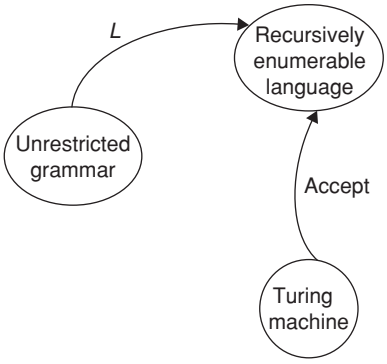
Chapter 3

Recursively Enumerable Sets and Turing Machines, Decidability

LEARNING OBJECTIVES

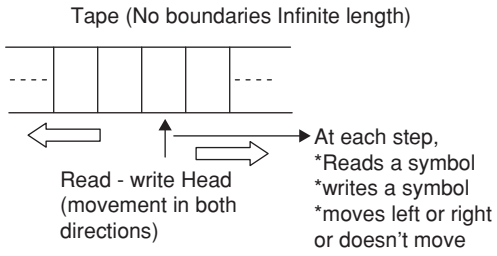
- *Turing machines*
- *Model of turing machine*
- *Types of turing machines*
- *Offline turing machine*
- *Universal turing machine*
- *Recursively enumerable languages*
- *Recursive language*
- *Undecidability*
- *Church's hypothesis*
- *Halting problem*
- *Post's correspondence problem*
- *7 P problems*
- *NP problems*
- *NP – complete problem*
- *NP – hard problem*
- *Closure properties of formal languages*

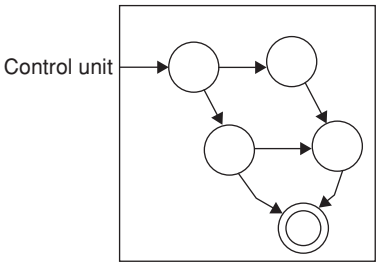
Turing Machines



A Turing machine is a kind of state machine, which is much more powerful in terms of languages it can recognize. At any time, the machine is in any one of the finite number of states. Instructions for a turing machine include the specification of conditions, under which the machine will make transitions from one state to other.

Model of Turing Machine

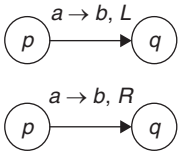




- A TM (turing machine) consists of Tape, Head, control unit.
- **Tape:** A tape is divided into a sequence of numbered cells, Each cell contains a symbol and cells that have not been written before are assumed to be filled with a blank symbol (B). The set of symbols of tape is denoted by Γ. The tape is assumed to be arbitrarily extensible to the left as well as to the right.
- **Head:** In a single step, a tape head reads the contents of a cell on the tape (reads a symbol), replaces it with some other characters (writes a symbol) and repositions itself to the next cell to the right or to the left of the one it has just read or does not move (moves left or right or does not move).
- **Control unit:** The reading from the tape or writing into the tape is determined by the control unit. It contains a finite set of states, Q. The states are:
  1. Initial state, q_0
  2. Halt state, h: This is state in which TM stops all further operations. There can be one or more halt states in a TM.
  3. Other states.

Note: A TM on entering the halt state stops making moves and whatever string is there on the tape, will be taken as the output, irrespective of whether the position of head is at the end or in the middle of the string on the tape.

Transition Diagram of TM



Reads a symbol
writes a symbol
Move Right (R)
Move Left (L)
No Move (N)

Specification of TM

5-Tuple specification:
TM = (state1, Read symbol, write symbol, $L/R/N$, state 2).
7 - Tuple specification of TM:
A TM, M is represented as a 7-tuple:
$M = (Q, \Sigma, \Gamma, \delta, q_0, B, h)$ where
$Q \rightarrow$ Finite set of states
$\Sigma \rightarrow$ Finite set of non-blank symbols
$\Gamma \rightarrow$ Set of tape characters
$q_o \rightarrow q_o \in Q$, initial state
$B \rightarrow$ Blank character
$h \rightarrow h \subseteq Q$, final state
$\delta \rightarrow$ Transition function, $Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, N\}$

String classes in TM

Every TM, over the alphabet Σ, divides set of input string w into three classes:

1. **Accept (TM):** It is the set of all strings $w \in \Sigma^*$ ∋ if the tape initially contains w and the TM is then run, then TM ends in a halt state.
2. **LOOP (TM):** It is the set of all strings, $w \in \Sigma^*$ ∋ if the tape initially contains w and the TM is then run, then the TM loops forever (infinite loop).
3. **Reject (TM):** It is the set of all strings $w \in \Sigma^*$ ∋, any of the following 3-cases arise.

Case I: There may be a state and a symbol under the tape head, for which δ does not have a value.

Case II: If the head is reading the left most cell (i) containing the symbol x, the state of TM is say q, then δ (q, x) suggests a move to the left of the current cell. However as there is no cell to the left, no move is possible.

Case III: If TM enters an infinite loop or if a TM rejects a given string w, because of above two cases, TM crashes (terminates unsuccessfully).

Languages Accepted by a TM

- The language accepted by TM is the set of accepted strings $w \in \Sigma^*$.
- Formally, let $M = (Q, \Sigma, \Gamma, \delta, q_0, B, h)$ be a TM. The language accepted by M denoted by $L(M)$ is defined as, $L(M) = \{w/w \in z^*$ and if $w = a_1 \ldots a_n$ then, $(q_o, \varepsilon, a_1, a_2, \ldots a_n)$ $(h, b, \ldots, b_{i-1}, b_j\, b_n)$ for some $b_1, b_2 \ldots b_n \in N^*$ ∋$\}$

  $L(M) = \{W: q_o\, w \vdash^* x_1 h x_2\}$
- There are three types of turing machine related languages:
  1. **Turing Acceptable language:** A language, L over some alphabet is said to be turing acceptable language if there exists a TM, M ∋ $L = L(M)$
  2. **Turing Decidable Language:** A language L over Σ i.e., $L \subseteq \Sigma^*$ is said to be turing decidable, if both languages, L and its complement $\Sigma^* - L$ are turing acceptable.
  3. **Recursively Enumerable Language:** A language L is recursively enumerable, if it is accepted by a TM.

Example 1: Let M be a turing machine has $M = (Q, \Gamma, \Sigma, \delta, S, B, F)$ with $Q = \{q_o, q_1, q_2, q_3, q_4\}$, $\Gamma = \{a, b, X, Y, \#\}$, $\Sigma = \{a, b\}$, $S = q_0$, $B = \#$, δ given by:

	a	*b*	*X*	*Y*	#
q_0	(q_1, X, R)	–	–	(q_3, Y, R)	
q_1	(q_1, a, R)	(q_2, Y, L)	–	(q_1, Y, R)	
q_2	(q_2, a, L)	–	(q_0, X, R)	(q_2, Y, L)	
q_3	–	–	–	(q_3, Y, R)	$(q_4, \#, R)$
q_4	–	–	–	–	–

Which of following is true about M?
(A) M halts on L having 'baa' as substring
(B) M halts on L having 'bab' as substring
(C) M halts on $L = \{a^n\, b^n / n \geq 1\}$
(D) M halts on L not having 'bbaa' as substring.

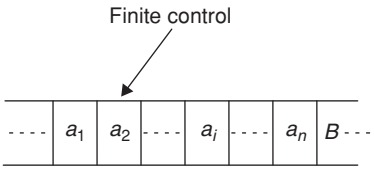
Solution: (C)
M accepts $a^n b^n$.

Example: *aaabbb*

$(q_0, \in, aaabbb) \rightarrow (q_1, XXXYY, b)$
$(q_1 X, aabbb) \rightarrow (q_2 XXXY, YY)$
$(q_1, Xa, abbb) \rightarrow {}^1(q_2, XXX, YYY)$
$(q_1, Xaa, bbb) \rightarrow (q_2, XY, XYYY)$
$(q_1, Xa, aYbb) \rightarrow (q_0, XXX, YYY)$
$(q_2, X, aaYbb) \rightarrow (q_3, XXXY, YY)$
$(q_2, \in, XaaYbb) \rightarrow (q_3, XXXYY, Y)$
$(q_o, X, aaYbb) \rightarrow (q_3, XXXYYY, \in)$
$(q_1, XX, aYbb) \rightarrow (q_4, XXXYYY\#, \in)$

Types of Turing Machines

Two-way infinite turing machine

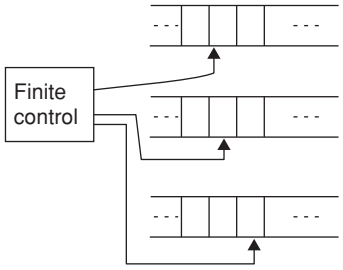


- A TM with a two-way infinite tape is denoted by $M = (Q, \Sigma, \Gamma, \delta, q_o B, F)$, as in original model.
- The tape is infinite to the left as well as to the right.

If $\delta(q, x) = (p, Y, L)$ then $q\ x\ \alpha \vdash_m pBY$. The tape, is infinte towards left.

If $\delta(q, x) = (p, B, R)$ then $q\ x\ \alpha \vdash_m p\alpha$ the is infinite towards right.

Multiple turing machines



- A multiple TM consists of a finite control with k tape heads and k-tapes, each tape is infinite in both directions, on a single move, depending on the state of the finite control and the symbol scanned by each of tape heads, the machine can,
  - change state
  - print new symbol on each of the cells scanned by its tape head
  - move each of its tape heads, independently, one cell to the left or right or keep it stationary.
- Initially, the input appears on the first tape and other tapes are blank.

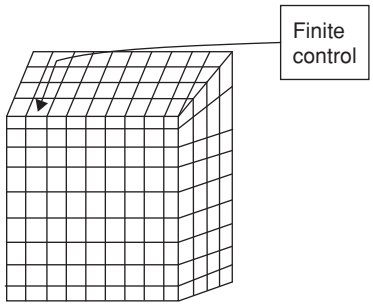
Non-deterministic turing machines

- A non-deterministic turing machine is a device with a finite control and a single one way infinite tape.
- For a given state and a tape symbol scanned by the tape head, the machine has a finite number of choices for next move.

Note: Non-deterministic TM is not permitted to make a move in which the next state is selected from one choice, and the symbol printed and direction of head motion are selected from other choices.

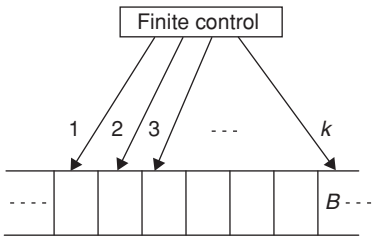
- The non-deterministic TM accepts its input if any sequence of choices of moves leads to an accepting state.

Multi-dimensional TM's



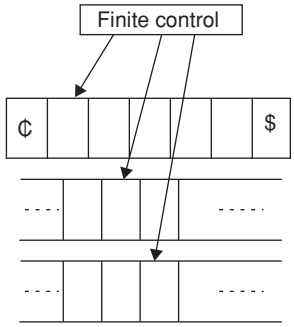
- The tape consists of a k-dimensional array of cells infinite in all 2k directions, for some fixed k.
- Depending on the state and the symbol scanned, the device changes it's state, prints a new symbol and moves its tape head in one of the $2k$ directions, either positively or negatively, along one of the k-axes.

Multihead TM



- A K-head TM has some fixed 'K' number of heads. The heads are numbered from 1 through k, and a move of the TM depends on the state and on the symbol scanned by each head.

Offline turing machine



- An offline TM is a multi tape TM, whose input tape is read only. The input is surrounded by end markers, ¢ on left and $ on right. The TM is not allowed to move the input tape head off the region between ¢ and $.

Multi stack machine

- A deterministic two stack machine is a deterministic TM with a read only input and two storage tapes.

Note:

- All these types of TM's does not add any language accepting power and all these are equivalent to the basic model.
- Any language accepted by a 2-PDA can be accepted by some TM and any language accepted by a TM can be accepted by some 2-PDA. Accepting power of a TM = accepting power of a computer.
- Any language accepted by a PDA with n stacks ($n \geq 2$), can also be accepted by some TM.

Example 2: Consider the following statement about L:

1. L is accepted by multi-tape turing machine M_1.
2. L is also accepted by single tape turing machine M_2.

Which of following statement is correct?

(A) Acceptance by M_2 is slower by $O(n^2)$
(B) Acceptance of M_2 is slower by $O(n)$
(C) Acceptance of M_2 is faster by $O(n)$
(D) Acceptance of M_2 is faster by $O(n^2)$

Solution: (A)
While simulating multi-tape TM on a single tape TM the head has to move at least $2k$ cells per move, where k is the number of tracks on single tape TM. Thus for k moves, $\sum_{i=1}^{k} 2i = 2k^2$.
Which means quadratic slow down?
Thus, acceptance of multi-tape is faster by $O(n^2)$.

Universal turing machine

A Universal turing machine is a turing machine that can simulate an arbitrary turing machine on arbitrary input.

- The machine consists of an input output relation to the machine computes.
- The input is given in binary form on the machine tape and the output consists of the contents of the tape when the machine halts.
- The contents of the tape will change based on the Finite State Machine (FSM) inside the TM.
- The problem with TM is that a different machine will be constructed for every new computation to be performed.
- A UTM can simulate any other machine.

Combining turing machines

If TM_1 and TM_2 are turing machines, then we can combine these machines and create a Turing machine which will first behave like TM_1 and TM_2.

To combine two turing machines follow below steps:

1. Change all states in TM_2, so that they do not conflict with the state names in TM_1.
2. Change all halts in TM_1's transition table to the new name of the start state of TM_2.
3. Append TM_2's transition table to the foot of TM_1's transition table.

- If TM_1 and TM_2 are combined in this way, we will write it as $TM_1 \rightarrow TM_2$.

So this new machine starts off in the initial state of TM_1, operates as per TM_1 until TM_1 would halt then it launches TM_2 and operates a TM_2, until TM_2 would halt.

Recursively Enumerable Languages

- A language L over the alphabet Σ is called 'recursively enumerable' if there is a TM, M that accept every word in L and either rejects or loops for every word in language L', the complement of L.
  Accept $(M) = L$
  Reject (M) + Loop $(M) = L'$.
- When TM, M is still running on some input of recursively enumerable languages, it is not decided that M will eventually accept, if let it run for long time or M will run forever (in loop).

Recursive language

- A language is said to be recursive, if there exists a TM which will halt and accept when presented with any input string $w \in \Sigma^*$, only if the string is in the language otherwise will halt and reject the string.
- Thus, for turing decidable language L, there is a TM which halts for a large number of inputs w belonging to L.
- A TM that always halts is known as a decider or a total turing machine and is said to decide the recursive language. The recursive language is also called as recursive set of decidable.
- A language accepted by a TM is said to be recursively enumerable language. The subclass of recursively enumerable sets are said to be recursive sets or recursive language.

Note:

- All recursive languages are also recursively enumerable.
- There may be languages which are recursively enumerable but not recursive.
- Set of all possible words over the alphabet of the recursive language is a recursive set.

- Set of all possible words, over the alphabet of the recursive enumerable language, is a recursively enumerable set.

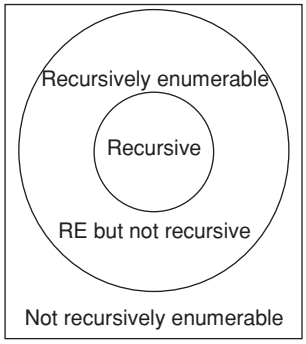


Figure 1 Relationship between the recursive, RE and non-RE languages.

Properties of Recursive and Recursively Enumerable Languages

- If a language L is recursive, then there is a TM T that accepts it and always halts.
- If L and L_1 are both recursively enumerable, then L and L_1 are recursive.
- Union of two recursive languages is recursive.
- Recursively enumerable languages are closed under union.
- If L, L_1 and L_2 are recursive languages, then so are $L_1 \cup L_2$, $L \cap L_2$, L_1L_2, L^*, $L_1 \cap L_2$ and $L_1 - L_2$.
- If L, L_1 and L_2 are recursively enumerable languages, then so are $L_1 \cup L_2$, L^*, $L_1 \cap L_2$, L_1L_2.
- If Σ is an alphabet, $L \subseteq \Sigma^*$, is a recursively enumerable language and $\Sigma^* - L$ is recursively enumerable, then L is recursive.

Example 3: If $\Sigma = \{0,1\}$, the canonical order is $\{\in,0,1,00,01,10,11,000,\ldots\}$ where w is the i^{th} word and M_j is TM whose code is the integer j, written in binary. The language generated is $L(M_j)$. The diagonalized language, L_d is a.

(A) Recursively enumerable language but not recursive
(B) Recursive language
(C) Non-recursively enumerable language
(D) Both (a) and (c)

Solution: (C)
Non-recursively enumerable language.

Non-recursively enumerable language

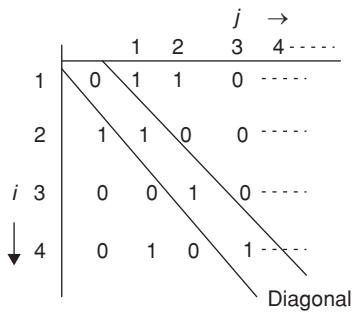
Non-Recursively Enumerable Language: A language which is not accepted by any turing machine is non-recursively enumerable.

Example: Power set of an infinite set.

- These languages cannot be defined by any effective procedure.

For any non-empty Σ, there exist languages that are not Recursively Enumerable.

Infinite table for all i and j is:



To guarantee that no TM accepts L_d:

w_i is in L_d if and only if the (i, i) entry is 0, that is, if M_i does not accept w_i.

Suppose that some TM M_j accepted L_d. Then it contradicts if w_j is in L_d, (j, j) entry is 0, implying that w_j is not in $L(M_i)$ and contradicting $L_d = L(M_i)$.

If w_i is not in L_d, then the (j, j) entry is 1, implying that w_i is in $L(M_j)$, which again contradicts $L_d = L(M_j)$, as w_j is either in or not in L_d, assumption, $L_d = L(M_j)$ is false.

Thus no TM in the list accepts L_d, Hence L_d is non-recursively enumerable language.

Decidable: A problem with two answers (Yes/No) is decidable if the corresponding language is recursive.

Example:

1. $A_{DFA} = \{(M, w)\ M$ accepts the input string $w\}$.

- A Language L is turing decidable, if there exists a TM M such that on input x, M accepts if $x \in L$ and M rejects otherwise. L is called undecidable if it is not decidable.
- Decidable Languages correspond to algorithmically solvable Decision problems.
- Undecidable language corresponds to algorithmically unsolvable decision problems.

Closure properties of decidable languages

- Decidable Languages are closed under complement, union, intersection, concatenation and star (closure) operations.

Note 1: A language is decidable if both the language and its complement are recognizable.

Note 2: Turing Decidable languages are Recursive languages.

Undecidability

There are problems that can be computed. There are also problems that cannot be computed. These problems which cannot be computed are called 'computationally undecidable problems'.

Church's Hypothesis

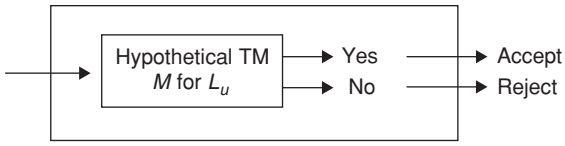
There is an assumption that the intuitive notion of computable functions can be identified with partial recursive functions.

However, this hypothesis cannot be proved. The computability of recursive function is based on following assumptions:

1. Each elementary function is computable.
2. Let 'f' be a computable function and 'g' be another function which can be obtained by applying an elementary operation to f, then g becomes a computable function.
3. Any function becomes computable, if it is obtained by rule (1) and (2).

Undecidability of the universal languages

- The universal language, L_u is a recursively enumerable language but not recursive.



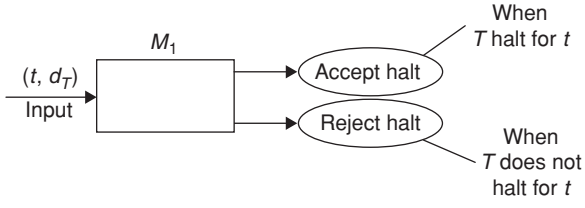
Halting Problem

The given configuration of TM is required to state halting problem. The output of TM can be:

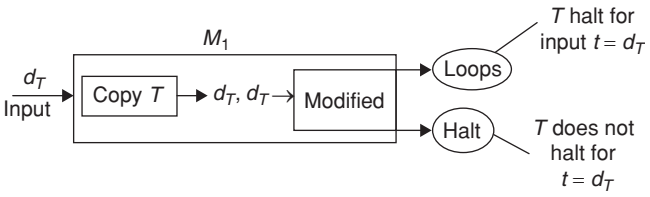
1. **Halt:** The machine starting at this configuration will halt after a finite number of states.
2. **No Halt:** The machine starting at this configuration never reaches a halt state, no matter how long it runs.

- The halting problem is unsolvable because, let, there exists a TM, M, which decides whether or not any computation by a TM, T will ever halt when a description d_T of T and tape t of T is given. That means the input to machine M, will be (machine, tape) pair. Then for every input (t, dT) to M_I if T halt for input t, M_I also halts which is called accept halt.

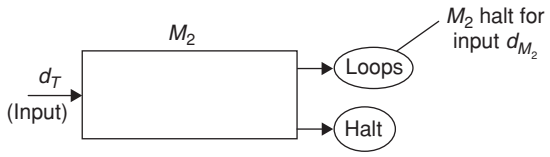
Similarly if T does not halt for input t then the M_1 will halt which is called reject halt.



- Consider another Turing Machine, M_2 which takes an input d_T. It first copies d_T on its tape and then this duplicated tape information is given as input to M_1. But M_1 is a modified machine.



Replace T by M_2 i.e., $M_2 = \mathrm{T}$



That's means, a machine M_1, which can tell whether any other TM will halt on particular input does not exist. Hence halting problem is unsolvable.

Post's Correspondence Problem (PCP)

The Undecidability of strings is determined with the help of Post's Correspondence Problem (PCP).

'The PCP consists of two lists of strings that are of equal length over the input Σ. The two lists are $A = w_1, w_2, w_3, \ldots w_n$ and $B = x_1, x_2, \ldots x_n$ then there exists a non-empty set of integers $i_1, i_2, \ldots i_n$ such that $w_1, w_2, \ldots w_n = x_1, x_2, \ldots x_n$'.

To solve PCP, try all the combinations of $i_1, i_2, \ldots i_n$ to find the $w_i = x_i$ then, PCP has a solution.

Example 4: What is the solution for the following system of post correspondence problem. $A = \{100, 0, 1\}$ $B = \{1, 100, 00\}$

(A) 1113322 (B) 1311322
(C) 2233111 (D) No solution

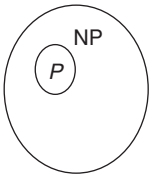
Solution: (B)
The string is:
A1A3A1A1A3A2A2 = 100 + 1 + 100 + 100 + 1 + 0 + 0 = 1001100100100,
B1B3B1B1B3B2B2 = 1 + 00 + 1 + 1 + 00 + 100 + 100 = 1001100100100.

Problems

- P stands for deterministic polynomial time. A deterministic machine at each time executes an instruction. Depending on instruction, it then goes to next state which is unique. Hence, time complexity of deterministic TM is the maximum number of moves made by M in processing an input string of length n, taken over all inputs of length n.
- A language, L is said to be in class P, if $\exists$ a (deterministic) TM, M is of time complexity $P(n)$ for some polynomial P and M accepts L.
- Class P consists of those problems that are solvable in polynomial time by a deterministic TM.

NP Problems

- NP stands for non-deterministic polynomial time.
- A language, L is in class NP, if there is a non-deterministic TM, M is of time complexity $P(n)$ for some polynomial P and M accepts L.
- Class NP consists of problems for which solutions are verified quickly. P consist of problems which can be solved quickly.



- NP languages are closed under union, Intersection, concatenation, Kleen star.
- NP problems are classified into two types:
  1. NP-complete
  2. NP-hard problems.

Example: Vertex (Graph) coloring problem, Travelling salesman problem, the vertex cover problem, the Hamiltonian circuit problem.

NP-Complete Problem

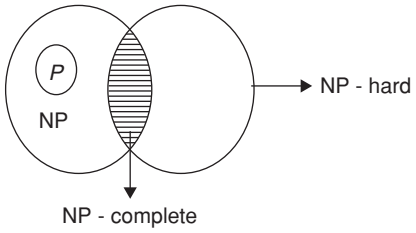
- A class of problems are known as NP-complete problems whose status is unknown. No polynomial time has yet been discovered for NP-complete problems nor has any one been able to prove that no polynomial time exists for any of them. These are hardest of NP-problems. The P and NP-complete problems are disjoint.

Example: (Cook's Theorem) SAT is NP-complete, Bin packing problem, Knapsack Problem.

- A language L is said to be NP-complete if $L \in NP$ and if every $L^I \in$ NP is polynomial-time reducible to L.

A language L_1 is said to be polynomial time reducible to some language L_2 if there exists a DTM by which any w_1 in the alphabet of L_1 can be transformed in polynomial time to a w_2 in the alphabet of L_2 in such a way that $w_1 \in L_1$ if $w_2 \in L_2$. It follows that if some L_1 is NP-complete and polynomial time reducible to L_2, then L_2 is also NP-complete.

NP-Hard Problem



- A problem that is NP-hard has a property that all problems that are in NP can be reduced in polynomial time to it.
- A language, L in NP-hard complete if and only if,

Condition 1: For every language, L^I in NP, there is a polynomial time reduction of L^I to L.

Condition 2: L is not necessarily in NP.

Table 1 *NP-Hard versus NP-complete problems:*

NP-Hard	NP-Complete
(1) A decision problem P_i is NP-hard if every problem in NP is polynomial time reducible to P_i.	(1) A Decision problem P_i is NP-complete if it is NP-hard and is also in class NP itself.
(2) In terms of symbols 'P_i' is NP-hard if for every $P_j \rightarrow$ NP	(2) In terms of symbols, 'P_i' is NP-complete, if P_i is NP-hard and $P_j \rightarrow$ NP
(3) P_i is 'as hard as' all the problem in NP	(3) P_i is one of the hardest problems in NP
(4) If any problem in NP is proved intractable, then P_i must also be intractable	(4) If any one ever shows that as NP-complete problem is also intractable, then every NP-complete problem is also intractable.

Example 5: Which of following is FALSE?

(A) $\{< x, y > \mid x$ and y are integers, gcd $(x, y) = 1\}$ is a NP class problem.
(B) CLIQUE is a NP class problem.
(C) Eulerian PATH is a P class problem
(D) Dijkstra's algorithm is a problem in P.

Solution: (A)
Choice (A) is a P class problem.
Consider the following table:

D – Decidable, U – Undecidable,? – Open question,T – Trivially Decidable Question	Regular Sets	DCFL's	CFL's	CSL's	Recursive Sets	Recursively Enumerable Sets
(1) Membership problem?	D	D	D	D	D	D
(2) Emptiness problem?	D	D	D	U	U	U
(3) Completeness problem is $L = \Sigma^*$?	D	D	D	U	U	U
(4) Equality problem?	D	?	U	U	U	U
(5) Subset problem is $L_1 \subseteq L_2$?	D	U	U	U	U	U
(6) Is L Regular?	T	D	U	U	U	U
(7) Is the intersection of two languages, a language, of the same type?	T	U	U	T	T	T
(8) Is the complement of a language, also a language of the same type?	T	T	U	?	T	U
(9) Is L is finite or infinite?	D	D	D	U	U	U

Table 2 *Closure properties of formal languages*

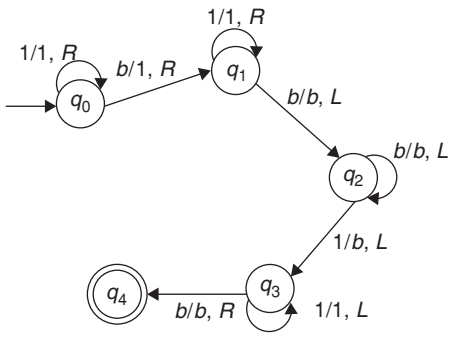
	Regular sets	DCFL'S	CFL'S	CSL'S	Recursive sets	Recursively enumerable sets
(1) Union	*Y*	*N*	*Y*	*Y*	*Y*	*Y*
(2) Concatenation	*Y*	*N*	*Y*	*Y*	*Y*	*Y*
(3) Kleen star	*Y*	*N*	*Y*	*Y*	*Y*	*Y*
(4) Intersection	*Y*	*N*	*N*	*Y*	*Y*	*Y*
(5) Complementation	*Y*	*Y*	*N*	*Y*	*Y*	*N*
(6) Homomorphism	*Y*	*N*	*Y*	*N*	*N*	*Y*
(7) Inverse Homomorphism	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*
(8) Reversal	*Y*	*N*	*Y*	*Y*	*Y*	*Y*
(9) Substitution	*Y*	*N*	*Y*	*Y*	*N*	*Y*
(10) Intersection with regular ets	*Y*	*Y*	*Y*	*Y*	*Y*	*Y*

EXERCISES

Practice Problems I

Directions for questions 1 to 15: Select the correct alternative from the given choices.

1. The TM M over $\Sigma = \{1\}$ is given below



What does M generate?

(A) The output is total recursive multiplication function.
(B) The output is addition of two integers.
(C) The output is subtraction of two integers.
(D) The output should be w_1w_2 if input = (w_1w_2) a pair of words.

2. Consider language,

$A = \{<M>: M$ is a DFA which doesn't accept any string containing odd number 1's$\}$

Which of following is true about A?

(A) A is Trivially decidable (B) A is undecidable
(C) A is decidable (D) None of these

3. Consider $EQ_{CFG} = \{<G_1G_2>: G_1, G_2$ are CFGs and $L(G_1) = L(G_2)\}$. Which of following is true about EQ_{CFG}?

(A) Recognizable (B) Co-Recognizable
(C) Un-recognizable (D) None of the above.

4. A language is given as $INFINITE_{DFA} = \{<A>: A$ is a DFA and $L(A)$ is an infinite language$\}$. Which of following is true?

(A) Un-decidable (B) Decidable
(C) Trivially decidable (D) None of above.

5. A TM designed over an alphabet {0, 1, #}, where 0 indicates blank, which takes a non-null string of 1's and #'s and transfer's the right-most symbol to the left-most end contains-states. (Ex: 000#1#1#1000 … becomes 0001#1#1#000)

(A) 4 (B) 3
(C) 6 (D) 5.

6. Which of following statements are true?

(i) Let K, L be decidable languages. The concatenation of languages, K, L is also decidable language.

(ii) Let L be Turing recognizable language. Then the complement, L^1 is also Turing recognizable language.

(A) (i) and (ii) (B) Only (ii)
(C) Both are false (D) Only (i)

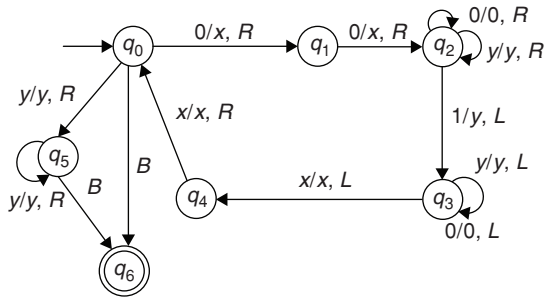
7. Let T_i: denote i th TM. Given, X determines whether $X \in S$, Where the set S is defined inductively as follows: If $u \in S$, then $u^2 + 1$, $3u + 2$ and $u!$ are all members of S. Which of following is true about the given decision problem?

(A) Decidable (B) Un-decidable
(C) Trivially decidable (D) No solution.

8. Fermat's last theorem asserts that there are no integer solution (x, y, z, n) to equation $x^n + y^n = z^n$ satisfying $x, y > 0$ and $n > 2$. Which of the following is true regarding the halting problem?

(A) Decidable
(C) Un-decidable
(C) Trivially decidable
(D) May or may not have solution.

9. The TM, T is designed as



Which of following is true?

(A) T halts on 0^n1^n, $n \geq 0$

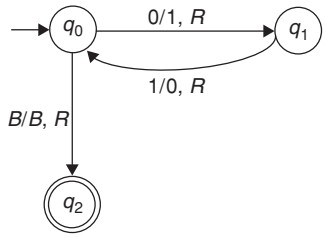
(B) T halts on (01) $(0^n 1^n)$, $n \geq 0$

(C) T halts on $0^{n^2}1^{n^2}$, $n \geq 0$

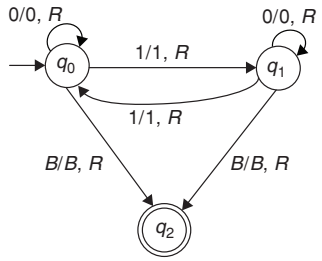
(D) T halts on $0^{2n}1^n$, $n \geq 0$

10. Design TM, which reads an input and starts inverting 0's to 1's till the first 1. The first 1 also inverted. After it has inverted first 1, it read the next symbols and keeps them as they are till the next 1. After encountering 1, it starts repeating the cycle by inverting the symbol till next 1. It halts when it encounters a blank symbol?

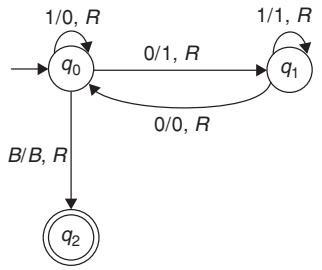
(A)



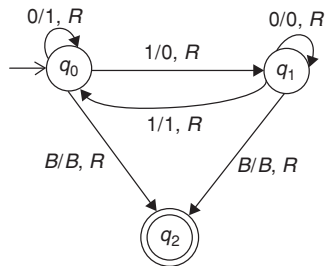
(B)



(C)



(D)



11. Consider three problems, P_1, P_2 and P_3. It is known that P_1 has polynomial time solution, P_2 is NP-complete and P_3 is in NP. Which one of the following is true?

(A) P_3 has polynomial time solution if P_1 is polynomial time reducible to P_3.

(B) P_3 is NP-complete if P_3 is polynomial time reducible to P_2.

(C) P_3 is NP complete if P_2 is reducible to P_3

(D) P_3 has polynomial time complexity and P_3 is reducible to P_2.

12. Let FHAM be the problem of finding a Hamiltonian cycle in a graph G and DHAM be the problem of determining if a Hamiltonian cycle exists in a graph. Which one of the following is true?

(A) Both FHAM and DHAM are NP-hard.

(B) FHAM is NP-hard, but DHAM is not.

(C) DHAM is NP-hard but FHAM is not.

(D) Neither DHAM nor FHAM is NP-hard.

13. The solution for the system of post correspondence problem, $A = \{ba, abb, bab\}$, $B = \{bab, bb, abb\}$ is

(A) 1312212 (B) 15234434

(C) 1311322 (D) No solution.

14. A language, prefix_free REX = $\{R/R$ is a regular expression where $L(R)$ is prefix_free$\}$. Which of following is true about prefix _free REX?

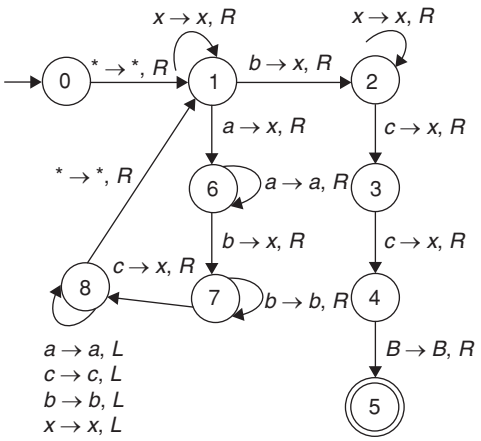
(A) Decidable

(B) Un-decidable

(C) Trivially decidable.

(D) Can't be determined.

15. The TM, M is designed as:



Which of following is true about M?

(A) M is designed for $a^n b^n c^n$, $n \geq 0$

(B) M is designed for $a^{n^2} b^{n^3} c^{n^4}$, $n \geq 0$

(C) M is designed for $a^n b^{n+1} c^{n+2}$, $n \geq 0$

(D) M is designed for $a^n b^n c^n$, $n > 0$

Practice Problems 2

Directions for questions 1 to 15: Select the correct alternative from the given choices.

1. Consider the language, $A\varepsilon_{\overline{CFG}} = \{<G>: G \text{ is a CFG that generates} \varepsilon\}$. Which of the following is true?
(A) Undecidable
(B) Decidable
(C) Trivially decidable.
(D) None of the above.

2. The TM is designed with input and output as binary form. (# represents blank). The turing machine TM (M) is

	0	1	#
$q0$	$(q1, 0, R)$	$(q, 1, R)$	ϕ
$q1$	$(q1, 0, R)$	$(q, 1, R)$	$(q2, \#, L)$
$q2$	$(q3, \#, L)$	$(q3, \#, L)$	ϕ
$q3$	$(q3, 0, L)$	$(q3, 1, L)$	$(q4, \#, L)$
$q4$	ϕ	ϕ	ϕ

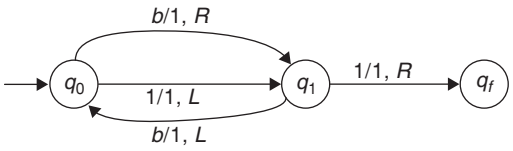
Which of following is true?
(A) M accepts $2n$
(B) M accepts n^2
(C) M replaces left most symbol with #
(D) M replaces right most symbol with #

3. The TM is designed with 3-characters 0, 1, # to compute function $f(n) = 2n$. Input and output are to be in binary form and string represented by 'n' is enclosed between two #'s on left and right of it. b is blank symbol. TM contains _____ states.
(A) 4 (B) 3
(C) 2 (D) 1

4. The language $\{1^n \mid n \text{ is a prime number}\}$ is
(A) Undecidable
(B) Decidable
(C) Trivially decidable
(D) None of the above

5. Which of following statement(s) are true?
(i) Let L be Turing decidable language. Then the complement $\overline{L}$ is also Turing decidable language.
(ii) Let K and L be two Turing recognizable languages. The intersection, $K \cap L$ is also Turing recognizable language.
(A) Both (i) and (ii)
(B) Only (i)
(C) Only (ii)
(D) Neither (i) nor (ii) are true.

6. For the following two-way infinite TM, the equivalent one-way TM contains _____ states.



(A) 7 (B) 6
(C) 5 (D) 4

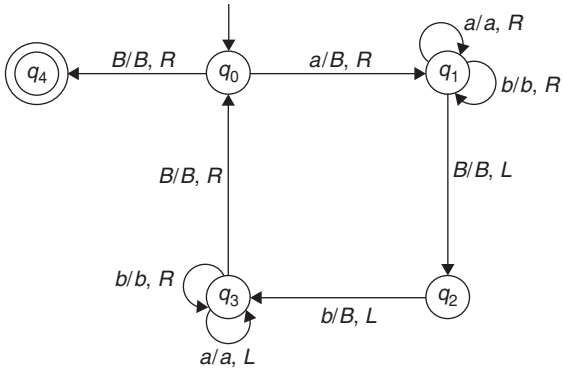
7. L contains at least two strings. Which of following is true?
(A) L has recursively enumerable sets and recursive.
(B) L is recursive.
(C) L has recursively enumerable sets but not recursive.
(D) L does not contain recursively enumerable sets and also is not recursive.

8. Consider the following TM:

Input / State	0	1	B
→ q_0	$(q0, 1, R)$	$(q0, 0, R)$	$(q1, B, R)$
$q1$	–	–	–

What does TM generates?
(A) It display's the negative of given binary number.
(B) It computes one's complement of a binary number.
(C) It computes two's complement of a binary number
(D) It generates double the 0's as 1's.

9. Consider the following TM, M:



Which of following is true?
(A) M halts on $a^{n+1}\, b^n$, $n \geq 0$.
(B) M halts on a^{n^2}, b^{n^3} $n \geq 0$.
(C) M halts on $(ab)\,(a^n)$, $n \geq 0$.
(D) M halts on $a^n\, b^n$, $n \geq 0$.

10. A TM, M is designed generates language $L = \{a^n b^m : n \geq 1 \text{ and } n \neq m\}$. The number of states used are _____
(A) 5 (B) 6
(C) 7 (D) 4

11. Consider three decision problems p_1, p_2 and p_3. It is known that p_1 is decidable, p_2 is undecidable. Which one of following is true?
(A) p_3 is decidable if p_1 is reducible to p_3
(B) p_3 is undecidable if p_3 is reducible to p_2
(C) p_3 is undecidable if p_2 is reducible to p_3
(D) p_3 is decidable if p_3 is reducible to p_2's complement.

12. Which one of following is not decidable?
(A) Given a TM, M, a string S, and an integer K, M accepts S with in K-steps.
(B) Equivalence of two given Turing machines.
(C) Language accepted by a given DFSA is non-empty.
(D) Language accepted by a CFG is non-empty.

13. What is the solution for the correspondence system with two lists $x = \{b, bab^3, ba\}$ and $y = \{b^3, ba, a\}$
(A) 1312213 (B) 2113
(C) 3112 (D) No solution.

14. Given a Turing machine M, a state 'q'and a string 'w'. To determine whether M ever reaches state q when started with input w from its initial state is?
(A) Decidable
(B) Un-decidable
(C) Trivially decidable.
(D) Can not be determined.

15. Given a Turing machine, M to determine whether M ever moves its head to the left when started with input W is:
(A) Decidable
(B) Un-decidable
(C) Trivially decidable.
(D) Can not be determined.

Previous Years' Questions

1. For $s \in (0 + 1)^*$, let $d(s)$ denote the decimal value of s (e.g., $d(101) = 5$). **[2006]**

Let $L = \{s \in (0 + 1)^* | d(s) \bmod 5 = 2 \text{ and } d(s) \bmod 7 \neq 4\}$

Which one of the following statements is true?
(A) L is recursively enumerable, but not recursive
(B) L is recursive, but not context-free
(C) L is context-free, but not regular
(D) L is regular

2. Which of the following is true for the language $\{a^p \mid p$ is a prime$\}$? **[2008]**
(A) It is not accepted by a Turing Machine
(B) It is regular but not context-free
(C) It is context-free but not regular
(D) It is neither regular nor context-free, but accepted by a Turing machine

3. If L and $\overline{L}$ are recursively enumerable then L is **[2008]**
(A) regular
(B) context-free
(C) context-sensitive
(D) recursive

4. Let $L = L_1 \cap L_2$, where L_1 and L_2 are languages as defined below:

$L_1 = \{a^m b^m c a^n b^n \mid m, n \geq 0\}$

$L_2 = \{a^i b^j c^k \mid i, j, k \geq 0\}$

Then L is **[2009]**
(A) Not recursive
(B) Regular
(C) Context free but not regular
(D) Recursively enumerable but not context free.

5. Let L_1 be a recursive language. Let L_2 and L_3 be languages that are recursively enumerable but not recursive. Which of the following statements is not necessarily true? **[2010]**
(A) $L_2 - L_1$ is recursively enumerable
(B) $L_1 - L_3$ is recursively enumerable
(C) $L_2 \cap L_1$ is recursively enumerable
(D) $L_2 \cup L_1$ is recursively enumerable

6. Which of the following statements is/are FALSE? **[2013]**
1. For every non-deterministic Turing machine, there exists an equivalent deterministic Turing machine.
2. Turing recognizable languages are closed under union and complementation.
3. Turing decidable languages are closed under intersection and complementation.
4. Turing recognizable languages are closed under union and intersection.

(A) 1 and 4 only (B) 1 and 3 only
(C) 2 only (D) 3 only

7. Let L be a language and $\overline{L}$ be its complement. Which one of the following is NOT a viable possibility? **[2014]**
(A) Neither L nor $\overline{L}$ is recursively enumerable (*r. e*)
(B) One of L and $\overline{L}$ is r.e. but not recursive, the other is not r. e.
(C) Both L and $\overline{L}$ are r.e. but not recursive
(D) Both L and $\overline{L}$ are recursive

8. Let $A \leq_m B$ denotes that language A is mapping reducible (also known as many-to-one reducible) to language B. Which one of the following is FALSE? **[2014]**
(A) If $A \leq_m B$ and B is recursive then A is recursive.
(B) If $A \leq_m B$ and A is undecidable then B is undecidable.
(C) If $A \leq_m B$ and B is recursively enumerable then A is recursively enumerable.

(D) If $A \leq_m B$ and B is not recursively enumerable then A is not recursively enumerable.

9. Let <M> be the encoding of a Turing machine as a string over $\Sigma = \{0, 1\}$. Let $L = \{<M>|M$ is a *Turing machine that accepts a string of length* 2014$\}$. Then, L is

(A) Decidable and recursively enumerable
(B) Undesirable but recursively enumerable
(C) Undesirable and not recursively enumerable
(D) Decidable but not recursively enumerable

10. For any two languages L_1 and L_2 such that L_1 is context-free and L_2 is recursively enumerable but not recursive, which of the following is/are necessarily true? **[2015]**

I. $\overline{L}_1$ (complement of L_1) is recursive
II. $\overline{L}_2$ (complement of L_2) is recursive
III. $\overline{L}_1$ is context-free
IV. $\overline{L}_1 \cup L_2$ is recursively enumerable

(A) I only (B) III only
(C) III and IV only (D) I and IV only

11. Consider the following statements.

I. The complement of every Turning decidable language is Turing decidable.
II. There exists some language which is in NP but is not Turing decidable.
III. If L is a language in NP, L is Turing decidable.

Which of the above statements is/are true? **[2015]**

(A) Only II (B) Only III
(C) Only I and II (D) Only I and III

12. Let X be a recursive language and Y be a recursively enumerable but not recursive language. Let W and Z be two languages such that $\overline{y}$ reduces to W, and Z reduces to $\overline{x}$ (reduction means the standard many-one reduction). Which one of the following statements is **TRUE?** **[2016]**

(A) W can be recursively enumerable and Z is recursive.
(B) W can be recursive and Z is recursively enumerable.
(C) W is not recursively enumerable and Z is recursive.
(D) W is not recursively enumerable and Z is not recursive.

13. Consider the following types of languages: L_1: Regular, L_2: Context - free, L_3: Recursive, L_4: Recursively enumerable. Which of the following is / are TRUE? **[2016]**

I. $\overline{L}_3 \cup L_4$ is recursively enumerable
II. $\overline{L}_2 \cup L_3$ is recursive
III. $L^*_1 \cap L_2$ is context - free
IV. $L_1 \cup \overline{L}_2$ is context - free

(A) I only (B) I and III only
(C) I and IV only (D) I, II and III only

14. Consider the following languages. **[2016]**

$L_1 = \{<M> \mid M$ takes at least 2016 steps on some input$\}$,

$L_2 = \{<M> \mid M$ takes at least 2016 steps on all inputs$\}$ and

$L_1 = \{<M> \mid M$ accepts $\varepsilon\}$

where for each Turing machine M, <M> denotes a specific encoding of M. Which one of the following is TRUE?

(A) L_1 is recursive and L_2, L_3 are not recursive
(B) L_2 is recursive and L_1, L_3 are not recursive
(C) L_1, L_2 are recursive L_3 is not recursive
(D) L_1, L_2, L_3 are recursive

15. Let A and B be finite alphabets and let # be a symbol outside both A and B. Let f be a total function from A^* to B*. We say f is *computable* if there exists a turning machine M which given an input x in A^*, always halts with $f(x)$ on its tape. Let L_f denote the language $\{x \# f(x) \mid x \in A^*\}$. Which of the following statements is true: **[2017]**

(A) f is computable if and only if L_f is recursive.
(B) f is computable if and only if L_f is recursively enumerable.
(C) If f is computable then L_f is recursive, but not conversely.
(D) If f is computable then L_f is recursively enumerable, but not conversely.

16. Let $L(R)$ be the language represented by regular expression R. Let $L(G)$ be the language generated by a context free grammar G. Let $L(M)$ be the language accepted by a Turing machine M. Which of the following decision problems are undecidable? **[2017]**

I. Given a regular expression R and a string w, is $w \in L(R)$?
II. Given a context-free grammar G, is $L(G) = \emptyset$?
III. Given a context-free grammar G, is $L(G) = \Sigma^*$ for some alphabet Σ ?
IV. Given a Turing machine M and a string w, is $w \in L(M)$?

(A) I and IV only (B) II and III only
(C) II, III and IV only (D) III and IV only

17. The set of all recursively enumerable languages is: **[2018]**

(A) Closed under complementation.
(B) Closed under intersection.
(C) A subset of the set of all recursive languages.
(D) An uncountable set.

18. Consider the following problems. $L(G)$ denotes the language generated by a grammar G. $L(M)$ denotes the language accepted by a machine M.

(I) For an unrestricted grammar G and a string w, whether $w \in L(G)$

(II) Given a Turing machine M, whether $L(M)$ is regular

(III) Given two grammars G_1 and G_2, whether $L(G_1) = L(G_2)$

(IV) Given and NFA N, whether there is a deterministic PDA P such that N and P accept the same language.

Which one of the following statements is correct? **[2018]**

(A) Only I and II are undecidable
(B) Only III is undecidable
(C) Only II and IV are undecidable
(D) Only I, II and III are undecidable

Answer Keys

Exercises

Practice Problems 1

1. D	**2.** C	**3.** B	**4.** B	**5.** D	**6.** D	**7.** A	**8.** D	**9.** D	**10.** D
11. C	**12.** A	**13.** D	**14.** A	**15.** C					

Practice Problems 2

1. B	**2.** D	**3.** B	**4.** B	**5.** A	**6.** B	**7.** C	**8.** B	**9.** D	**10.** B
11. C	**12.** B	**13.** B	**14.** B	**15.** A					

Previous Years' Questions

1. D	**2.** D	**3.** D	**4.** C	**5.** B	**6.** C	**7.** C	**8.** D	**9.** B	**10.** D
11. D	**12.** C	**13.** D	**14.** C	**15.** A	**16.** D	**17.** B	**18.** D		