



April 2026

Fundamental IT Engineer Examination (Subject B)

Questions must be answered in accordance with the following:

Question Nos.	Q1 – Q20
Question Selection	All questions are compulsory.
Examination Time	12:30 – 14:10 (100 minutes)

Instructions:

1. Use a pencil. If you need to change an answer, erase your previous answer completely and neatly. Wipe away any eraser debris.
2. Mark your examinee information and test answers in accordance with the instructions below. Your answer will not be graded if you do not mark properly. Do not mark or write on the answer sheet outside of the prescribed places.

(1) **Examinee Number**

Write your examinee number in the space provided, and mark the appropriate space below each digit.

(2) **Date of Birth**

Write your date of birth (in numbers) exactly as it is printed on your examination admission card, and mark the appropriate space below each digit.

(3) **Answers**

Mark your answers as shown in the sample question below.

[Sample Question]

Which of the following should be used for marking your answer on the answer sheet?

Answer group

- a) Ballpoint pen b) Crayon c) Fountain pen d) Pencil

Since the correct answer is “d) Pencil”, mark the answer as below:

[Sample Answer]

Sample	☐ a	☐ b	☐ c	☒	☐ e	☐ f	☐ g	☐ h	☐ i	☐ j
--------	-------------------------	-------------------------	-------------------------	----------------------------------	-------------------------	-------------------------	-------------------------	-------------------------	-------------------------	-------------------------

**Do not open the exam booklet until instructed to do so.
Inquiries about the exam questions will not be answered.**

Pseudo programming language notations

In algorithm and programming questions that use pseudo programming language, the following notations are used unless otherwise stated:

[Pseudo programming language notations]

Notation	Description
$\bigcirc$ <i>procedure</i> (<i>type</i> : <i>arg1</i> , ...)	Declares a <i>procedure</i> and its argument(s) <i>arg1</i> ,
$\bigcirc$ <i>ret-type</i> : <i>function</i> (<i>type</i> : <i>arg1</i> , ...)	Declares a <i>function</i> , its argument(s) <i>arg1</i> , ... , and type of return value <i>ret-type</i> .
<i>type</i> : <i>var1</i> , ... <i>type</i> []: <i>array1</i> , ...	Declares variables <i>var1</i> , ... and arrays <i>array1</i> , ... by data <i>type</i> such as integer, real, and string.
<i>/* comment */</i>	Describes a comment between <i>/*</i> and <i>*/</i> .
<i>// comment</i>	Describes a comment after <i>//</i> till end of line.
<i>variable</i> $\leftarrow$ <i>expression</i>	Assigns the value of the <i>expression</i> to the <i>variable</i> .
<i>procedure</i> (<i>arg1</i> , ...)	Calls a <i>procedure</i> by passing arguments <i>arg1</i> ,
<i>function</i> (<i>arg1</i> , ...)	Calls a <i>function</i> by passing arguments <i>arg1</i> , ... , and receiving the return value.
output <i>arg1</i> , ...	Outputs values of <i>arg1</i> , ... to a printing device.
return <i>ret-val</i>	Finishes a function by passing back a return value <i>ret-val</i> .
<pre> if (<i>condition-i</i>) } *1 <i>process-i</i> elseif (<i>condition-ei</i>) } *2 <i>process-ei</i> else } *3 <i>process-e</i> endif </pre>	Indicates the selection process. *1 If <i>condition-i</i> is true, then execute <i>process-i</i>. Otherwise, proceed to the next elseif or else. *2 If <i>condition-ei</i> is true, then execute <i>process-ei</i>. Otherwise, proceed to the next elseif or else. *3 If all conditions are false, execute <i>process-e</i>. Note: *2 and *3 can be omitted. *2 may exist twice or more.
<pre> for (<i>sequence</i>) <i>process</i> endfor </pre>	Indicates the “for” iteration process. In the order specified in the <i>sequence</i>, execute the <i>process</i> repeatedly.
<pre> while (<i>condition</i>) <i>process</i> endwhile </pre>	Indicates the “while” iteration process. While the <i>condition</i> is true, execute the <i>process</i> repeatedly.
<pre> Do <i>process</i> while (<i>condition</i>) </pre>	Indicates the “do - while” iteration process. Execute the <i>process</i> once, and then while the <i>condition</i> is true, execute the <i>process</i> repeatedly.

Pseudo programming language notations (continued)

[Operators and their precedence]

Type of operator	Operators	Precedence	Note
Expression	(), . ⁽¹⁾	High ↑ ↓ Low	⁽¹⁾ accessing member or method
Unary operator	+, -, not ⁽²⁾		⁽²⁾ logical negation
Binary operator	×, ÷, mod ⁽³⁾		⁽³⁾ remainder
	+, -		
	>, <, ≥, ≤, =, ≠		
	and ⁽⁴⁾		⁽⁴⁾ logical product
	or ⁽⁵⁾		⁽⁵⁾ logical sum

[Boolean-type constants]

true, false

[Array reference]

	1-dimensional array	2-dimensional array	Array of arrays
Array declaration	<i>type</i> []: <i>name</i> ...	<i>type</i> [,]: <i>name</i> ...	<i>type</i> [] []: <i>name</i> ...
Example	integer []: a1 1 2 3 4 5 1 3 5 7 9	integer [,]: a2 1 2 3 1 11 12 13 2 14 15 16 3 17 18 19	integer [] []: aa 1 2 3 1 21 22 2 23 24 25 3 26
Data reference	Data 7 is referred to by a1[4]	Data 16 is referred to by a2[2, 3]	Data 25 is referred to by aa[2][3]
Notation of array contents	{1, 3, 5, 7, 9}	{11, 12, 13}, {14, 15, 16}, {17, 18, 19}}	{21, 22}, {23, 24, 25}, {26}}

Note: The indexes of example arrays start at 1.

[undefined state]

undefined is a state in which no value is set to a variable (or an element of an array).

By setting undefined to a variable, the variable is transformed into undefined state.

Q1. From the answer group below, select the correct answer to be inserted into in the description. Here, the array index starts at 1.

When the program is executed, the output is “”.

[Program]

```
integer: x ← 1
integer: y ← 2
integer: z ← 3
integer []: ar ← {0, 0}
y ← x
ar[x] ← y
z ← y
x ← z
ar[2] ← z + ar[1]
output ar[1], ar[2]    // the values are separated by ", "
```

Answer group

- | | | |
|---------|---------|---------|
| a) 1, 0 | b) 1, 1 | c) 1, 2 |
| d) 2, 1 | e) 2, 2 | f) 2, 3 |

Q2. From the answer group below, select the correct combination of answers to be inserted into through in the description.

Function `f` receives an integer value as argument `argval` and returns the grade character. The return value of `f(92)`, `f(72)`, and `f(-10)` are “,” “,” and “,” respectively. Note that the function has a specification bug.

[Program]

```

O character: f(integer: argval)
  character: retvar ← "w"
  if (argval > 90)
    retvar ← "a"
  endif
  if (argval > 70)
    retvar ← "b"
  elseif (argval > 60)
    retvar ← "c"
  elseif (argval ≤ 60)
    retvar ← "d"
  endif
  return retvar

```

Answer group

	A	B	C
a)	a	b	d
b)	a	c	d
c)	a	c	w
d)	b	b	d
e)	b	c	d
f)	b	c	w

Q3. From the answer group below, select the correct answer to be inserted into in the program. Here, the array indexes start at 1.

The function `distance` receives integer arrays `S` and `T` of size `n` as arguments and returns the Manhattan distance of the two vectors that can be computed as $\sum_{i=1}^n |S_i - T_i|$. Function `abs`, which is used by the function `distance`, returns the absolute value of an integer. Assume that `n` is 1 or greater.

[Program]

```
O integer: distance(integer []: S, integer []: T, integer: n)
  integer: p ← 0
  integer: i
  for (increase i from 1 to n by 1)
    
  endfor
  return p
```

Answer group

- a) `p ← p + abs(S[i] + T[i])`
- b) `p ← p + abs(S[i] - T[i])`
- c) `p ← p - abs(S[i] + T[i])`
- d) `p ← p - abs(S[i] - T[i])`

Q4. From the answer group below, select the correct combination of answers to be inserted into through in the program.

The function `isHarshad` checks whether the given integer `n` is a Harshad number. The function returns `true` if the given number is a Harshad number, or `false` otherwise. A Harshad number, also known as a Niven number, is a positive integer that is divisible by the sum of its digits. For instance, 156 is a Harshad number because the sum of its digits ($1 + 5 + 6 = 12$) divides it evenly without a remainder ($156 \div 12 = 13$).

[Program]

```
O boolean: isHarshad(integer: n)
    integer: num ← n
    integer: sum ← 0
    if (num < 1)
        return false
    endif
    while (num > 0)
        sum ← sum + ()
        num ← ()
    endwhile
    return () // Return true if n is a Harshad number,
                                // or false otherwise.
```

Answer group

	A	B	C
a)	integer part of ($\text{num} \div 10$)	$\text{num} \bmod 10$	$\text{sum} = \text{n}$
b)	integer part of ($\text{num} \div 10$)	$\text{num} \bmod 10$	$\text{num} \bmod \text{sum} = 0$
c)	integer part of ($\text{num} \div 10$)	$\text{num} \bmod 10$	$\text{n} \bmod \text{sum} = 0$
d)	$\text{num} \bmod 10$	integer part of ($\text{num} \div 10$)	$\text{sum} = \text{n}$
e)	$\text{num} \bmod 10$	integer part of ($\text{num} \div 10$)	$\text{num} \bmod \text{sum} = 0$
f)	$\text{num} \bmod 10$	integer part of ($\text{num} \div 10$)	$\text{n} \bmod \text{sum} = 0$

Q5. From the answer group below, select the correct answer to be inserted into in the description.

When the procedure proc1 is called, the output is “” in turn.

[Program]

☐ proc1()
 proc2()
 proc3()
 output "A, "

☐ proc2()
 output "B, "
 proc3()

☐ proc3()
 output "C, "

Answer group

- a) A, B, B, C,
- b) A, B, C, C,
- c) B, A, C, C,
- d) B, C, B, A,
- e) B, C, C, A,
- f) C, B, A, C,
- g) C, B, B, A,
- h) C, B, C, A,

Q6. From the answer group below, select the correct combination of answers to be inserted into and in the program.

The quadratic equation $a_1x^2 + b_1x + c_1 = 0$ can be formulated as $x^2 - 2(-b_1/2a_1) + (c_1/a_1) = 0$, where $a_1 \neq 0$. If we replace $(-b_1/2a_1)$ and (c_1/a_1) by b and c respectively, it can be further formulated as $x^2 - 2bx + c = 0$, and the roots are as follows:

Case 1: if $b^2 - c = 0$, double root b ,

Case 2: if $b^2 - c > 0$, two real roots $b \pm \sqrt{b^2 - c}$,

Case 3: if $b^2 - c < 0$, two imaginary roots $b \pm i\sqrt{c - b^2}$, where $i = \sqrt{-1}$.

The procedure `quadEquation` takes the coefficients of a quadratic equation in arguments `a1`, `b1`, and `c1` and uses the above formula to determine and output the roots. Here, `a1` is not 0 and function `sqrt` returns the principal square root of its non-negative argument.

[Program]

```

O quadEquation(real: a1, real: b1, real: c1)
  real: b ← -b1 ÷ (2 × a1)
  real: c ← c1 ÷ a1
  real: val ← b × b - c
  real: tmp
  if (val = 0)
    output "root1=root2=", 
  elseif (val > 0)
    tmp ← sqrt(val)
    output "root1=", b + tmp
    output "root2=", b - tmp
  else
    tmp ← 
    output "root1=", b, "+", tmp, "i"
    output "root2=", b, "-", tmp, "i"
  endif

```

Answer group

	A	B
a)	-b	-sqrt(val)
b)	-b	sqrt(-val)
c)	b	-sqrt(val)
d)	b	sqrt(-val)

Q7. From the answer group below, select the correct combination of answers to be inserted into and in the program. Here, the array index starts at 1.

The function `linearSearch` linearly searches the global array `D` using the recursive function `recursive`, and if an element with the same value as the argument `searchKey` exists, it returns that element index, otherwise -1 is returned. Assume that the array `D` has no duplicate elements.

[Program]

```
global: integer [: D ← {3, 4, 2, 5, 9}
global: integer: N ← the number of elements of D
```

```
○ integer: linearSearch(integer: searchKey)
  return recursive(searchKey, N)
```

```
○ integer: recursive(integer: searchKey, integer: index)
  if (index < 1)
    return -1
  elseif (D[index] = searchKey)
    return 
  else
    return recursive(searchKey, )
  endif
```

Answer group

	A	B
a)	1	index - 1
b)	1	N - 1
c)	1	N - index
d)	Index	index - 1
e)	Index	N - 1
f)	Index	N - index
g)	index - 1	index - 1
h)	index - 1	N - 1
i)	index - 1	N - index

Q8. From the answer group below, select the correct combination of answers to be inserted into and in the program. Here, the array index starts at 0.

The program implements a queue that has a fixed capacity and only accepts integers. If the queue is not full, the function enqueue appends the specified value to its end and returns true. Otherwise, it returns false. If the queue is not empty, the function dequeue removes an element from the queue and returns its value. Otherwise, it returns undefined. The figure illustrates the working of a queue with capacity to hold up to seven integers. Initially, the queue contains five integers, with front, rear, and count set to 3, 1, and 5, respectively. Then, enqueue(51) and dequeue() are called in that order.

index	0	1	2	3	4	5	6
item	34	undefined	undefined	17	13	27	19

enqueue(51) is called.

index	0	1	2	3	4	5	6
item	34	51	undefined	17	13	27	19

dequeue() is called.

index	0	1	2	3	4	5	6
item	34	51	undefined	undefined	13	27	19

Figure Queue with capacity of seven

[Program]

```
global: integer: capacity  $\leftarrow$  7 /* Capacity of the queue */
global: integer [:] item  $\leftarrow$  {7 undefined}
global: integer: count  $\leftarrow$  0 /* Number of elements in the queue */
global: integer: front  $\leftarrow$  0 /* Index of the front element */
global: integer: rear  $\leftarrow$  0 /* Index of the element next to the rear */
```

```
○ boolean: enqueue(integer: val)
  if (count )
    item[rear]  $\leftarrow$  val
    count  $\leftarrow$  count + 1
    rear  $\leftarrow$  (rear + 1) mod capacity
    return true
  else
    return false
  endif
```

```
○ integer: dequeue()
  integer val
  if (count > 0)
    val  $\leftarrow$  item[front]
    item[front]  $\leftarrow$  undefined
    count  $\leftarrow$  count - 1
    front  $\leftarrow$  
    return val
  else
    return undefined
  endif
```

Answer group

	A	B
a)	< 0	$(\text{front} + 1) \bmod \text{capacity}$
b)	< 0	$(\text{front} - 1) \bmod \text{capacity}$
c)	> 0	$(\text{front} + 1) \bmod \text{capacity}$
d)	> 0	$(\text{front} - 1) \bmod \text{capacity}$
e)	$< \text{capacity}$	$(\text{front} + 1) \bmod \text{capacity}$
f)	$< \text{capacity}$	$(\text{front} - 1) \bmod \text{capacity}$
g)	$> \text{capacity}$	$(\text{front} + 1) \bmod \text{capacity}$
h)	$> \text{capacity}$	$(\text{front} - 1) \bmod \text{capacity}$

- Q9.** From the answer group below, select the correct combination of answers to be inserted into and in the program.

The function `isPerfectBinaryTree` checks whether the binary tree with a given root node is perfect. In a perfect binary tree, all the internal nodes have exactly two children, and all leaf nodes are at the same level. The function returns `true` if the tree is perfect, or `false` otherwise.

The class `Node` models a node of a binary tree as described in Table 1. A `Node`-type variable holds a reference to an instance of the class `Node`.

Table 1 Class Node

Member variable	Type	Description
Key	integer	Integer value stored in a node of a binary tree.
Left	Node	A reference to the left child of the node. If the left child does not exist, it is set to <code>undefined</code> .
Right	Node	A reference to the right child of the node. If the right child does not exist, it is set to <code>undefined</code> .

Table 2 describes the functions used in the program.

Table 2 Functions

Function	Return value	Description
<code>pow(integer a, integer b)</code>	integer	Returns the result of raising a to the power of b (i.e., a^b).
<code>max(integer a, integer b)</code>	integer	Returns the maximum of the two integers a and b.

[Program]

```
○ boolean: isPerfectBinaryTree(Node: root)
    // The argument root holds a reference to the root node
    // of a binary tree.
    integer: height, size
    height ← getHeight(root)
    size ← getSize(root)
    return 

○ integer: getHeight(Node: root)
    integer: leftHeight, rightHeight
    if (root is undefined)
        return 0
    endif
    leftHeight ← getHeight(root.left)
```

```

rightHeight ← getHeight(root.right)
return 1 + max(leftHeight, rightHeight)

```

```

O integer: getSize(Node: root)
  if (root is undefined)
    return 0
  endif
  return B

```

Answer Group

	A	B
a)	size + 1 = pow(2, height)	getSize(root.left) + getSize(root.right)
b)	size + 1 = pow(2, height)	getSize(root.left) + getSize(root.right) + 1
c)	size + 1 = pow(2, height)	getSize(root.left) + getSize(root.right) - 1
d)	size = pow(2, height - 1)	getSize(root.left) + getSize(root.right)
e)	size = pow(2, height - 1)	getSize(root.left) + getSize(root.right) + 1
f)	size = pow(2, height - 1)	getSize(root.left) + getSize(root.right) - 1
g)	size = pow(2, height)	getSize(root.left) + getSize(root.right)
h)	size = pow(2, height)	getSize(root.left) + getSize(root.right) + 1
i)	size = pow(2, height)	getSize(root.left) + getSize(root.right) - 1

Q10. From the answer group below, select the correct combination of answers to be inserted into through in the program.

The function `mergeTwoSortedList` concatenates two linear sorted linked lists in a sorted order and returns one linear linked list of sorted order. Each element of the linear linked lists is represented by the class `ListElement`. The table lists the description of the class `ListElement`. `ListElement`-type variable holds a reference to an instance of the class `ListElement`. Here, `head1` and `head2` represent the headers of the two sorted linked lists, respectively and are of type `ListElement`. After concatenation of two existing linear linked lists, the function returns a linear linked list headed with `result`, which is of type `ListElement`. Both the input linked lists and returned linked list are sorted in ascending order of the member variable `val`.

Table Explanation of the member variables of the class `ListElement`

Member variable	Type	Description
<code>val</code>	<code>integer</code>	The value of an element.
<code>next</code>	<code>ListElement</code>	Reference to the instance that holds the next element in the list. If no next element exists, the status is undefined.

[Program]

```
O ListElement: mergeTwoSortedList(ListElement: head1,  
                                  ListElement: head2)  
  
  ListElement: result  
  if (head1 is undefined)  
    return head2  
  elseif (head2 is undefined)  
    return head1  
  endif  
  if ()  
    result ← head1  
    result.next ← mergeTwoSortedList(, head2)  
  else  
    result ← head2  
    result.next ← mergeTwoSortedList(head1, )  
  endif  
  return result
```

Answer group

	A	B	C
a)	head1.next is undefined	head1.next	head2.next
b)	head1.next is undefined	head2.next	head1.next
c)	head1.val $\leq$ head2.val	head1.next	head2.next
d)	head1.val $\leq$ head2.val	head2.next	head1.next
e)	head1.val = head2.val	head1.next	head2.next
f)	head1.val = head2.val	head2.next	head1.next
g)	head1.val $\geq$ head2.val	head1.next	head2.next
h)	head1.val $\geq$ head2.val	head2.next	head1.next

Q11. From the answer group below, select the correct combination of answers to be inserted into and in the program. Here, the array indexes start at 1.

Counting Sort is a non-comparison-based sorting algorithm. It is suitable for sorting a collection of objects according to keys that are small positive integers by means of counting the occurrences of key values in all data and then using those counts to place the values in their correct sorted positions.

The function `countSort` below describes a simple variant of counting sort algorithm. The function `countSort` receives 2 arguments as follows: `arr`, the array of small positive integers to be sorted, and `M`, the max value for the range of the array (from 1 to `M`), and returns the sorted array. Element `count[k]` of local array `count` holds the frequency of key value `k` in the range 1 to `M`.

For instance, when the function `countSort` is called as `countSort({3, 6, 1, 5, 3, 4, 5}, 6)`, the value of array `count` is {1, 0, 2, 1, 2, 1}, and the function returns {1, 3, 3, 4, 5, 5, 6}.

[Program]

```
○ integer [: countSort(integer [: arr, integer M)
  integer [: count ← {M zeros}
  integer: i, j, k
  integer: n ← the number of elements of arr
  integer [: result ← {n zeros}

  // counting each element of arr
  for (increase i from 1 to n by 1)
    k ← arr[i]
    count[k] ← count[k] + 1
  endfor

  // forming array result from each key value and its count
  i ← 1
  for (increase k from 1 to M by 1)
    if ()
      for (increase j from 1 to count[k] by 1)
        result[i] ← k
        
      endfor
    endif
  endfor
  return result // sorted array
```

Answer group

	A	B
a)	$\text{count}[i] > 0$	$i \leftarrow i + 1$
b)	$\text{count}[i] > 0$	$i \leftarrow j + 1$
c)	$\text{count}[i] > 0$	$i \leftarrow k + 1$
d)	$\text{count}[k] > 0$	$i \leftarrow i + 1$
e)	$\text{count}[k] > 0$	$i \leftarrow j + 1$
f)	$\text{count}[k] > 0$	$i \leftarrow k + 1$

Q12. From the answer group below, select the correct combination of answers to be inserted into

A

 and

B

 in the program. Here, the array indexes start at 1.

The function `isIsomorphic` compares the two character arrays `s1` and `s2` that are given as arguments. Both `s1` and `s2` have one or more elements.

Two character arrays are considered isomorphic if the characters in `s1` can be consistently replaced to obtain `s2` without changing the order of characters. Each character in `s1` must map to exactly one character in `s2`, and no two characters in `s1` may map to the same character in `s2`. A character may map to itself.

If `s1` and `s2` do not have the same number of elements, the function returns `false`. Otherwise, it returns `true` if `s1` and `s2` are isomorphic and `false` if they are not.

The table lists examples of `s1` and `s2` given to the function `isIsomorphic` and the return values.

Table Examples of `s1` and `s2` given to the function `isIsomorphic` and the return values

s1	s2	Return Value
{"e", "g", "g"}	{"a", "d", "d"}	true
{"f", "o", "o"}	{"b", "a", "r"}	false
{"p", "a", "p", "e", "r"}	{"t", "i", "t", "l", "e"}	true
{"a", "b"}	{"c", "c"}	false

[Program]

```

O boolean: isIsomorphic(character []: s1, character []: s2)
  integer: len, i, j
  len ← number of elements in s1
  if (number of elements in s2 ≠ len)
    return false
  endif
  for (increase i from 1 to (len - 1) by 1)
    for (increase j from (i + 1) to len by 1)
      if ((s1[i] = s1[j] and 

|   |
|---|
| A |
|---|

)
        or (s1[i] ≠ s1[j] and 

|   |
|---|
| B |
|---|

))
        return false
      endif
    endfor
  endfor
  return true

```

Answer group

	A	B
a)	$s1[i] = s2[j]$	$s1[i] \neq s2[j]$
b)	$s1[i] \neq s2[j]$	$s1[i] = s2[j]$
c)	$s2[i] = s2[j]$	$s2[i] \neq s2[j]$
d)	$s2[i] \neq s2[j]$	$s2[i] = s2[j]$

Q13. From the answer group below, select the correct answer to be inserted into in the program. Here, the array indexes start at 1.

The function `extractEmails` receives text as an argument, extracts email addresses from the text, and returns all collected email addresses. Here, a string in the format “name@domain” is considered to be an email address, where “name” and “domain” comprise only alphanumeric characters and the characters “.” (dot), “_” (underscore), and “-” (hyphen-minus). The table shows the description of the functions used in the function `extractEmails`.

Table Functions used in the function `extractEmails`

Function	Return value	Description
<code>findSubstring(</code> string: str, string: substr, integer: start)	integer	Returns the position of the first occurrence of the substring <code>substr</code> in the string <code>str</code> , starting from the position <code>start</code> . If the substring is not found, it returns 0.
<code>getChar(string: str,</code> integer: index)	character	Returns the character at the position specified by <code>index</code> in the string <code>str</code> .
<code>length(string: str)</code>	integer	Returns the length of the string <code>str</code> .

`extractEmails("Contact us at john.doe@example.com or jane_smith@test.org for support.")` returns {"john.doe@example.com", "jane_smith@test.org"}. In addition, `extractEmails("a@b@c")` returns {"a@b", "b@c"}.

[Program]

```
○ string []: extractEmails(string: txt)
  character []: validChars ← {alphanumeric characters, ".", "_", "-"}
  string []: results ← {}
  integer: index ← 1
  integer: atIndex, i, count
  string: candidate
  while (true)
    atIndex ← findSubstring(txt, "@", index)
    if (atIndex = 0)
      exit the while block
    endif
    candidate ← ""
    for ( by 1)
      if (getChar(txt, i) exists in validChars)
        prepend getChar(txt, i) to candidate
      else
        exit the for block
```

```

        endif
    endfor
    count ← 0
    if (length(candidate) > 0)
        append "@" to candidate
        for (increase i from atIndex + 1 to length(txt) by 1)
            if (getChar(txt, i) exists in validChars)
                append getChar(txt, i) to candidate
                count ← count + 1
            else
                exit the for block
            endif
        endfor
        if (count > 0)
            append the value of candidate at the end of results
        endif
    endif
    index ← atIndex + 1
endwhile
return results

```

Answer group

- a) decrease i from atIndex - 1 to index
- b) decrease i from atIndex to index
- c) increase i from index to atIndex
- d) increase i from index to atIndex - 1

Q14. From the answer group below, select the correct combination of answers to be inserted into and in the program. Here, the array indexes start at 1.

Percentile is a statistical measure used to describe the position of a particular value in a dataset relative to the remainder of the values. In education, percentiles are commonly used to rank students' performance. For instance, if a student is in the 90th percentile, it implies the score of the student is higher than 90% of the other students. To calculate percentile of a student, use the following formula:

$$\text{percentile} = \frac{\text{number of students who scored below the target student's score}}{\text{total number of students} - 1} \times 100$$

For instance, if the scores of five students are given as an array {30, 60, 80, 72, 15}, then the percentile array for each student will be {25, 50, 100, 75, 0}. This implies that the number of students with scores less than 30 is 1; thus, the percentile for the first student is $(1 \div (5 - 1) \times 100) = 25$, the number of students with scores less than 60 is 2, and the percentile for the second student is $(2 \div (5 - 1) \times 100) = 50$. The same applies thereafter.

The function `calculatePercentile` receives the scores of each student in the argument `arr` and calculates and returns the percentile of each student. Assume that the argument `arr` contains two or more numbers between 0 and 100.

[Program]

```
○ real []: calculatePercentile(integer []: arr)
  integer: n ← the number of elements of arr
  real []: percentiles ← {n zeros}
  integer: i, j, count
  for (increase i from 1 to n by 1)
    count ← 0
    for (increase j from 1 to  by 1)
      if (arr[i] > arr[j])
        count ← count + 1
      endif
    endfor
    percentiles[i] ← 
  endfor
  return percentiles
```

Answer group

	A	B
a)	i	$((\text{count} - 1) \div (n - 1)) \times 100$
b)	i	$(\text{count} \div (n - 1)) \times 100$
c)	i - 1	$((\text{count} - 1) \div (n - 1)) \times 100$
d)	i - 1	$(\text{count} \div (n - 1)) \times 100$
e)	n	$((\text{count} - 1) \div (n - 1)) \times 100$
f)	n	$(\text{count} \div (n - 1)) \times 100$

Q15. From the answer group below, select the correct combination of answers to be inserted into and in the description.

Bag of Words is a representation of text that describes the occurrence of words in a document. It is a feature extraction method in text mining. From a document collection (or corpus), a feature vector for each document can be created by considering the count of each word in that document.

The following are two documents that construct a text corpus:

document-I: John likes to watch movies. Mary likes movies too.

document-II: Mary also likes to watch football games.

Here, the word sequence, defined by the unique words (ignoring case and punctuation) in the documents, is as follows:

{John, Mary, likes, to, watch, movies, too, also, football, games}

Now, feature vectors for **document-I** and **document-II** can be created using the unique words.

Feature Vector for document-I = {1, 1, 2, 1, 1, 2, 1, 0, 0, 0}

Feature Vector for document-II = {0, 1, 1, 1, 1, 1, 0, 0, 1, 1}

Again, two new text documents and the corresponding word sequence are given below:

document-III: He is a good boy. An honest boy is good.

document-IV: He is an honest boy. The good boy is honest.

Here, the word sequence for those two documents is {He, is, a, an, good, boy, honest, the}. Then, the Feature Vectors for **document-III** and **document-IV** will be and , respectively.

Answer group

	A	B
a)	{1, 1, 1, 1, 2, 2, 1, 0}	{1, 1, 0, 1, 1, 2, 2, 1}
b)	{1, 1, 1, 1, 2, 2, 1, 0}	{1, 2, 0, 1, 1, 1, 1, 1}
c)	{1, 1, 1, 1, 2, 2, 1, 0}	{1, 2, 0, 1, 1, 2, 2, 1}
d)	{1, 2, 1, 1, 1, 1, 1, 0}	{1, 1, 0, 1, 1, 2, 2, 1}
e)	{1, 2, 1, 1, 1, 1, 1, 0}	{1, 2, 0, 1, 1, 1, 1, 1}
f)	{1, 2, 1, 1, 1, 1, 1, 0}	{1, 2, 0, 1, 1, 2, 2, 1}
g)	{1, 2, 1, 1, 2, 2, 1, 0}	{1, 1, 0, 1, 1, 2, 2, 1}
h)	{1, 2, 1, 1, 2, 2, 1, 0}	{1, 2, 0, 1, 1, 1, 1, 1}
i)	{1, 2, 1, 1, 2, 2, 1, 0}	{1, 2, 0, 1, 1, 2, 2, 1}

Q16. From the answer group below, select the correct answer to be inserted into in the program. The same answer goes into both blanks . Here, the array index starts at 1.

The program converts the code point of a Unicode character to a 2-byte UTF-8 encoding. In this question, “₍₁₆₎” after a numerical value indicates a hexadecimal value. Each Unicode character is given an integer value that is known as a code point. A character with a code point that ranges from 80₍₁₆₎ to 7FF₍₁₆₎ is encoded to a 2-byte value as below.

Let the bit pattern with a 2-byte length be: 110xxxxx 10xxxxxx. The underlined 11 “x” positions in the bit pattern store the 11-bit code point. The code point is justified to the right, and 0 is stored in any leftover “x” positions. This 2-byte value is the encoded result. For instance, when the code point AE₍₁₆₎ for Registered Sign character “®” is represented in binary, it is 10101110. When this is stored right-justified in the “x” positions in the bit pattern above, it is 110xxx10 10101110. When 0 is stored in the three leftover “x” positions, the UTF-8 encoding for Registered Sign character “®” 11000010 10101110 is obtained. In decimal form, this corresponds to {194, 174}.

The function encode converts a Unicode code point that is passed as an argument to UTF-8 encoding, and returns an integer array that stores it one byte per element from the start of the array. It is assumed that only an integer value that ranges from 80₍₁₆₎ to 7FF₍₁₆₎ is passed to encode as an argument.

[Program]

```
○ integer []: encode(integer: codePoint)
  /* the initial value of utf8Bytes is the value when “x”s in the bit pattern are replaced
    with 0, divided into two 8-bit blocks, and each deemed to be binary */
  integer []: utf8Bytes ← {192, 128}
  utf8Bytes[2] ← utf8Bytes[2] + (codePoint mod )
  utf8Bytes[1] ← utf8Bytes[1] + integer part of (codePoint ÷ )
  return utf8Bytes
```

Answer group

- a) 2
- b) 32
- c) 64
- d) 256
- e) `utf8Bytes[1]`
- f) `utf8Bytes[2]`

Q17. From the answer group below, select the most appropriate combination of answers to be inserted into and in the description.

Company P conducts training sessions for junior security analysts within its security team. As part of this training activity, the team reviews simulated authentication log data and discusses how different password attack techniques may manifest in real-world systems. The purpose of this exercise is to help analysts accurately identify attack patterns, understand attacker behavior, and evaluate the defensive controls that are effective against each type of attack.

During the exercise, the team focuses on three major password attack techniques, understanding that their visibility in authentication logs varies as summarized in Table 1.

Table 1 Major password attack techniques

Type	Characteristics
brute force attack	Repeatedly attempting numerous different passwords against a single user account. It typically generates several failed authentication attempts for the same username within a short period of time.
credential stuffing attack	Attempting to log in using previously compromised username-password pairs. This often results in authentication attempts against numerous different user accounts, with some attempts succeeding.
offline password cracking	Attempting to recover passwords by cracking stolen password hashes offline, without interacting with the authentication system. This attack does not generate authentication logs at an authentication server.

As part of the training, the security team reviews multiple sample authentication logs recorded at an authentication server, as summarized in Tables 2 and 3. The participants analyze the authentication patterns observed in both tables and determine the correspondence of each pattern corresponds to the password attack techniques listed in Table 1.

Table 2 Sample authentication logs (Pattern A)

Timestamp	Source IP	Username	Event	Result / Reason
2025-03-12 11:30:00	203.0.113.30	admin	auth_fail	InvalidPassword
2025-03-12 11:32:00	203.0.113.30	admin	auth_fail	InvalidPassword
2025-03-12 11:34:00	203.0.113.30	admin	auth_fail	InvalidPassword
2025-03-12 11:36:00	203.0.113.30	admin	auth_fail	InvalidPassword
2025-03-12 11:38:00	203.0.113.30	admin	auth_fail	InvalidPassword
2025-03-12 11:40:00	203.0.113.30	admin	auth_fail	InvalidPassword

Table 3 Sample authentication logs (Pattern B)

Timestamp	Source IP	Username	Event	Result / Reason
2025-03-12 11:05:00	198.51.100.45	john.smith	auth_fail	InvalidPassword
2025-03-12 11:07:30	198.51.100.45	emma.jones	auth_success	Authenticated
2025-03-12 11:10:00	198.51.100.45	michael.brown	auth_fail	InvalidPassword
2025-03-12 11:12:40	198.51.100.45	sarah.miller	auth_fail	InvalidPassword
2025-03-12 11:15:10	198.51.100.45	kevin.wilson	auth_fail	InvalidPassword
2025-03-12 11:17:50	198.51.100.45	olivia.davis	auth_success	Authenticated

Based on this analysis, they conclude that the pattern observed in Table 2 is categorized as A, and that observed in Table 3 is categorized as B, as summarized in Table 1.

Answer group

	A	B
a)	brute force attack	credential stuffing attack
b)	brute force attack	offline password cracking
c)	credential stuffing attack	brute force attack
d)	credential stuffing attack	offline password cracking
e)	offline password cracking	brute force attack
f)	offline password cracking	credential stuffing attack

Q18. From the answer group below, select the most appropriate combination of answers to be inserted into and in the description.

A research laboratory at University Q plans to introduce a cloud-based database to centrally store and manage data for one of its research projects. This database will be used by professors, researchers, students, and administrative staff participating in the project. Table 1 summarizes the data types, descriptions, and the roles and permitted operations for each user group.

Table 1 Data Types and User Requirements (Excerpt)

Data Type	Description	User Roles
Research Proposals	Detailed research plans and objectives	- Professors and Researchers create and edit project proposals. - Students can view the proposals to understand the project objectives. - Administrative staff are not permitted to access this data.
Experimental Data	Raw and processed experimental results	- To maintain data integrity, only Researchers are allowed to create and modify data. - Professors and Students review the data to understand experiment results and research progress. - Administrative staff are not permitted to access this data.

The system administrator at University Q, who is responsible for introducing the database, reviewed the data types and user roles with the information security officer of the university. According to the information security policy of University Q, users must be granted only the permissions necessary for their roles to minimize potential damage in the event of account compromise. Based on this policy, the system administrator created user groups and configured access permissions for each content type. Table 2 lists the resulting access right assignments for the database.

Table 2 Access Right Assignments (Excerpt)

Data Type	Professors	Researchers	Students	Administrative Staff
Research Proposals	A			(-)
Experimental Data	B			(-)

Note:

R: Read-only (viewing allowed)

RW: Read and Write (includes creating and modifying)

(-): Not allowed (no access)

Answer group

	A			B		
	Professors	Researchers	Students	Professors	Researchers	Students
a)	R	RW	R	R	RW	R
b)	R	RW	R	RW	RW	(-)
c)	RW	RW	R	R	RW	R
d)	RW	RW	R	RW	RW	(-)
e)	RW	RW	(-)	R	RW	R
f)	RW	RW	(-)	RW	RW	(-)

Q19. From the answer group below, select the most appropriate combination of answers to be inserted into and in the description.

Company R is an e-commerce company that sells smartphone accessories through its website. In response to the increasing prevalence of SQL injection attacks in recent years, Company R decided to conduct a security training program for its security analysts, focusing specifically on SQL injection vulnerabilities.

[Explanation of SQL Behavior]

As part of the training, an explanation was provided to help the security analysts understand the working mechanism of SQL statements. In typical web applications, authentication is performed by constructing an SQL statement based on a predefined template. The application inserts the user-provided username and the hashed value of the user-provided password into the template and submits the resulting SQL statement to the database to check whether a matching record exists. An example of such an SQL query template is shown in figure given below.

```
SELECT * FROM users
WHERE username = '<input_username>'
      AND password = '<hashed_input_password>'
```

Note: The SQL query constructed from this template is executed by the database engine as a single SQL statement. Line breaks in the figure are included solely for readability.

Figure SQL Query Template for Authentication

If user input is incorporated into an SQL statement without proper handling, the structure of the SQL query may be altered, enabling attacks such as SQL injection. For instance, SQL syntax elements included in the input may be interpreted by the database. One common element is “--”, which starts a comment; any text that follows “--” is ignored by the database engine, effectively disabling the remaining part of the SQL statement, including the password check.

[Hands-on Verification]

The security analysts participated in a hands-on lab using a dedicated training server. They submitted crafted input values through the application’s authentication form. The application processed these inputs, inserted the username and the hashed password value into the SQL template shown in the figure above, generated a complete SQL statement, and executed it against the database.

The database contained a preregistered admin user with a hashed password that was not disclosed to the participants. They tested several input values for <input_username> while leaving the password input field empty in the authentication form to determine whether authentication could be bypassed, and summarized the results in table given below.

Table Evaluation Results

Input Values for <input_username>	Authentication Bypass Results
Admin	Fail
admin' --	A
admin' AND '1'='2' --	B

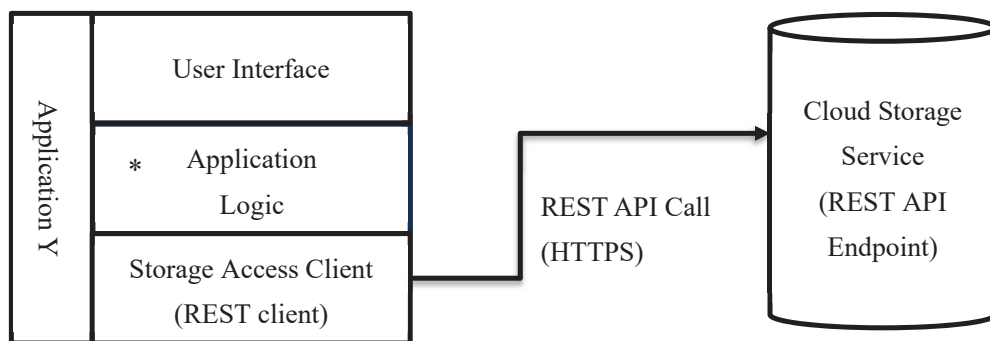
Answer Group

	A	B
a)	Fail	Fail
b)	Fail	Success
c)	Success	Fail
d)	Success	Success

Q20. From the answer group below, select the most appropriate answer to be inserted into in the description.

Company S had recently released a photo editing application named Application Y for mobile platforms.

As storage for users to upload the photos they wish to process, Application Y utilizes a cloud storage service. The cloud storage service exposes REST API endpoints that accept requests from REST client. Application Y makes direct REST API calls to the cloud storage service, and the access tokens required to invoke these APIs are embedded directly within the application logic. Figure 1 shows the portion of the Application Y architecture related to access to the cloud storage service. Other components are omitted for simplicity.



Note: * indicates the location where the access token used to invoke REST API calls is stored

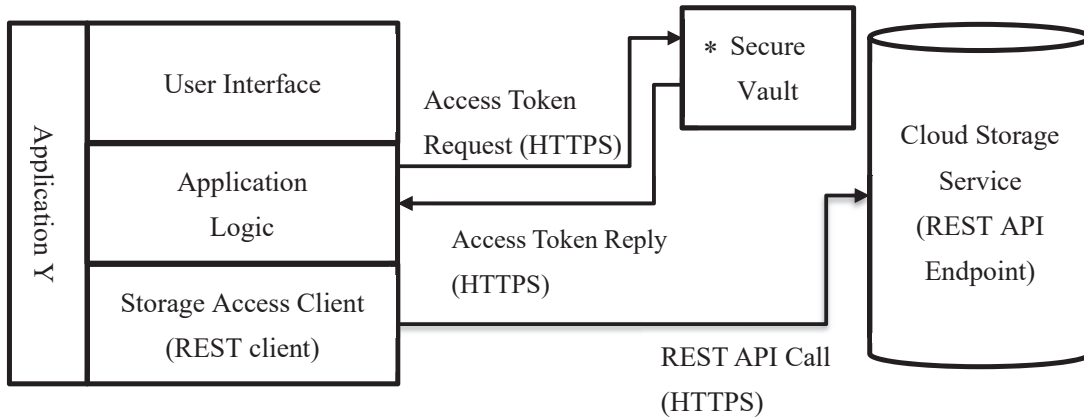
Figure 1 Application Y architecture related to access to the backend storage service

A security researcher discovered that the access token embedded in Application Y could be extracted from Application Y. The researcher was able to invoke the REST API directly and access the storage service using the extracted token.

The developer team of Company S proposed the following for architectures to prevent future exposure of storage access tokens.

Architecture One: Use a Secure Vault to manage access tokens.

In this architecture, access tokens are stored in a Secure Vault, which is a digital safe for protecting sensitive data such as API access tokens. When Application Y needs to access the storage service, it retrieves a temporary token from the Secure Vault and uses the token to make REST API calls directly to the cloud storage service over HTTPS.

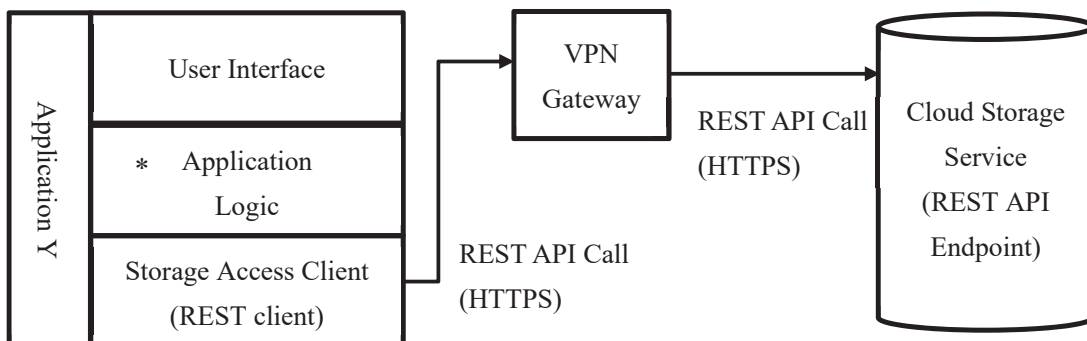


Note: * indicates the location where the access token used to invoke REST API calls is stored

Figure 2 Architecture One

Architecture Two: Use a VPN to secure communication.

In this architecture, Application Y establishes a VPN connection to the cloud environment. All REST API calls from Application Y to the cloud storage service are transmitted through the VPN. The client application continues to use an access token to invoke the REST APIs of the storage service.

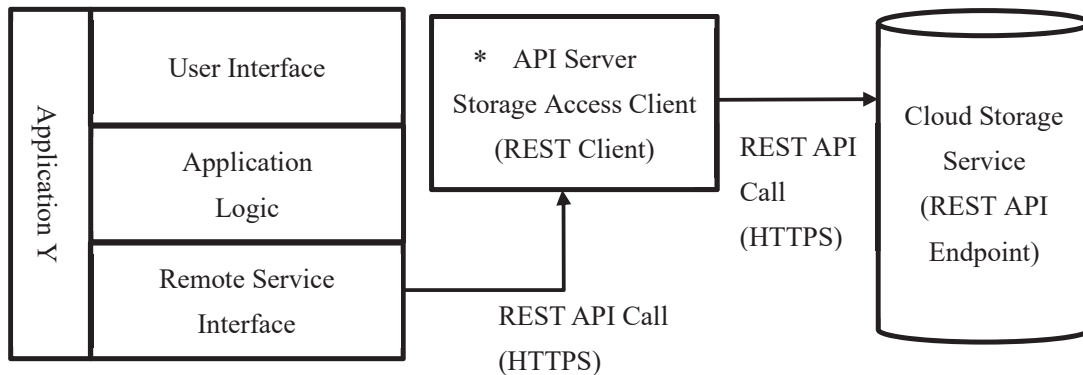


Note: * indicates the location where the access token used to invoke REST API calls is stored

Figure 3 Architecture Two

Architecture Three: Use an API Server.

In this architecture, an API server is introduced between Application Y and the cloud storage service. The API server securely stores the access tokens and performs REST API Calls. Application Y communicates only with the API server over HTTPS and does not possess access tokens for the cloud storage service. All authentication, authorization, and business logic are enforced on the API server, which performs REST API calls to the storage service on behalf of the client.

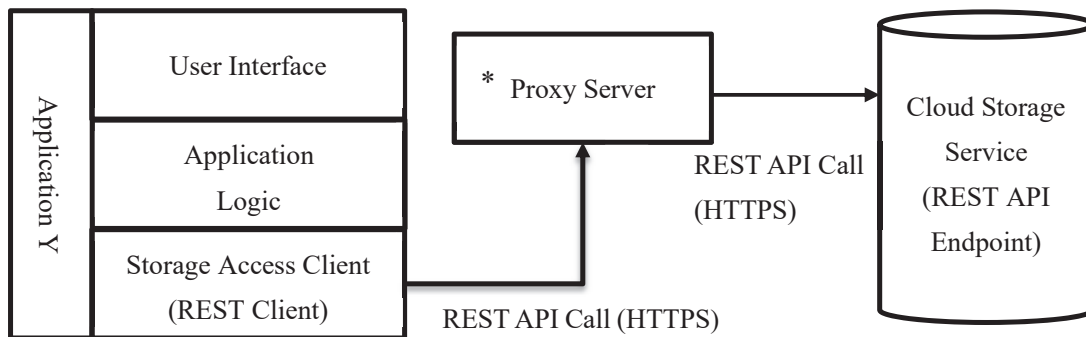


Note: * indicates the location where the access token used to invoke REST API calls is stored

Figure 4 Architecture Three

Architecture Four: Use a proxy server.

In this architecture, a proxy server is deployed between Application Y and the cloud storage service. The proxy server stores the access credentials required to access the storage service. Application Y sends REST API requests to the proxy server over HTTPS, and the proxy server forwards these requests to the cloud storage service using the stored credentials. The proxy server does not perform application-specific authentication, authorization, or business logic enforcement, and simply relays REST API requests from the client to the storage service.



Note: * indicates the location where the access token used to invoke REST API calls is stored

Figure 5 Architecture Four

By comparing them, the security team of Company S suggested the application development team redesign the application architecture to Architecture .

Answer group

- a) One
- b) Two
- c) Three
- d) Four

Company names and product names appearing in the test questions are trademarks or registered trademarks of their respective companies. Note that the ® and ™ symbols are not used within the text.