Solution of Homework 2

1. (a) i. This language is in $C \setminus R$, it is not in $R$ since it complement is not in $R$ and it is in $C$ because there is a CF grammar for it:
   
   \[
   S \rightarrow A|aAa|bAb
   \]
   
   \[
   A \rightarrow aBb|bBa|a|b
   \]
   
   \[
   B \rightarrow aB|bB|e
   \]

   ii. This language is not in $C$, see example 2.22 in the textbook

   iii. This language is in $C \setminus R$, see example 2.11 in the textbook and 1 (iii).

   (b) i. $a^n b^n c^n d^n$

   ii. $S_0 \rightarrow A E | A D | B F | B C | e$

   \[
   S \rightarrow A E | A D | B F | B C
   \]

   $S_2 \rightarrow B F | B C$

   $E \rightarrow S D$

   $F \rightarrow S_2 C$

   $A \rightarrow a$

   $B \rightarrow b$

   $C \rightarrow c$

   $D \rightarrow d$

2. (a) Yes. Push $a$'s onto stack until $b$ is encountered, pop an $a$ for every new $a$, reject iff stack becomes empty at the last $a$.

   (b) Yes. Push $a$'s onto stack, pop an $a$ for every $b$, push $c$'s onto stack, pop a $c$ for every $d$, reject if there is mismatch.

   (c) No. Using pumping lemma, let $w = a^p b^p c^p d^p = uvxy$ then $v$ and $y$ cannot contain both $a$'s and $c$'s or both $b$'s and $d$'s, hence $u^2 v^2 y^2 z$ will have imbalance of either $a$'s vs $c$'s or $b$'s vs $d$'s.

   (d) No. Using pumping lemma, let $w = a^p b^p a^p = uvxy$ then $u^2 v^2 y^2 z$ will either have more $a$'s on one side than on the other side, or more $b$'s than $a$'s combined.

   (e) No. Using pumping lemma, let $w = a^{p^2 + 3} = uvxyz$, then $u^2 v^2 y^2 z = a^{2^2 + 3} + |v| + |y|$ but $p^2 + 3 < p^2 + 3 + |v| + |y| \leq p^2 + 3 + p < (p+1)^2 + 3$, contradiction.

3. (a) i. $RC \not\subseteq C$. Let $L_1 = 0^*1^*$, $L_2 =$ language described in 6(i), then $L_1 \setminus L_2 = \text{the language } \{0^m1^n : m \neq n \text{ and } 2m \neq n \}$ which is not in $C$ - stdwd use of pumping lemma (many cases for choice of pumping constant $i$ though, depending on how adversary chooses to split string).

   ii. $CR \subseteq C$. Pushdown automata that accepts $L_1 \setminus L_2$ is similar to the one that accepts $L_1 \cap L_2$ except for final states.
(b) No. Let \( L = \{a^ib^jc^{3|j-i|} : i, j \in \mathbb{N}\} \). This language is in \( C \) because there is PDA for it - push all a’s onto a stack, pop an a for every b, either there are still some a remaining in the stack after b ended, or there are more b’s than a’s then push b’s onto a stack. By the time the first c appears there will be \( |i-j| \) symbols (a’s or b’s) in the stack. Check whether twice the number of elements in the stack is the number of c’s by popping a symbol for every two c’s.

Let \( x = a^ib^jc^{|i-j|} \in L \), then \( |x| = i+j+2\max(i,j)-2\min(i,j) \), \( |x|/2-i-j = \max(i,j) - \min(i,j) \) so \( x \) contains any c’s if \( \max(i,j) - \min(i,j) - i/2 - j/2 > 0 \) i.e. \( i > 3j \) if \( i = \max(i,j) \) or \( j > 3i \) if \( j = \max(i,j) \).

Therefore \( 1/2L = \{a^ib^jc^{|i-3j|}/2 \ : \ i \geq 3j\} \cup \{a^ib^jc^{|j-3i|}/2 \ : \ j \geq 3i\} \cup \{a^ib^jc^{|j-i-3i|}/2 \ : \ i \leq j < 3i\} \cup \{a^ib^jc^{|i-j-3i|}/2 \ : \ j \leq i < 3j\} \).

Take a regular language \( A = a^*b^*c^*(cc)^* \). If \( 1/2L \) were context-free then \( 1/2L \cap A = \{a^ib^jc^{|i-3j|}/2 \ : \ i \geq 3j\} \) (and \( (i-3j)/2 \) is odd) \( \cup \{a^ib^jc^{|j-3i|}/2 \ : \ j \geq 3i\} \) (and \( (j-3i)/2 \) is odd) were context-free.

Let \( p \) be a constant given by pumping lemma, \( w = a^pb^{3p+2}c \), then \( w \in 1/2L \cap A \) since \( a^pb^{3p+2}c^{k+4} \in L \). Since \( |wxy| < p \), \( wxy \) could be a string of a’s, a string of b’s and c’s, a string of b’s, a string of a’s and c’s or a string of c’s.

Suppose that \( wxy \) is a string of a’s. Then \( uv^ixy^iz \notin 1/2L \cap A \) for \( i = 1000p \) since number of a’s increases number of c’s should also increase, but there are no c’s in \( vxy \).

Suppose that \( wxy \) is a string of b’s. Then \( uv^ixy^iz \notin 1/2L \cap A \) for \( i = 1000p \) since number of b’s increases number of c’s should also increase, but there are no c’s in \( vxy \).

Suppose that \( wxy \) is a string of c’s. Then \( uv^ixy^iz \notin 1/2L \cap A \) for \( i = 2 \) since we cannot have even number of c’s (recall that \( A = a^*b^*c^*(cc)^* \)).

Suppose that \( wxy \) is a string of b’s and c’s. Then \( uv^ixy^iz \notin 1/2L \cap A \) for \( i = 2 \) since we cannot have even number of c’s.

Suppose that \( wxy \) is a string of a’s and b’s. Let \( r \) denote the number of a’s in \( vxy \), \( q \) denote the number of b’s in \( vxy \). Then if \( q \leq r < 3q \) or \( q \leq r < 3r \) then \( uv^ixy^iz \notin 1/2L \cap A \) for \( i = 1000p \) since there should not be any c’s present. If \( r \geq q \) or \( q \geq 3r \) then \( uv^ixy^iz \notin 1/2L \cap A \) for \( i = 1000p \) since as number of a’s or b’s increases number of c’s should also increase, but there are no c’s in \( vxy \).

Therefore \( 1/2L \) is not context-free.

Note: there a stronger version of pumping lemma that might have been helpful here.

**Stronger version of pumping lemma:** There is a number \( p \) such that for all strings \( s \in L \) with \( |s| > p \) for all ways of marking \( p \) positions of \( s \), there is a split of \( s \) into \( uvxyz \) such that:
a) $v$ and $y$ together contain at least 1 marked position
b) $vxy$ together contains at most $p$ marked positions
c) For all $i$ $uv^ixyz \in L$

For example, above we could have marked all $a's$, then we wouldn’t have to consider cases where $vxy$ contains $c's$.

The proof of this slightly stronger pumping lemma proceeds exactly as in the pumping lemma case, except

(i) you use the Chomsky normal form for the derivation tree, so each vertex has exactly 2 children.

(ii) you can easily show there is a path in the tree not of length $|V| + 1$, but which has at least $|V| + 1$ “good” vertices *both* of whose children yield marked positions. So there must be at least one repeated variable among these “good vertices.” The rest of the proof follows similarly.

(iii) you argue (a) slightly differently, not based on smallest parse tree, but based on the fact that it is a CNF parse tree, so consider the variable $A$ that derives $vxy$. It first splits into $B$ and $C$ (using a production $A \rightarrow BC$ in CNF). Now the $w$ part, derived by the next occurrence of $A$ along the chosen path, sits either entirely under the $C$ or entirely under the $B$. This means either $v$ or $y$ is nonempty, since there are no productions of the form $V \rightarrow \epsilon$, except for the start variable.

(c) No. Language $L = \{a^n b^n\}$ is in $C$ but $\text{REFLECT}(L) = \{a^n b^n a^n\}$ is not in $C$, see 3(iv).

(d) No. Let $L = a^i b^j c^{2i-j} d^{2(m+n)} b^n a^n$. This language is context-free (it is easy to design a PDA for it). Then $\text{Inv - Reflect}(L) = a^i b^j c^{2i-j} d^{2(m+n)}$. This language is not context-free. Take $s = a^b \epsilon c^p \in \text{Inv - Reflect}(L)$. Using stronger version of pumping lemma described in 4(ii) we mark all $p a's$. Hence $vxy$ doesn’t contain any $c's$.

We need to consider 3 cases: $vy$ contains only $a's$, only $b's$, or both $a's$ and $b's$. In all three cases the string $uv^ixyz \notin 1/2L \cap A$ for $i = p$ since when combined number of $a's$ and $b's$ is greater than $4p$ number of $c's$ should also be greater than $4p$, but there are no $c's$ in $vxy$, hence number of $c's$ cannot increase.

(e) Yes. Let $L$ be regular then it has context-free grammar that is a set of rules

$A_i \rightarrow b_i C_j$

$D_k \rightarrow \epsilon$
then the context-free grammar for $\text{REFLECT}(L)$ is a set of rules

\[ A_i \rightarrow b_iC_jb_i \]
\[ D_k \rightarrow \epsilon \]

for every rule of $L$.

4. (a) This language is not regular since it complement 2(i)(c) is not regular.

It has a following context-free grammar, hence it is context-free.

\[ S \rightarrow AB\mid BA \]
\[ A \rightarrow a\mid aAa\mid aBb\mid bAa\mid bAb \]
\[ B \rightarrow b\mid bBb\mid bBa\mid aBb\mid aBa \]

(b) Class $C$ is not closed under $\text{Shuffle}$ operation. Let $L_1 = \{a^n b^n\} \in C, L_2 = \{c^m d^m\} \in C$. Let $A = \text{Shuffle}(L_1, L_2)$. Let $B = \{a^*c^*b^*d^*\} \in R$ then if $A$ were in $C$ then $A \cap B = \{a^n c^n b^n d^n\}$ were in $C$, but it is not, see 3(iii), hence $A \notin C$.

Class $R$ is closed under $\text{Shuffle}$ operation. NFA for $\text{Shuffle}(L_1, L_2)$ has $|Q_1| \times |Q_2|$ states where $Q_i$ is a set of states of language $L_i, i = 1, 2$. Transition function for $\text{Shuffle}(L_1, L_2)$ will be $\delta'((q_1, q_2), a) = \{(\delta_1(q_1, a), q_2), (q_1, (\delta_2(q_2, a)))\}$ Final states of $\text{Shuffle}(L_1, L_2) = (F_1, F_2)$.

5. (a) No, there is no such computable bijection.

Proof by contradiction (and diagonalization). Let $A$ be the set of algorithms, i.e machines $\{M : \forall x\exists y$ such that $M$ halts on $x$ in $y$ steps $\}$. Suppose that such computable bijection $f : A \leftrightarrow N$ exists.

Let $A = \{M_1, M_2, \ldots\}$. Define set $D = \{i \in N | i \notin L(M_i)\}$. Given $i$ we can compute $M_i = f^{-1}(i)$, simulate $M_i$ on $i$ and accept if $M_i$ rejects and reject if $M_i$ accepts. Therefore $D$ is decidable and there a decide $M_j$ for $D$. But if $M_j$ accepts $j$ than it should reject $j$ and if $M_j$ rejects $j$ it should accept $j$, contradiction.

(b) Halting problem for machines that have finite memory is decidable. Suppose that machine $M$ has $n$ bits of memory. Then $M$ has at most $2^n$ different states. Therefore if $M$ runs for more than $2^n$ steps then it is caught in the infinite loop and will never halt. Hence halting problem for $M$ could be decided as follows: run $M$ for $2^n$ steps, if $M$ has halted then accept, otherwise reject.