repeats) or it uses infinite cells. In all these cases $M'$ uses infinite cells.
   Comment Even if you got the basic intuitions correct in terms of contradiction but did not present an explicit many-one reduction function, some marks were deducted depending on the case analysis.

5. Is NTIME($f(n)$) $\subseteq$ DSPACE($f(n)$) ? Justify. (12)

   NTIME($f(n)$) $\Rightarrow$ NSPACE($f(n)$), so any ID can be written in $f(n)$ space. The non-deterministic choices from any ID is fixed (depends on the transition function of the TM), so from any given ID, we can retrace the previous ID, using fixed amount of extra space. We do not need to store the exact position of the tape head - only the relative movement suffices.

   A deterministic TM can explore all the computational trajectories using a DFS based approach where we store the present path from the initial ID with $O(1)$ information per node. Since NTIME is bounded by $f(n)$, the maximum depth of the tree is $f(n)$. So the total space we need is $O(f(n))$. Using space compression techniques this can be reduced to $f(n)$, so the entire computation is in DSPACE($f(n)$).

   Comment Using Savitch’s theorem would lead to incorrect conclusion.

6. Dominating set Given an undirected graph $G = (V,E)$ a dominating set is a subset $D \subset V$ such that,
size \leq k. For example a triangle has a dominating set of size one.

Show that the problem is NP-Complete. (15)

Note: You can use any of the NPC problems discussed in the class.

The problem is in NP, since we can verify if a subset is dominating in polynomial time very easily.

Let us reduce Vertex cover problem to Dominating set problem. Given a graph $G = (V, E)$, we construct $G' = (V \cup W, E \cup E')$ where $W = \{v_{i,j} | (v_i, v_j) \in E\}$ and $E' = \{(v_{i,j}, v_i) \cup (v_{i,j}, v_j) | v_{i,j} \in W\}$. So $G'$ contains $|E|$ additional vertices and $2|E|$ additional edges.

**Claim** $G$ contains a vertex cover of size $k$ iff $G'$ contains a dominating set of size $k$.

Clearly any vertex cover in $G$ forms a dominating set in $G'$. For the converse, assume that the dominating set $D$ contains some vertices in $W$. We now map a vertex $v_{i,j} \in W$ (arbitrarily) to either $v_i$ or $v_j$. Therefore edge $(v_i, v_j)$ is covered. For the other edges $(v_i, v_j) \in E$, either $v_i$ or $v_j$ must be in $D$, otherwise $v_{i,j}$ will not be dominated. Hence after reassigning the vertices, we have a cover of $G$ that has size $k$.

**Comment** Trying to use reduction from 3SAT along the lines of the proof of hardness of vertex cover would be difficult since choosing any vertex from a triangle is a dominating set whereas the correlations between clauses are not considered.