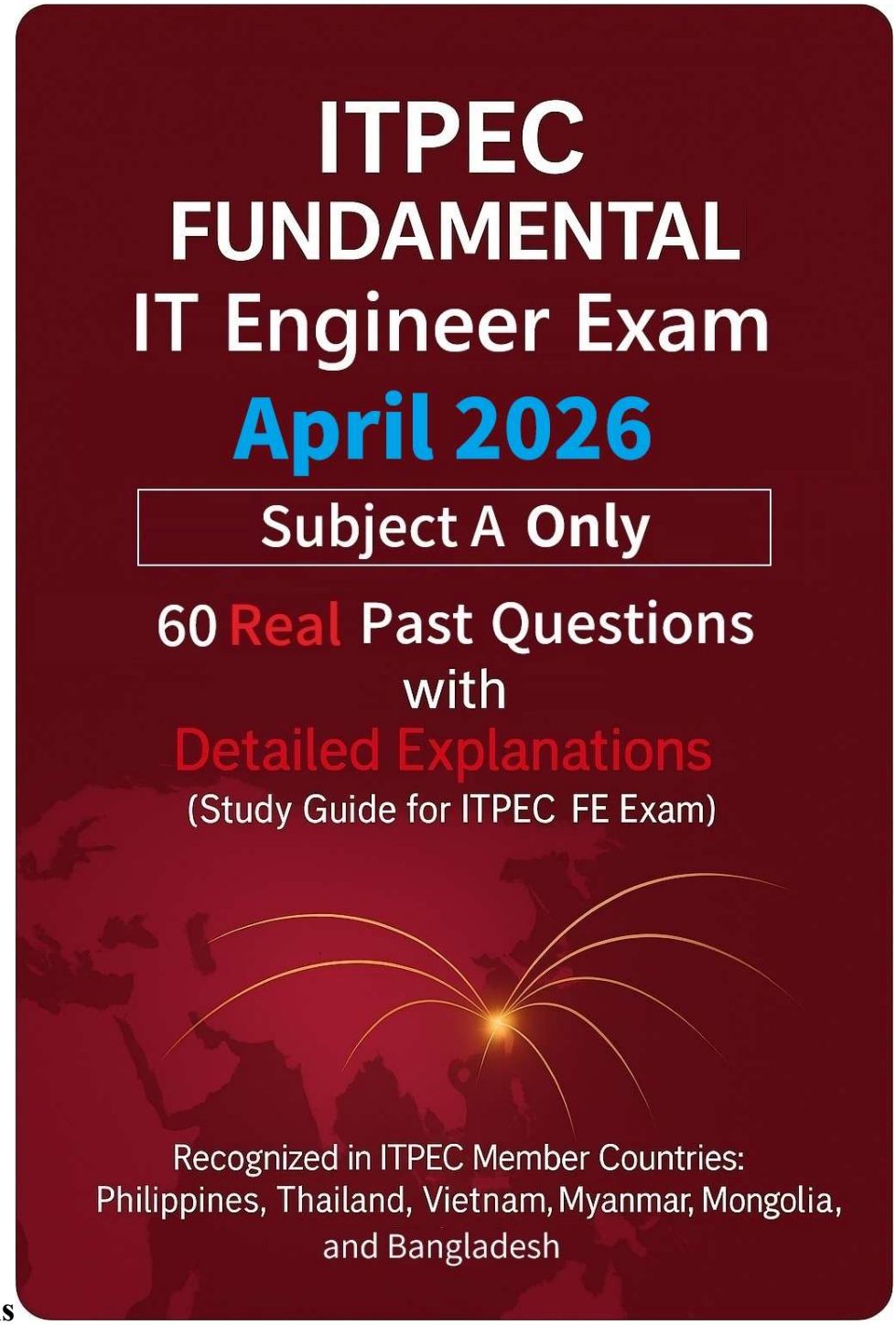
This Book is Sample Edition



his

ITPEC Fundamentals of Information Technology Engineer Examination (FE)

– English Edition April 2026 Subject A Version

Author: Takashi Narita

© 2026 Takashi Narita. All rights reserved.

Based on past **FE Examination** questions published by ITPEC.

Disclaimer

This book contains past exam questions from the Fundamental Information Technology Engineer Examination (FE), administered by the Information Technology Professionals Examination Council (ITPEC), specifically from the April 2026 Examination. The questions are reproduced with permission for educational purposes, and the copyright of the original questions remains with ITPEC.

All explanations, translations, and analyses are original work by the author, Takashi Narita.

Preface

The **Fundamental Information Technology Engineer Examination (FE)** is a core examination that demonstrates your ability to apply fundamental knowledge of information technology, programming, algorithms, and system design. Originally developed in Japan, it is now recognized internationally through the ITPEC (Information Technology Professionals Examination Council) mutual recognition framework. Member countries include the Philippines, Thailand, Vietnam, Myanmar, Mongolia, and Bangladesh.

In many of these countries, the FE serves as a gateway for those aiming to work in Japan or for Japanese companies abroad. Where Japanese-owned IT firms operate, holding this certification can also provide a strong advantage in hiring and career advancement, as it reflects both technical competence and familiarity with Japanese corporate culture and IT standards.

This book is designed for learners who wish to prepare for the **FE Examination in English**. It offers clear explanations, answer analyses, and study tips to help you build genuine understanding rather than relying on rote memorization. Whether you are a student, a working professional, or someone seeking to advance your IT career, this book will support efficient, practical, and motivating study.

Important Notice

The ITPEC Fundamentals of Information Technology Engineer Examination (FE) consists of two parts: Subject A and Subject B.

Each subject has different characteristics and types of questions.

This book covers only Subject A.

➞ For Subject B, please purchase the separate volume.

➞ A combined edition (Subjects A & B) is also available.

About the Author

Takashi Narita is an IT instructor with more than 20 years of industry experience in system design, software development, programming, and database management. Now based in the Philippines, he teaches programming, system development, and system design remotely to students in Japan.

He began creating this English edition of **Fundamental Information Technology Engineer Examination (FE)** practice questions to support IT human resource development in the Philippines. Over time, he recognized that it could also serve as a valuable resource for learners across Asia seeking to succeed in Japanese-owned companies or in Japan itself.

Through this book, he aims to help readers build the skills and confidence needed to contribute meaningfully in Japanese and Japan-affiliated workplaces. He is passionate about making complex IT concepts accessible to learners of all backgrounds and believes that studying in English opens doors to international opportunities.

How to Use This Book

The fastest way to pass the **Fundamental Information Technology Engineer Examination (FE)** is to practice past questions repeatedly and understand the logic behind each answer.

This book provides past FE questions in English, along with clear explanations and answer analyses. Always review the explanations for any questions you miss, make sure you understand the reasoning, and then try again.

Recommended study flow:

1. Solve one set of questions under timed conditions, just like the actual exam.
2. Check which answers were incorrect and study the explanations carefully.
3. Reattempt the same set a few days later until you can achieve a high score with confidence.
4. Move on to the next set and repeat the process.

Study Tips

• **Do more than one set** – One set of past questions is not enough for thorough preparation. Aim for at least three sets to build both knowledge and confidence.

• **Consistency is key** – Even 10 minutes a day makes a difference. Use your phone, PC, or tablet to open this book and solve just one question. From my own experience earning multiple IT certifications, **daily habit** is the single most important factor in success. For me, solving one question before bed soon grew naturally to two or three without feeling forced. Short, consistent study sessions lead to long-term retention.

• **Set a study schedule** – Work backward from your exam date, create a plan, and track your progress.

• **Get used to English IT terms** — many are 3–4-letter abbreviations (e.g., CPU, LAN, DNS) and can feel intimidating at first, but with repeated exposure they'll become second nature.

Exam Overview

• **Format: Two parts** — **Subject A** (formerly “Morning”) and **Subject B** (formerly “Afternoon”).

• **Exam Areas (Subject A):** 60 multiple-choice questions in 90 minutes.

Subject A consists of **60 questions** drawn from the three domains below.

1) Technology Domain

IT fundamentals, hardware, software, databases, networks, security, etc.

2) Management Domain

project management, service management, system audit, etc.

3) Strategy Domain

management strategy, business law, system strategy, corporate activities, etc.

You should aim to spend about 1.5 minutes (90 seconds) per question.

• **Exam Areas (Subject B):** 20 multiple-choice questions in 100 minutes.

Algorithms & Programming (pseudo-code); Information Security.

(**20 questions** total: 16 Algorithms & Programming, 4 Information Security)

You should aim to spend about 5 minutes per question.

• **Passing Criteria (FE):**

You must pass both Subject A and Subject B. The passing mark for each subject is 60 out of 100.

Links

The test mode (CBT or paper), schedule, application method, and venue vary by country. Details are subject to change; always check your country's official website for the latest information.
Official Information Links:

- ITPEC Official Site: https://itpec.org/
- Philippines (DICT): https://dict.gov.ph/itpec/
- Thailand (NECTEC): https://www.nectec.or.th/itpec/
- Vietnam (VITC): https://www.vitc.vn/
- Myanmar (ITPEC Myanmar): https://www.myanmaritpec.org/
- Bangladesh (BCC): https://www.bcc.gov.bd/
- Mongolia (MITC): https://www.mitc.gov.mn/

Author's Note: A quick word on Subject A

Subject A assesses **fundamental IT knowledge**. The most reliable way to reach the passing mark is to **practice many questions**. In recent years, Subject A has often included **reused or closely related items** from past exams, so broad exposure to previous questions is especially effective.
If you are new to the FE, consider taking the **IT Passport** exam first. Building that foundation makes Subject A **feel much easier** for many learners, allowing you to focus your energy on **Subject B** afterward.

*ITPEC provides official Subject A materials in the downloadable past exams. For authoritative conventions—especially **logic-circuit notation and rules**—please refer to those documents.*

Exams are always a race against time. If you glance at a question and feel a strong sense of dread, move on to another one. The best strategy is to start with the questions that make you think, "I can do this!"

A-Q1

Which of the following is the prefix notation for the infix expression ((*a* − *b*) / (*c*+*d*)) ^*e* .
Here, the symbols “+,” “−,” “/,” and “^” represent addition, subtraction, division, and exponentiation operators, respectively.

a) ^ *e*/+*dc* – *ba*
b) ^ − *ab*/+*cde*
c) ^ / − *ab*+*cde*
d) ^ *e* / − *ab*+*cd*

(Source:2026S,FE,Subject-A,Q1)

Answer: c

Explanation

To convert an infix expression into prefix notation, always identify the **main operator** first and then process each subexpression recursively.

The given expression is:

((a − b) / (c + d)) ^ e

The main operator is ^ **(exponentiation)**, so the prefix expression must begin with ^.

Next, convert the left operand:

(a − b) / (c + d)

The main operator here is /, so it becomes:

/ − ab + cd

Finally, attach the right operand **e**.

The complete prefix expression is:

^ / − ab + cd e

Therefore, the correct answer is:

c) ^ / − ab+cde

Author's Comments

Many students try to convert the expression all at once and become confused.

Instead, always find the **outermost operator** first. Then divide the expression into smaller parts and convert each part separately.

This step-by-step approach works not only for prefix notation but also for postfix notation and expression trees.

Column

Three Ways to Write an Expression

There are three common ways to write a mathematical expression.

The first is **infix notation**, which we use every day. For example:

a + b

The operator is written between the two values.

The second is **prefix notation**, where the operator comes first:

+ a b

The third is **postfix notation**, where the operator comes last:

a b +

Although prefix and postfix expressions may look strange at first, they have one important advantage: **they do not need parentheses**. The order of operations is clear from the position of the operators.

These expression formats are often used inside compilers and calculators. They are also important topics in computer science and programming.

A-Q2

What is the decimal equivalent value of the octal number 1234.2?

a) 27.0
b) 668.25
c) 1542.5
d) 5346.0

(Source:2026S,FE,Subject-A,Q2)

Answer: b

Explanation

To convert an octal number to its decimal equivalent, multiply each digit by the corresponding power of **8**.

The given number is:

1234.2_8

Expand it as follows:

- **$1 \times 8^3 = 512$**
- **$2 \times 8^2 = 128$**
- **$3 \times 8^1 = 24$**
- **$4 \times 8^0 = 4$**
- **$2 \times 8^{-1} = 0.25$**

Now add all the values:

512 + 128 + 24 + 4 + 0.25 = **668.25**

Therefore, the correct answer is:

b) 668.25

Author's Comments

When converting numbers between different bases, don't try to memorize the answers. Instead, remember the basic rule:

Multiply each digit by the corresponding power of the base.

This method works not only for octal numbers but also for binary and hexadecimal numbers. Once you understand the rule, you can convert numbers in any base with confidence.

Column

Why Do Computers Use Different Number Systems?

We use the decimal system because we have ten fingers. However, computers work with only two states: **ON** and **OFF**. That is why computers use the **binary** number system.

Other number systems are also useful. **Octal (base 8)** and **hexadecimal (base 16)** are shorter ways to write long binary numbers. For example, one hexadecimal digit represents four binary digits, making large values much easier to read.

Although programmers today rarely write numbers in octal, hexadecimal numbers are still widely used in programming, computer architecture, and debugging. Understanding different number systems will help you understand how computers store and process data.

A-Q3

Which of the following is the ratio of the mean arrival rate to the mean service rate in queuing theory?

a) Utilization
b) Average inter-arrival time
c) Average service time
d) Average retention number

(Source:2026S,FE,Subject-A,Q3)

Answer: a

Explanation

In queuing theory, **utilization** is the ratio of the **mean arrival rate** to the **mean service rate**.

It is calculated as:

Utilization = Mean Arrival Rate / Mean Service Rate

Utilization indicates how busy a service system is. For example, a utilization of **0.8** means the server is busy 80% of the time.

Therefore, the correct answer is:

a) Utilization

Author's Comments

Don't confuse **arrival rate** with **service time**.

The arrival rate tells us how frequently customers or jobs arrive, while the service rate tells us how many customers or jobs can be processed in a given time.

Remember that **utilization compares these two rates**. This is one of the most important concepts in queuing theory.

Column

Why Is 100% Utilization a Problem?

It may seem that a server working at **100% utilization** is operating perfectly. After all, no time is being wasted.

In reality, however, a system with very high utilization often performs poorly. When a new customer or job arrives while the server is already busy, it must wait. As utilization gets closer to 100%, the waiting line can grow very quickly.

Think about a supermarket with only one cashier. If the cashier is busy every minute of the day, even a few extra customers can create a long line. Adding a second cashier or leaving some idle time can greatly reduce waiting times.

For this reason, many real-world computer systems are designed to operate below full utilization. A small amount of spare capacity helps the entire system respond more smoothly when demand suddenly increases.

A-Q4

In regular expressions (regex), which of the following is the string that correctly matches the pattern ^¥d{2,4}-¥d+-¥d{3,4}$? Here, “¥d” represents a decimal digit 0-9, the “+” symbol indicates one or more repetitions, and the “$” symbol denotes the line's end.

a) 123-456-12
b) 1-234-5678
c) 123-456-7890
d) 12345-678-910

(Source:2026S,FE,Subject-A,Q4)

Answer: c

Explanation

Let's examine the regular expression step by step.

^¥d{2,4}-¥d+-¥d{3,4}$

- ^ means the beginning of the line.
- **¥d{2,4}** means **2 to 4 digits**.
- **-** is a hyphen.
- **¥d+** means **one or more digits**.
- Another **-** follows.
- **¥d{3,4}** means **3 or 4 digits**.
- **$** means the end of the line.

Now check each choice:

- **a)** The last part has only **2 digits**, so it does not match.
- **b)** The first part has only **1 digit**, so it does not match.
- **c)** The first part has **3 digits**, the middle part has **3 digits**, and the last part has **4 digits**. This matches the pattern.
- **d)** The first part has **5 digits**, which exceeds the maximum of 4 digits.

Therefore, the correct answer is:

c) 123-456-7890

Author's Comments

When solving regex questions, read the pattern from **left to right** instead of trying to understand it all at once.

Pay special attention to symbols such as **{ }**, **+**, *****, **^**, and **$**. Each symbol has a specific meaning, and understanding these meanings makes regex much easier to read.

In many exam questions, checking each part of the pattern one by one is the fastest way to find the correct answer.

Column

Where Are Regular Expressions Used?

Regular expressions, or **regex**, are used in many computer applications. One common example is **form validation** on websites. When you enter an email address, phone number, or postal code, regex can check whether the input has the correct format.

Regex is also useful for searching and replacing text. Many text editors, programming languages, and command-line tools support regex, allowing users to find complex text patterns quickly.

Although regular expressions may look difficult at first, they are built from a small set of symbols and rules. Once you learn the basic patterns, you can solve many text-processing tasks with just a single expression.

A-Q5

Which of the following is the postfix notation for the expression $a+(b\times c)+(d\times e)$?

a) $a+bc\times de+\times$

b) $abc\times+de+\times$

c) $abc\times+de\times+$

d) $abc+\times de\times+$

(Source:2026S,FE,Subject-A,Q5)

Answer: c

To convert an infix expression to postfix notation, write the **operands first** and place each **operator after its operands**.

The expression is:

a + (b × c) + (d × e)

First, convert each multiplication:

- **b × c → bc×**
- **d × e → de×**

Next, add **a** and **bc×**:

- **a + (b × c) → abc×+**

Finally, add **de×**:

- **(a + (b × c)) + (d × e) → abc×+de×+**

Therefore, the correct answer is:

c) abc×+de×+

Author's Comments

A useful strategy is to convert the expression **one operation at a time**.

Instead of converting the whole expression at once, first convert the expressions inside the parentheses. Then combine the results step by step.

This approach reduces mistakes and is especially helpful for longer expressions.

Column

Why Do Compilers Like Postfix Notation?

Computers do not read mathematical expressions the same way people do. Humans can easily understand parentheses and operator precedence, but computers need clear rules to evaluate expressions.

Postfix notation is useful because it removes the need for parentheses. A computer can evaluate a postfix expression from left to right using a **stack**, a simple data structure that stores values in order.

For example, when the computer reads **abc**×+, it pushes **a**, **b**, and **c** onto the stack. When it reaches ×, it multiplies **b** and **c**. When it reaches +, it adds the result to **a**.

This simple process is one reason why postfix notation has been widely used in compilers, calculators, and expression evaluators.

Mastering Tree Traversals (Post-order)

Core idea

Post-order means **Left → Right → Root**. Think "**children before parent**." The root is **always last**—a quick way to eliminate wrong options.

Why it matters

- Used to **delete/free** an entire tree safely (process children, then parent).
- Evaluating **expression trees** follows post-order (operands first, operator last).
- Many divide-and-conquer algorithms naturally produce post-order results.

How to trace quickly (exam technique)

1. Put your pencil on the **root**.
2. Go as far **left** as possible. When a node has no left child, try **right**; when both are done, **record the node** and climb up.
3. Repeat until the root is recorded last.

   Tip: Lightly mark a node with dots: first time seen "•", after left "••", after right "•••" → write it down and return.

Mini-pseudocode (to memorize)

post(v): if v==null return; post(v.left); post(v.right); visit(v)

Common pitfalls

- Mixing orders (e.g., writing the root too early).
- Forgetting to finish the right subtree before listing the parent.
- Confusing **in-order** (Left–Root–Right) and **pre-order** (Root–Left–Right).

This Book is Sample Edition

Notes and Disclaimers

Source Note:

The questions in this book are based on the past questions from the April 2026 FE (Fundamentals of Engineering) Examination published on the official ITPEC website. The author translated them into English and added supplementary explanations.

Source: ITPEC – FE Examination Past Questions (https://itpec.org/)

Relationship with ITPEC:

This book is an independent publication. It is created in accordance with the usage policy for past exam questions published by ITPEC, but it is not officially affiliated with, authorized, or endorsed by ITPEC.

General Disclaimer:

The content of this book is based on information available at the time of writing. The examination format, scope, or content may change. Please refer to the official ITPEC website for the most up-to-date **information.**

License and Usage:

This book is intended for personal study purposes. Redistribution or commercial use of the explanations and translations by the author without permission is prohibited.

Feedback Welcome:

If you find any errors or have suggestions for improvement, please feel free to contact the author. Your input will help improve future editions of this book.

Copyright & Source Information

Copyright © 2026 Takashi Narita

All rights reserved.

This book contains past exam questions from the **FE (Fundamentals of Information Technology Engineer) Examination** published by ITPEC, reproduced in accordance with ITPEC's public usage policy for educational purposes. The copyright of the original questions remains with ITPEC.

Source: https://itpec.org/