TOC Important Questions PDF (Unit-wise)

The most important Theory of Computation questions are designing DFAs for given languages, NFA to DFA conversion, DFA minimisation, Mealy and Moore machines, regular expression to finite automaton (and back with Arden's theorem), the pumping lemma for regular and context-free languages, CFG simplification, CNF and GNF, PDA design, CFG to PDA conversion, Turing machine design, and decidability with the halting problem.

Key takeaways

- Automaton design and conversion questions carry the most marks.
- Draw every automaton with a clear start state, final states and a transition table.
- Pumping lemma proofs follow a fixed template: learn it once and reuse it.
- These are practice questions, not past papers; check your own syllabus and official previous papers.

About these important questions

This page lists the most important Theory of Computation questions for B.Tech CSE and IT exams, arranged unit by unit. The units follow the standard syllabus that most Indian technical universities share, so the order may differ slightly from yours.

Please note: these are practice questions based on common syllabus topics. They are not verified past papers, and we do not claim that any question will appear in your exam. Check your own university syllabus first, then solve the official previous year papers. AKTU students can download them from the AKTU old question paper archive.

Unit 1: Finite Automata

Covers alphabets, languages, DFA, NFA and minimisation.

Short questions (2 marks)

- Define a DFA as a 5-tuple.
- Differentiate between a DFA and an NFA.
- What is an epsilon-closure?

Long questions

1. Design a DFA that accepts strings over {0, 1} ending in 01, or with an even number of 0s and 1s, or whose binary value is divisible by 3.

2. Convert a given NFA (with or without epsilon moves) to a DFA using subset construction.
3. Minimise a DFA using the partition (equivalence) method or the table-filling method.
4. Design a Mealy machine and a Moore machine for a given problem, and convert one into the other.
5. Explain the Myhill-Nerode theorem in brief.

Unit 2: Regular Expressions and Regular Languages

Covers regular expressions, Arden's theorem, the pumping lemma and closure properties.

Short questions (2 marks)

- Write a regular expression for strings over {a, b} containing "aa".
- State Arden's theorem.
- List three closure properties of regular languages.

Long questions

1. Convert a regular expression such as (a + b)*abb into an epsilon-NFA and then a DFA.
2. Find the regular expression of a given DFA using Arden's theorem or state elimination.
3. Prove that L = {a^n b^n | n ≥ 0} is not regular using the pumping lemma.
4. Prove the equivalence of two regular expressions.
5. Explain decision properties of regular languages (emptiness, finiteness, membership).

Unit 3: Context-Free Grammars

Covers CFGs, derivations, ambiguity, simplification and normal forms.

Short questions (2 marks)

- Define a context-free grammar.
- When is a grammar ambiguous?
- What is a unit production?

Long questions

1. Write a CFG for a language such as {a^n b^n}, palindromes, or balanced brackets, and draw the leftmost derivation and parse tree for a string.
2. Show that a grammar such as E → E + E | E * E | id is ambiguous.
3. Simplify a CFG by removing null, unit and useless productions.
4. Convert a CFG into Chomsky normal form (CNF) and Greibach normal form (GNF).
5. Prove that {a^n b^n c^n} is not context-free using the pumping lemma for CFLs.
6. Explain the closure properties of context-free languages.

Unit 4: Pushdown Automata

Covers PDA design, acceptance and equivalence with CFG.

Short questions (2 marks)

- Define a PDA as a 7-tuple.
- Differentiate between acceptance by final state and by empty stack.
- Is a deterministic PDA as powerful as a non-deterministic PDA?

Long questions

1. Design a PDA for {a^n b^n | n ≥ 1} and show its instantaneous descriptions for aabb.
2. Design a PDA for even-length palindromes {w w^R}.
3. Convert a given CFG into an equivalent PDA.
4. Convert a given PDA into a CFG.
5. Explain two-stack PDA and why it is as powerful as a Turing machine.

Unit 5: Turing Machines and Decidability

Covers Turing machine design, variants, the Chomsky hierarchy and undecidability.

Short questions (2 marks)

- Define a Turing machine.
- State the Church-Turing thesis.
- What is a recursively enumerable language?

Long questions

1. Design a Turing machine for {a^n b^n c^n}, or to add two unary numbers, or to find the 2's complement.
2. Explain variants of Turing machines: multi-tape, non-deterministic and universal.
3. Explain the Chomsky hierarchy with the machine for each type.
4. Prove that the halting problem is undecidable.
5. Explain Post's correspondence problem and check an instance for a solution.
6. Differentiate between recursive and recursively enumerable languages.

Sample answer outline: Prove a language is not regular with the pumping lemma

For L = {a^n b^n | n ≥ 0}, use this template:

1. **Assume L is regular.** Then a pumping length p exists.
2. **Choose a string.** Take w = a^p b^p, which is in L and has length 2p ≥ p.
3. **Split it.** Any split w = xyz with |xy| ≤ p and |y| ≥ 1 means y has only a's, say y = a^k with k ≥ 1.
4. **Pump.** Take i = 2: xy^2z = a^(p+k) b^p. It has more a's than b's, so it is not in L.
5. **Conclude.** This contradicts the pumping lemma, so L is not regular.

The pumping lemma for CFLs works the same way, except the split is w = uvxyz with |vxy| ≤ p and |vy| ≥ 1, and you must check every case of where vxy can fall. For {a^n b^n c^n}, vxy cannot touch all three letters, so pumping breaks the equal counts.

Marks weightage and how to write answers

A common pattern, used by AKTU and many other universities, has a compulsory short-answer section that covers every unit, followed by long-answer questions with a choice inside each unit. Marks per question change between universities and years, so confirm the scheme on a recent official paper.

- **Short answers (about 2 marks):** a crisp definition, one example or diagram, and nothing more. Three to five lines is enough.
- **Long answers (7 to 10 marks):** start with a definition, add a labelled diagram or table, explain the steps, and finish with an example, advantages or a comparison.
- For DFA design, write the meaning of every state (for example, "last symbol was 0") beside the diagram. It shows logic, not guesswork.
- For conversions, give the transition table as well as the diagram, so the examiner can check each step.

A 7-day revision plan for Theory of Computation

1. **Days 1 to 5:** one unit a day. Read your notes, then write the long answers for that unit from memory and check them against your notes.
2. **Day 6:** redo NFA to DFA, DFA minimisation, Arden's theorem, CNF conversion, one PDA and one Turing machine design.
3. **Day 7:** solve one full official previous year paper in three hours, then check which units cost you time.

Keep a single sheet of formulas, definitions and diagrams for the last hour before the exam.

Frequently asked questions

Which Theory of Computation topics are most important?

DFA design and minimisation, NFA to DFA conversion, regular expressions, the pumping lemma, CFG normal forms, PDA design and Turing machine design are asked most often, along with the halting problem.

How can I get better at designing DFAs?

Decide what each state must remember, such as the remainder so far or the last two symbols. Write that meaning first, then add transitions. Test your DFA on three accepted and three rejected strings.

Is Theory of Computation useful for programming?

Yes. Regular expressions, parsers, compilers and input validation all use ideas from automata theory.

Are these Theory of Computation questions from previous year papers?

No. They are practice questions built from topics that appear in most B.Tech Theory of Computation syllabi. Use them with your own university's official previous papers, for example the AKTU archive at https://aktu.ac.in/old-question-paper.html.

Is this PDF free?

Yes. Fill in the short form on this page and the PDF downloads right away.

Sources and further reading

- AKTU: old question papers (official archive)
- MIT OpenCourseWare: Theory of Computation (18.404J)
- NPTEL: online courses from IITs (search for Theory of Computation)

About this PDF

These are practice questions for revision, prepared by the Uncodemy trainer team from common B.Tech syllabus topics. They are not past papers, a certificate or a promise of marks. Spot an error? Write to info@uncodemy.com and the team will review it.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550