1. Consider the context-free grammar

\[ S \rightarrow aaSb \mid aSb \mid b \]

Here \( S \) is the start nonterminal, and \( a, b \) are terminals.

(i) Give a parse tree for the string \( aabb \).
(ii) Give a parse tree for the string \( aaabbb \).
(iii) Is the above grammar ambiguous? Justify your answer!
(iv) Does the grammar allow the use of recursive-descent parsing? Justify your answer!
(v) Is the language generated by the grammar regular? Circle the correct answer:
   YES \[ \text{NO} \] (The answer to (v) does not require an explanation.)

\[ \text{iii) Grammar is ambiguous because } a^3b^3 \text{ has also parse tree }  \]

\[ \text{iv) No. } \text{first}(aaSb) \cap \text{first}(aSb) \neq \emptyset \]
2. In this question the alphabet is $\Sigma = \{0, 1\}$. For the following three sets of strings over $\Sigma$, give a deterministic state transition diagram that accepts it.

(i) All strings over $\{0, 1\}$ that have an odd number of 0's (and any number of 1's).
   Deterministic state diagram:
   
   ![Diagram](image)

(ii) All strings over $\{0, 1\}$ that have at least one occurrence of 0 and end with a 1.
    Deterministic state diagram:
    
    ![Diagram](image)

(iii) All strings over $\{0, 1\}$ that have do not have 010 as a substring.
     Deterministic state diagram:
     
     ![Diagram](image)
3. (i) (1.5 marks) Give a regular expression that defines the language
\[ A = \{ c^{4m+3}b^{3k+2}a^{2i+1} \mid m \geq 0, k \geq 0, i \geq 0 \} \]
\[(c^4)^* c^3 (b^3)^* b^2 (a^2)^* a\]

(ii) (1.5 marks) Give a context-free grammar that generates the language
\[ B = \{ b^ic^kd^m \mid k = i + m, i \geq 0, k \geq 0, m \geq 0 \} \]

\[ S \rightarrow AB \]
\[ A \rightarrow bAc \mid \varepsilon \]
\[ B \rightarrow cBd \mid \varepsilon \]

(iii) (2 marks) Using left-factoring and/or elimination of left-recursion give grammars equivalent to the below two grammars where the immediate problems preventing use of recursive-descent parsing have been removed. Capital letters denote variables and the set of terminals is \{a, b, c, d\}.

(a) \[ S \rightarrow abSc \mid acS \mid ad \mid cbS \mid dc \]
\[ S \rightarrow aSS' \mid cbS \mid dc \]
\[ S' \rightarrow bSc \mid cS \mid d \]

(b) \[ S \rightarrow Sbc \mid Sdc \mid Sadd \mid abS \mid \varepsilon \]
\[ S \rightarrow abSS' \mid S' \]
\[ S' \rightarrow bcS' \mid dcS' \mid addS' \mid \varepsilon \]
4. Verify the validity of the following correctness statements by adding all the intermediate assertions (that is, give the proof tableau). All variables are of type int. Clearly state any mathematical facts and inference rules used.

(i) \[
\text{ASSERT}\ (x < 3 \land y > 2) \\
x = x - y; \\
y = x + y; \\
x = x + 2; \\
\text{ASSERT}\ (x < 2 \land y < 3) \\
\text{Solution:} \\
\text{ASSERT}\ (x < 3 \land y > 2) \\
// x < 3 \implies \& \land y > 2 \implies x-y < 0 \text{ when } x, y \text{ int} \\
\text{ASSERT}\ (x - y < 0 \land x - y + y < 3) \\
x = x - y; \\
\text{ASSERT}\ (x < 0 \land x + y < 3) \\
y = x + y; \\
\text{ASSERT}\ (x + 2 < 2 \land y < 3) \\
x = x + 2; \\
\text{ASSERT}\ (x < 2 \land y < 3) \\
\]

(ii) \[
\text{ASSERT}\ (x == 5 \land y == 6 \land z < -1) \\
x = y + z; \\
y = y - x; \\
x = x + z; \\
\text{ASSERT}\ (x \leq 5 \land y \geq 2) \\
\text{Solution:} \\
\text{ASSERT}\ (x == 5 \land y == 6 \land z < -1) \\
// z < -1 \implies z \leq -2 \text{ when } z \text{ is integer} \\
// y == 6 \land z < -1 \implies y+2*z \leq 5 \\
\text{ASSERT}\ (y + 2*z \leq 5 \land y - (y+z) \geq 2) \\
x = y + z; \\
\text{ASSERT}\ (x + z \leq 5 \land y - x \geq 2) \\
y = y - x; \\
\text{ASSERT}\ (x + z \leq 5 \land y \geq 2) \\
x = x + z; \\
\text{ASSERT}\ (x \leq 5 \land y \geq 2) \\
\]
5. (i) (2 marks) What should the pre-condition \( P \) be in each of the following correctness statements for the statement to be an instance of Hoare's axiom scheme? All variables are of type int.

(a) \( P \{ z = x + y; \} \) \( \exists y \) \( (y = 0; y < z) \) \( 9 * z + 2 \geq w + y \)

\[ \exists y \ (a = 0; a < x+y) \ 9 \times (x+y) + 2 \geq w + a \]

(b) \( P \{ x = y + z; \} \) \( \forall y \) \( (y=0; y < x) \) \( \exists z \) \( (z = 0; z < y) \) \( 3 * x \geq y * z + 1 \)

\[ \forall y \ (a = 0; a < y+z) \ \exists z \ (b = 0; b < a) \ 3 \times (y+z) \geq a \times b + 1 \]

(ii) (2 marks) Use the array-component assignment axiom (two times) to find the most general sufficient pre-condition \( P \) for the following code fragment:

```c
ASSERT(P) /*determine what is P*/
A[m] = A[k];
A[k] = x;
ASSERT( A[i] >= A[k] + 2 )
```

Above \( A \) is an array of integers, \( x \) is an integer variable and we assume that all the subscripts are within the range of subscripts for \( A \).

Write the assertion \( P \) first using the notation from the array-component assignment axiom, and then rewrite \( P \) in a logically equivalent form that does not contain any notation \( (A | I \mapsto E) \).

```c
ASSERT( ( (A[m]) | (k \mapsto x) )[i] >= x + 2 )
A[m] = A[k];
ASSERT( (A[k]) | (k \mapsto x) )[i] >= (A[k])[k] + 2 )
A[k] = x;
ASSERT( A[i] >= A[k] + 2 )
```

Evaluate pre-cond. \( i = k \): \( x = x + 2 \) \( \text{false} \)

\( i \neq k \): \( i = m \): \( A[k] >= x + 2 \)

\( i \neq m \): \( A[i] >= x + 2 \)

\( (i \neq k \&\& i = m \&\& A[k] >= x + 2 ) \)

\( (i \neq k \&\& i \neq m \&\& A[i] >= x + 2 ) \)
6. Are the following languages $A$ and $B$ context-free or not context-free?

- If a language is context-free, give a context-free grammar that generates it.
- If a language is not context-free, prove that it is not context-free.

(i) $A = \{ a^i b^k c^m d^m \mid i \geq 0, \; k \geq m \geq 0 \}$

(ii) $B = \{ d^k c^i b^m a^k \mid i \neq m, \; i \geq 0, \; m \geq 0, \; k \geq 0 \}$

i) We prove that $A$ is not CF. Suppose $A$ is CF and let $p$ be the constant given by the PL. Choose $s = b^p c^p d^p \epsilon A$. By the PL we can write $s = uvwxy$ where the parts satisfy condition of the PL.

a) If $v$ or $x$ contains more than one type of symbol, $uv^2w^2x^2y$ can not in $A$. and hence not in $A$.

b) In the following thin $v$ and $x$ each have at most one type of symbol. If $v$ or $x$ has $d$'s, $uv^2w^2x^2y$ has more $d$'s than $b$'s or more $d$'s than $c$'s and is not in $A$ (because $v, x$ cannot have both $b$'s and $d$'s). The last possibility is that $v$ and $x$ have only $b$'s and $c$'s. Now $uv^0wx^0y = uvwy$ has fewer $b$'s than $d$'s or fewer $c$'s than $d$'s and is not in $A$.

ii) Grammar for $B$: $S \rightarrow dSa | X \cdot Y$

$X \rightarrow cXb | cX | c$

$Y \rightarrow cYb | Yb | b$
7. For each question, circle one answer. If you circle more than one answer, it will be considered a wrong answer. If in doubt, it is to your advantage to make a guess.

(i) Consider the language \( L_1 = \{ a^i b^k \mid i \geq 0, \ k \geq 0 \} \).
   (a) The language \( L_1 \) is regular and context-free.
   (b) The language \( L_1 \) is context-free but not regular.
   (c) The language \( L_1 \) is regular but not context-free.
   (d) None of the above.

(ii) Let \( L \) be the language consisting of all strings over the alphabet \( \{ b, c \} \) having an equal number of \( b \)’s and \( c \)’s. The language \( L \) is denoted by the regular expression:
   (a) \( (b + c)^* + (bb + cc)^* \)
   (b) \( (bc + cb)^* \)
   (c) \( (bc + cb + bbcc + bcbc + bcbb + cbcb + cbcc + ccbb)^* \)
   (d) \( (bc + cb)^* + (bbcc + bcbb + cbcb + cbcc + ccbb + cbcb + ccbb)^* \)
   (c) None of the above.

(iii) Consider a context-free grammar that has productions \( S \rightarrow \alpha \mid \beta \) where \( \alpha \) derives the empty string. Which of the following conditions always prevents the use of recursive-descent parsing with this grammar:
   (a) \( \text{first}(S) \cap \text{first}(\beta) \neq \emptyset \)
   (b) \( \text{follow}(S) \cap \text{first}(\beta) \neq \emptyset \)
   (c) \( \text{follow}(S) \cup \text{first}(\alpha) \neq \emptyset \)
   (d) None of the above conditions necessarily prevents the use of recursive-descent parsing.

(iv) Consider an inference rule for correctness statements: \( \frac{Q_1(C)P_1}{Q_1} \parallel \frac{Q_2(C)P_2}{Q_2} \).
   This inference rule is:
   (a) Generally valid.
   (b) Valid only if \( C \) has no assignments to variables used in the assertions \( P_1 \) and \( P_2 \).
   (c) Valid only if \( C \) does not interfere with variables used in the assertions \( Q_1 \) and \( Q_2 \).
   (d) None of the above.

(v) The Church-Turing thesis states that
   (a) The halting problem cannot be solved using programming languages (such as C) but it can be solved using Turing machines.
   (b) Turing machines cannot solve the halting problem.
   (c) Functional programming languages can implement certain functions that cannot be computed by Turing machines.
   (d) None of the above.

(vi) Consider algorithmic problems \( A \) and \( B \) such that \( A \) reduces to \( B \) and \( A \) is unsolvable. Then
   (a) \( B \) is always solvable.
   (b) \( B \) is always unsolvable.
   (c) It is possible that \( B \) is solvable and also possible that \( B \) is unsolvable.
   (d) Problems \( A \) and \( B \) satisfying the assumptions cannot exist.
8. Using the systematic method described in the course convert the state diagram below into an equivalent regular expression. The alphabet is $\Sigma = \{a, b, c, d\}$. Your answer should include the intermediate step(s) used in the construction.

$$\text{Regex: } (ac^*b)^*d \left( ac^*b(ac^*b)^*d \right)^*$$

9. Using the systematic method described in the course (subset construction), convert the below nondeterministic state diagram into a deterministic state diagram. Your answer should indicate how the deterministic state diagram is obtained from the nondeterministic one: the states of the deterministic diagram should be labeled by sets of states of the nondeterministic diagram.
Question 10 solution:

---

invariant I:
-1 <= i <= n-2 && ForAll(k=i+1; k<n) A[k] == (n-k)*(n-k+1)/2

ASSERT( 1 <= n < max )
// 1 <= n implies -1 <= n-2
// below ForAll reduces to
// (A| n-1 --> 1)[n-1] == (n-n+1)*n-n+1)/2, that is, 1 == 1
{ int i;
ASSERT(-1 <= n-2 <= n-2 && ForAll(k=n-1; k<n) (A| n-1 --> 1)[k] == (n-k)*(n-k+1)/2)
  i = n-2;
ASSERT(-1 <= i <= n-2 && ForAll(k=i+1; k<n) (A| n-1 --> 1)[k] == (n-k)*(n-k+1)/2)
  A[n-1] = 1;

ASSERT( I )

while( i >= 0 ) { ASSERT( I && i >= 0 )
// i >= 0 implies -1 <= i-1
// Invariant implies below ForAll-statement for values i+1 <= k < n
// because then (A | i --> A[i+1]+n-i)[k] == A[k].
// With value k=i, below we get
// (n-i-1)*(n-i-1+1)/2 + n - i == (n^2 - 2in + i^2 + n - i)/2
// == (n-i)*(n-i+1)/2 because by invariant
// A[i+1] == (n-i-1)*(n-i+1)/2
  ASSERT(-1 <= i-1 <= n-2 &&
         ForAll(k=i; k<n) (A | i --> A[i+1]+n-i)[k] == (n-k)*(n-k+1)/2)
  ASSERT(-1 <= i-1 <= n-2 && ForAll(k=i; k<n) A[k] == (n-k)*(n-k+1)/2)
  i = i-1;
  ASSERT(I)
}
//end while

ASSERT( I && i < 0 )
// -1 <= i && i < 0 implies i == -1
// with i == -1, ForAll in invariant becomes post-condition
ASSERT(ForAll(k = 0; k < n) A[k] == (n-k)*(n-k+1)/2)