Information Technology Engineers Examination

Information Technology Terms and Specifications of Programming Languages Used in Examination Questions

Ver 2.2

May 22, 2012

INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN

1. Terms on information technology

The terms on information technology used in the examination shall conform to the definitions by Japanese Industrial Standards (JIS), when they are specified in the JIS.

2. Symbols and figures

The definitions of main symbols and figures used in examinations shall conform to the following specifications. Those not defined in the following are defined in notes that appear in examination questions.

Documentation symbols and conventions for data, program and system flowcharts, etc	: JIS X 0121
Decision tables	: JIS X 0125
Computer system configuration diagram symbols	: JIS X 0127
Program constructs and conventions for their representation	: JIS X 0128

3. Programming languages

(1) Programming languages given as the questions on Software Development in Fundamental Information Technology Engineers Examination are the C, COBOL, Java and assembly languages (In addition to the 4 languages, a question on spreadsheet is given).

(2) Programming language given as the questions in Information Security Specialist Examination are the C++, Java and ECMAScript.

(3) The specifications shall conform to the following:

C	: JIS X 3010
COBOL	: JIS X 3002
Java	: The Java Languages Specification, Third Edition (JLS) ([Note1]) (URL http://docs.oracle.com/javase/specs/)
Assembly language	: ANNEX 1 "Specifications for Assembly Language" given on page 3
C++	: JIS X 3014
ECMAScript	: JIS X 3060

(Note 1) Note that the following are mainly used in the question, as newly-added features in JLS 3.0. However, it is not limited to these 3.

- Generics
- Extended '`for`' statement
- Enum types

Also note that meta-data is out of scope.

[1] "Java" is a registered trademark of Oracle Corporation and its subsidiary in the United States and other countries.

4. Database language

The database language used in the examinations shall conform to the following specification:

SQL : JIS X 3005 standards

5. Markup Languages

Markup languages used in the examinations shall conform the following specifications:

HTML : JIS X 4156

XML : JIS X 4159

6. Software packages including spreadsheet software

Spreadsheet software : ANNEX 2 “Spreadsheet Software Functions and Terminology (for IT Passport Examination)” given on page 11, and ANNEX 3 “Spreadsheet Software Functions and Terminology (for Fundamental Information Technology Engineers Examination)” given on page 16.

“Spreadsheet Software Functions and Terminology (for Fundamental Information Technology Engineers Examination)” includes the content of “Spreadsheet Software Functions and Terminology (for IT Passport Examination)”. Functions and terms not defined in the ANNEX 2 and ANNEX 3 are defined in notes that appear in the examination questions.

Functions and terms of software packages other than spreadsheets are defined in notes that appear in examination questions.

< Reference of JIS (website of Japanese Industrial Standards Committee) >

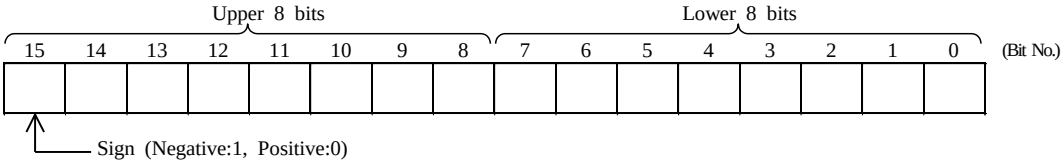
URL: http://www.jisc.go.jp/

ANNEX 1 Specifications for Assembly Language

1. COMET II Specifications

1.1 Hardware Specifications

(1) One word is 16 bits, and the bit format is as follows:



(2) Main storage capacity is 65,536 words with address numbers 0 through 65,535.

(3) Numeric values are expressed as 16-bit binary numbers. Negative numbers are expressed in complements of two.

(4) Control is sequential. COMET II utilizes a one-word or two-word instruction word.

(5) The COMET II has four types of registers: GR (16 bits), SP (16 bits), PR (16 bits) and FR (3 bits).

There are eight GR (General Register) registers, GR0 through GR7. These eight registers are used for arithmetic, logical, compare and shift operations. Of these, GR1 through GR7 are also used as index registers to modify addresses.

The stack pointer stores the address currently at the top of the stack.

The PR (Program Register) stores the first address of the next instruction.

The FR (Flag Register) consists of three bits: OF (Overflow Flag), SF (Sign Flag) and ZF (Zero Flag). The following values are set, depending on the result generated by certain operation instructions. These values are referenced by conditional branch instructions.

OF	When the result of an arithmetic operation instruction is out of the range of –32,768 to 32,767, the value is 1, and in other cases, the value is 0. When the result of a logical operation instruction is out of the range of 0 to 65,535, the value is 1, and in other cases, the value is 0.
SF	When the sign of the operation result is negative (bit number 15 is 1), the value is 1, and in other cases, the value is 0.
ZF	When the operation result is 0 (all bits are 0), the value is 1, and in other cases, the value is 0.

(6) Logical addition or logical subtraction: Treats the data to be added or subtracted as unsigned data, and performs addition or subtraction.

1.2 Instructions

Formats and functions of instructions are described in the following chart. When an instruction code has two types of operands, the upper operand shows the instruction between registers and the lower operand shows the instruction between register and main storage.

Instruction	Format		Description of instructions	FR setting
	Opcode	Operand		
(1) Load, store, load address instruction				
LoaD	LD	r1,r2	r1 ← (r2)	○*1
		r,adr [,x]	r ← (effective address)	
STore	ST	r,adr [,x]	Effective address ← (r)	–
Load ADdress	LAD	r,adr [,x]	r ← effective address	
(2) Arithmetic and logical operation instructions				
ADD Arithmetic	ADDA	r1,r2	r1 ← (r1) + (r2)	○
		r,adr [,x]	r ← (r) + (effective address)	
ADD Logical	ADDL	r1,r2	r1 ← (r1) $+_L$ (r2)	
		r,adr [,x]	r ← (r) $+_L$ (effective address)	
SUBtract Arithmetic	SUBA	r1,r2	r1 ← (r1) – (r2)	
		r,adr [,x]	r ← (r) – (effective address)	
SUBtract Logical	SUBL	r1,r2	r1 ← (r1) $-_L$ (r2)	
		r,adr [,x]	r ← (r) $-_L$ (effective address)	
AND	AND	r1,r2	r1 ← (r1) AND (r2)	○*1
		r,adr [,x]	r ← (r) AND (effective address)	
OR	OR	r1,r2	r1 ← (r1) OR (r2)	
		r,adr [,x]	r ← (r) OR (effective address)	
eXclusive OR	XOR	r1,r2	r1 ← (r1) XOR (r2)	
		r,adr [,x]	r ← (r) XOR (effective address)	

(3) Compare operation instructions

ComPare Arithmetic	CPA	r1,r2 r,adr [,x]	Performs an arithmetic compare or logical compare operation on (r1) and (r2) or (r) and (effective address), and sets FR as follows, according to the result of the compare operation.	○*1
ComPare Logical	CPL	r1,r2 r,adr [,x]		

Compare result	FR value	
	SF	ZF
(r1) > (r2) (r) > (effective address)	0	0
(r1) = (r2) (r) = (effective address)	0	1
(r1) < (r2) (r) < (effective address)	1	0

(4) Shift operation instructions

Shift Left Arithmetic	SLA	r,adr [,x]	Shifts (r) (excluding the sign bit) left or right by the number of bits specified by the effective address. When a left shift is performed, those bits that are left vacant by the shift operation are filled with zeroes. When a right shift is performed, those bits that are left vacant by the shift operation are filled with the same value as the sign bit.	○*2
Shift Right Arithmetic	SRA	r,adr [,x]		
Shift Left Logical	SLL	r,adr [,x]	Shifts (r) (including the sign bit) left or right by the number of bits specified by the effective address. Those bits that are left vacant by the shift operation are filled with zeroes.	
Shift Right Logical	SRL	r,adr [,x]		

(5) Branch instructions

Jump on PLus	JPL	adr [,x]	Branches to the effective address, depending on the value of FR. If control does not branch to a new address, execution continues with the next instruction.	–
Jump on MInus	JMI	adr [,x]		
Jump on Non Zero	JNZ	adr [,x]		
Jump on ZEro	JZE	adr [,x]		
Jump on OVerflow	JOV	adr [,x]		
unconditional JUMP	JUMP	adr [,x]	Branches unconditionally to the effective address.	

Instruc-tion	Value of FR in order to branch		
	OF	SF	ZF
JPL		0	0
JMI		1	
JNZ			0
JZE			1
JOV	1		

(6) Stack operation instructions

PUSH	PUSH	adr [,x]	SP ← (SP) $-_L$ 1, (SP) ← effective address	–
POP	POP	r	r ← ((SP)), SP ← (SP) $+_L$ 1	

(7) Call and return instructions

CALL subroutine	CALL	adr [,x]	SP ← (SP) $-_L$ 1, (SP) ← (PR), PR ← effective address	–
RETurn from subroutine	RET		PR ← ((SP)), SP ← (SP) $+_L$ 1	

(8) Other

SuperVisor Call	SVC	adr [,x]	Determines based on the effective address as the argument. After the execution, GR and FR are undefined.	–
No OPeration	NOP		N/A	

(Note)	r, r1, r2	All of these represent GR. Values from GR0 to GR7 can be specified.
	adr	This represents the address. A value from 0 to 65,535 can be specified.
	x	This represents GR used as the index register. A value from GR1 to GR7 can be specified.
	[ ]	These [ ] represents specifications in the brackets may be omitted.
	( )	These ( ) represents the register or address.
	Effective address	A value produced by adding, through "logical addition," adr and the contents of x, or the address pointed at by that value.
	←	This means that the operation result is stored in the left part register or address.
	$+_L$, $-_L$	Logical addition and logical subtraction.
	Effective address for FR setting	○ : Setting is performed. ○*1 : Setting is performed, but 0 is set to OF. ○*2 : Setting is performed, but the bit value sent lastly from the register is set to OF. – : The value before execution is stored.

1.3 Character Set

(1) A JIS X0201 Romaji/katakana character set that uses 8-bit codes is used.
(2) Part of the character set is shown in the right table. Eight bits are used to represent one character; the upper four bits indicate the column in the table, and the lower four bits indicate the row. For example, the hexadecimal codes for the space character, "4," "H," and "¥" are 20, 34, 48 and 5C, respectively. The characters that correspond to the hexadecimal codes 21 to 7E (and A1 to DF omitted in this table) are called "graphic characters." Graphic characters can be displayed (printed) as characters on an output device.
(3) If any characters not listed in this table and the bit configurations for those characters are needed, they are given in the questions.

	02	03	04	05	06	07
0	Space	0	@	P	`	p
1	!	1	A	Q	a	q
2	”	2	B	R	b	r
3	#	3	C	S	c	s
4	$	4	D	T	d	t
5	%	5	E	U	e	u
6	&	6	F	V	f	v
7	'	7	G	W	g	w
8	(	8	H	X	h	x
9	)	9	I	Y	i	y
10	*	:	J	Z	j	z
11	+	;	K	⌊	k	{
12	,	<	L	¥	l	|
13	-	=	M	⌋	m	}
14	.	>	N	^	n	~
15	/	?	O	_	o	

Specifications of the CASL II Assembly Language

2.1 Specifications of the language

(1) CASL II is an assembly language for the COMET II.
(2) A program consists of instruction lines and comment lines.
(3) One instruction is described in one instruction line, and cannot continue to the next line.
(4) Instruction lines and comment lines are written from the first character of the line in the following description formats:

Line type		Description format
Instruction line	With operand	[label]{blank}{instruction code}{blank}{operand}[{blank}[comment]]
	Without operand	[label]{blank}{instruction code}[{blank}[{;}[comment]]]
Comment line		[blank]{;}[comment]

(Note)	[ ]	These ([ ]) represents specifications in the brackets may be omitted.
	{ }	These ({ }) represents specifications in the braces is mandatory.
	Label	Label is the name used to refer to the address of (the first word of) the instruction from other instructions and programs. A label must be 1 to 8 characters in length, and the leading character must be an uppercase alphabetic letter. Either uppercase alphabetic letters or numeric characters can be used for the subsequent characters. Reserved words, GR0 through GR7, are not available.
	Blank	One or more space characters.
	Instruction code	The description format is defined by instruction.
	Operand	The description format is defined by instruction.
	Comment	Optional information such as memorandums that can be written in any characters allowed by the processing system.

2.2 Instruction Types

CASL II consists of four assembler instructions (START, END, DS and DC), two macro instructions (IN and OUT) and machine language instructions (COMET II instructions). The specifications are as follows:

Instruction type	Label	Instruction code	Operand	Function
Assembler instruction	Label	START	[Execution start address]	Defines the top of a program. Defines the starting address for execution of a program. Defines the entry name for reference in other programs.
		END		Defines the end of a program.
	[label]	DS	Word length	Allocates an area.
	[label]	DC	Constant[, constant]···	Defines a constant.
Macro instruction	[label]	IN	Input area, input character length area	Input character data from input devices.
	[label]	OUT	Output area, output character length area	Output character data from output devices.
Machine language instruction	[label]	(See "1.2 Instructions")		

2.3 Assembler Instructions

Assembler instructions are used for assembler control, etc.

(1)

START	[Execution start address]

The START instruction defines the top of a program.

The label name that is defined within this program specifies the execution start address. If the label is specified, execution begins from the address, and if the label is omitted, execution begins from the next instruction of the START instruction.

The label for this instruction can be referred to from other programs as the entry name.

(2)

END	

The END instruction defines the end of a program.

(3)

DS	Word length

The DS instruction allocates an area of the specified word length.

The word length is specified by a decimal constant (≥ 0). If "0" is specified for the word length of an area, the area is not allocated, but the label is valid.

(4)

DC	Constant[, constant] ···

The DC instruction stores the data that has been specified as a constant in (consecutive) words.

There are four types of constants: decimal constants, hexadecimal constants, character constants and address constants.

Type of constant	Format	Description of instruction
Decimal constant	n	This instruction stores the decimal value specified by "n" as one word of binary data. If "n" is outside of the range of –32,768 to 32,767, only the lower 16 bits of n are stored.
Hexadecimal constant	#h	Assume "h" is a four-digit hexadecimal number. (Hexadecimal notation uses 0 through 9 and A through F.) This instruction stores the hexadecimal value specified by "h" as one word of binary data. (0000 ≤ h ≤ FFFF)
Character constant	'character string'	This instruction allocates a continuous area for the number of characters (> 0) in the character string. The first character is stored in bits 8 through 15 of the first word, the second character is stored in bits 8 through 15 of the second word, and so on, so that the character data is stored sequentially in memory. Bits 0 through 7 of each word are filled with zeroes. Spaces and any of the graphics characters can be written in a character string. Apostrophes (') must be written twice consecutively.
Address constant	Label	This instruction stores an address corresponding to the label name as one word of binary data.

2.4 Macro Instructions

Macro instructions use a pre-defined group of instructions and operand data to generate a group of instructions that performs a desired function (the word length is undefined).

(1)

IN	Input area, input character length area

The IN instruction reads one record of character data from a previously assigned input device.

The input area operand should be the label of a 256-word work area, and the input data is input in this area beginning at the starting address, one character per word. No record delimiter code (such as a line return code, when using a keyboard) is stored. The storage format is the same as character constants with the DC instruction. If the input data is less than 256 characters long, the previous data is left as is in the remaining portion of the input area. If the input data exceeds 256 characters, the excess characters are ignored.

The input character length area should be the label of the one-word work area, and the character length that was input (≥ 0) is stored

as binary data. If the end-of-file indicator is encountered, -1 is stored.

When the IN instruction is executed, the contents of GR registers are saved but the contents of FR are undefined.

(2)

OUT	Output area, output character length area

The OUT instruction writes character data as one record of data to the previously assigned output device.

The output area operand should be the label of the area where the data to be output is stored, one character per word. The storage format is the same as character constants with the DC instruction. Bits 0 through 7 do not have to be zeroes because the OS ignores them.

The output character length area should be the label of the one-word work area, and the character length that is to be output (≥ 0) is stored as binary data.

When the OUT instruction is executed, the contents of the GR registers are saved but the contents of FR are undefined.

(3)

RPUSH	

The RPUSH instruction stores the contents of GR in the stack in order of GR1, GR2, … and GR7.

(4)

RPOP	

The RPOP instruction takes out of the contents of the stack sequentially, and stores in GR in order of GR7, GR6, … and GR1.

2.5 Machine Language Instructions

Operands of machine language instructions are described in the following formats:

r, r1, r2 — GR is specified using a symbol from GR0 to GR7.

x — GR used as the index register can be specified by a symbol from GR1 to GR7.

adr — The address is specified by a decimal constant, a hexadecimal constant, an address constant or a literal. A literal can be described by attaching the equal sign (=) before a decimal constant, a hexadecimal constant or a character constant. CASL II generates a DC instruction by specifying the constant after the equal sign as the operand, and sets the address to the adr value.

2.6 Other

(1) The relative positions of the instruction words and areas generated by the assembler conform to the order of the descriptions in the assembly language program. All DC instructions generated from literals are located just before the END instruction.

(2) The instruction words and areas that are generated occupy a continuous area in the main memory.

3. Guide to Program Execution

3.1 OS

The following arrangements exist regarding program execution.

(1) The assembler interprets undefined labels (of those labels written in the operand column, any that are not defined within the program) as entry names (START instruction labels) for other programs. In this case, the assembler refrains from determining the address and entrusts that task to the OS. Before executing the program, the OS performs link processing with entry names for other programs and determines the addresses (program linking).

(2) The program is started up by the OS. Although the area in the main memory where a program is loaded is undefined, the address value corresponding to the label in the program is corrected to the actual address by the OS.

(3) During program startup, the OS allocates enough stack area for the program, then adds one to the last address and sets that value in the SP.

(4) The OS passes control to the program by the CALL instruction. When returning control to the OS after executing the program, the RET instruction is used.

(5) The assignment of an input device to the IN instruction or of an output device to the OUT instruction is made by the user before executing the program.

(6) The OS handles the differences that may arise in input and output procedures due to the different I/O devices and media involved; I/O is performed using the system's standard format and procedures (including error handling). Therefore, the user of these IN and OUT instructions does not need to be concerned with differences among I/O devices.

3.2 Undefined Items

Ensure that any items concerning program execution that are not defined in these specifications are handled by the processing system.

Reference data

Reference data given below is provided to promote a better understanding of COMET II or to support the person responsible for creating processors for COMET II. Therefore, the following reference data does

not affect the specifications for COMET II or CASL II.

1. The structure of commands

Although the structure of commands is not defined, the structures shown below are assumed. Here, OP numerical values are shown as hexadecimal values.

15 11 7 3 0 15 0 ← Bit number

First word				Second word	Length of a command	Correspondence between commands and the assembler		
OP		r/r1	x/r2	adr		Machine language command		Meaning
Main OP	Sub OP							
0	0	-	-	-	1	NOP		no operation
1	0				2	LD	r, adr, x	load
	1				2	ST	r, adr, x	store
	2				2	LAD	r, adr, x	load address
	4			-	1	LD	r1, r2	load
2	0				2	ADDA	r, adr, x	add arithmetic
	1				2	SUBA	r, adr, x	subtract arithmetic
	2				2	ADDL	r, adr, x	add logical
	3				2	SUBL	r, adr, x	subtract logical
	4			-	1	ADDA	r1, r2	add arithmetic
	5			-	1	SUBA	r1, r2	subtract arithmetic
	6			-	1	ADDL	r1, r2	add logical
	7			-	1	SUBL	r1, r2	subtract logical
3	0				2	AND	r, adr, x	and
	1				2	OR	r, adr, x	or
	2				2	XOR	r, adr, x	exclusive or
	4			-	1	AND	r1, r2	and
	5			-	1	OR	r1, r2	or
	6			-	1	XOR	r1, r2	exclusive or
4	0				2	CPA	r, adr, x	compare arithmetic
	1				2	CPL	r, adr, x	compare logical
	4			-	1	CPA	r1, r2	compare arithmetic
	5			-	1	CPL	r1, r2	compare logical
5	0				2	SLA	r, adr, x	shift left arithmetic
	1				2	SRA	r, adr, x	shift right arithmetic
	2				2	SLL	r, adr, x	shift left logical
	3				2	SRL	r, adr, x	shift right logical
6	1	-			2	JMI	adr, x	jump on minus
	2	-			2	JNZ	adr, x	jump on non zero
	3	-			2	JZE	adr, x	jump on zero
	4	-			2	JUMP	adr, x	unconditional jump
	5	-			2	JPL	adr, x	jump on plus
	6	-			2	JOV	adr, x	jump on overflow
7	0	-			2	PUSH	adr, x	push
	1		-	-	1	POP	r	pop
8	0	-			2	CALL	adr, x	call subroutine
	1	-	-	-	1	RET		return form subroutine
9 - E						Other commands		
F	0	-			2	SVC	adr, x	supervisor call

2. Macro commands

Although command groups generated by macro commands are not defined (the number of words is indeterminate), the generation of command groups shown below is assumed:

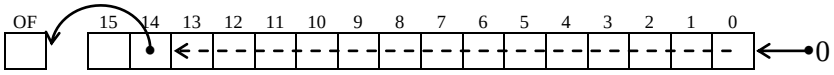
Example: IN command

```
LABEL    IN       IBUF, LEN
          ↓Macro generation
LABEL    PUSH     0, GR1
         PUSH     0, GR2
         LAD      GR1, IBUF
         LAD      GR2, LEN
         SVC      1
         POP      GR2
         POP      GR1
```

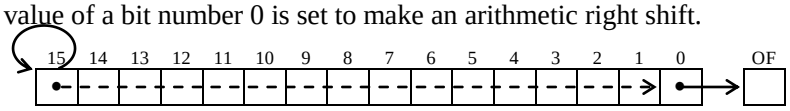
3. Bit movements when an operational shift statement is issued

When an operational shift statement is issued to achieve a one-bit shift, bits move and the OF changes, as shown below:

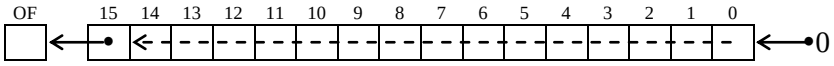
(1) The value of a bit number 14 is set to make an arithmetic left shift.



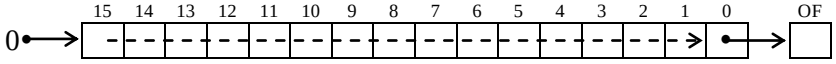
(2) The value of a bit number 0 is set to make an arithmetic right shift.



(3) The value of a bit number 15 is set to make a logical left shift.



(4) The value of a bit number 0 is set to make a logical right shift.



4. Sample program

```
COUNT1  START                   ;
;       Input       GR1: Word to search
;       Processing  Identifying the number of bits set to '1' in GR1
;       Output      GR0: The number of bits set to '1' in GR1
        PUSH    0, GR1          ;
        PUSH    0, GR2          ;
        SUBA    GR2, GR2        ; Count = 0
        AND     GR1, GR1        ; All bits are '0'?
        JZE     RETURN          ; If all of the bits are '0,' the routine ends.
MORE    LAD     GR2, 1, FR2     ; Count = Count +1
        LAD     GR0, -1, GR1    ; The least significant one bit set to '1' is
        AND     GR1, GR0        ; changed to '0.'
        JNZ     MORE            ; If bits set to '1' remain, the routine repeats.
RETURN  LD      GR0, GR2        ; GR0 = Count
        POP     GR2             ;
        POP     GR1             ;
        RET                     ; Return to the calling program
        END                     ;
```

ANNEX 2 Spreadsheet Software Functions and Terminology (for IT Passport Examination)

The following defines the basic functions and technical terms of spreadsheet software.
In addition, when worksheet functions not described here, such as save, read, print, line creation, and graph creation, are used, they are described in the exam question text.

1. Worksheets

(1) The work area that is composed of a grid of rows and columns is called a worksheet. A worksheet has 256 columns and 10,000 rows.

(2) The position of each column and row in a worksheet is represented by a column letter and a row number. The leftmost column letter is A, and all column letters are represented as A, B, ..., Z, AA, AB, ..., AZ, BA, BB, ..., BZ, ..., IU, and IV. The uppermost row number is 1, and all row numbers are represented as 1, 2, …, 10000.

(3) Multiple worksheets can be used. In this case, a unique name is assigned to each worksheet in order to distinguish worksheets from one another.

2. Cells and cell ranges

(1) Each box-shaped area in the grid that constitutes a worksheet is called a cell. The position of a cell is represented by a pair of the column letter and row number of the cell, and this is called a cell address.

Example: The cell address at column A and row 1 is A1.

(2) Where a group of all cells included in certain rectangular area of a worksheet is handled, the cell addresses of the top left and bottom right cells of the group are used with a semicolon " : ", and the group is represented as "top left cell address:bottom right cell address". This is called a cell range.

Example: The cell range with the top left cell at A1 and the bottom right cell at B3 is represented as A1:B3 .

(3) When a cell address or a cell range in another worksheet is specified, the worksheet name and an exclamation mark "!" are used, and they are represented as "worksheet name!cell address" or "worksheet name!cell range" respectively.

Example: When the cell range B5:G10 in the worksheet "sheet1," is specified from another worksheet, it is represented as sheet1!B5:G10.

3. Values and expressions

(1) A cell holds a value, and this value can be referenced with the cell address. A value can be a numeric value, a character string, a logical value, or a null value.

(2) A character string is enclosed in a pair of single quotation marks ' '.

Example: The character strings "A" and "BC" are represented as 'A' and 'BC' respectively.

(3) A logical value is represented as true or false.

(4) A null value is represented as null, and a cell that has a null value is called a blank cell. The initial state of a cell is null.

(5) An expression can be entered into a cell. A cell holds the resulting value of an evaluated expression.

(6) An expression is comprised of constants, cell addresses, operators, parentheses, and functions. A constant is a notation that represents a numeric value, a character string, a logical value, or a null value. A cell address in an expression refers to the value held in the cell at that cell address.

(7) An expression can be an arithmetic expression, a character expression, or a logical expression. An expression whose evaluation generates a numeric value is called an arithmetic expression, an expression whose evaluation generates a character value is called a character expression, and an expression whose evaluation generates a logical value is called a logical expression.

(8) When an expression is entered into a cell, it is evaluated immediately. When the value in a cell that is referenced by an expression is changed, the expression is immediately re-evaluated.

4. Operators

(1) Unary operators are the plus sign "+" and the minus sign "−".

(2) Arithmetic operators are addition "+", subtraction "−", multiplication "*", division "/", and exponentiation "^".

(3) Comparison operators are greater than ">", less than "<", greater than or equal to "≥", less than or equal to "≤", equal to "=", and not equal to "≠".

(4) For grouping symbols, a pair of parentheses "( )" are used.

(5) When there are multiple operations and parentheses in an expression, the order of calculation is prioritized as shown in the table below.

Type of operation	Operator	Priority
Parentheses	( )	High
Exponentiation	^	↕
Unary operator	+, −	
Multiplication and division	*, /	
Addition and subtraction	+, −	
Comparison operators	>, <, ≥, ≤, =, ≠	Low

5. Copying of cells

(1) A value or expression in a cell can be copied into other cells.

(2) In a cell copy operation, when the source cell holds an expression containing a cell address, the cell reference method that changes the cell address in the expression so that the difference in the cell addresses of the source cell and destination cell can be maintained is called a relative reference. In this case, the expression that is entered into the destination cell is the expression modified by

adding the difference in column letter and row number between the source cell and destination cell to each cell address in the expression that is entered into the source cell.

Example: When the expression A1 + 5 in cell A6 is copied into cell B8, the expression B3+5 is entered into cell B8.

(3) In a cell copy operation, when the source cell holds an expression containing a cell address, the cell reference method that does not change either of or both the column letter and row number of the cell address is called an absolute reference. A dollar symbol "$" is placed immediately before either of or both the column letter and row number to which an absolute reference is applied.

Example: When the expression A1+$A2+A$5 in cell B1 is copied into cell C4, the expression A1+$A5+B$5 is entered into cell C4.

(4) In a cell copy operation, when the source cell holds an expression that refers to another worksheet, the name of the referenced worksheet does not change at the copy destination.

Example: When the expression Sheet1!A1 in cell A6 of the worksheet "Sheet2" is copied into cell B8 of the worksheet "Sheet3", the expression Sheet1!B3 is entered into cell B8 of "Sheet3."

6. Functions

The functions defined in the table below are available for use in an expression.

Format	Description
SUM(cell range[1)])	Returns the sum of the values in a specified cell range. Example: SUM(A1:B5) returns the sum of the values in the cell range A1:B5.
AVG(cell range[1)])	Returns the average of the values in a specified cell range.
SAMPLE_STD_DEV (cell range[1)])	Sample standard deviation: Returns the standard deviation of the sample values in a cell range.
POPULATION_STD_ DEV(cell range[1)])	Population standard deviation: Returns the standard deviation of the population values in a specified cell range.
MAX(cell range[1)])	Returns the maximum value in a specified cell range.
MIN(cell range[1)])	Returns the minimum value in a specified cell range.
IF(logical expression, expression 1, expression 2)	Returns the value of expression 1 when the value of the logical expression is true, and returns the value of expression 2 when it is false. Example: IF(B3 > A4, 'Chicago', C4) returns the character string "Chicago", if the value in cell B3 is greater than the value in cell A4. Otherwise, the value in cell C4 is returned.
COUNT(cell range)	Returns the number of non-blank cells in a specified cell range.
COUNTIF(cell range, search condition)	Returns the number of cells within a specified cell range that meet the specified search condition. The search condition is described using a combination of a comparison operator and an expression, and the value of each cell contained in a cell range is evaluated using the specified comparison operator. Example 1: COUNTIF(H5:L9, > A1) returns the number of cells in the cell range H5:L9 that hold a greater value than the value in cell A1 . Example 2: COUNTIF(H5:L9, = 'ABC') returns the number of cells in the cell range H5:L9 that hold the character string "ABC".

INT(arithmetic expression)	Integer: Rounds a value down to the maximum integer that is less than or equal to the value of a specified arithmetic expression. Example 1: INTEGER(3.9) returns 3. Example 2: INTEGER(−3.9) returns −4.
MOD(arithmetic expression 1, arithmetic expression 2)	Modulo arithmetic: Returns the remainder after the value of arithmetic expression 1 (dividend) is divided by the value of arithmetic expression 2 (divisor) . The equation below holds for functions MOD and INT. $MOD(x, y) = x - y * INT(x / y)$ Example 1: MOD(10, 3) returns 1. Example 2: MOD(−10, 3) returns 2.
SQRT(arithmetic expression)	Square root: Returns the non-negative square root of the value of a specified arithmetic expression. The value of the arithmetic expression must be a non-negative numeric value.
AND(logical expression 1, logical expression 2, ...)[2)]	Returns true only when the values of logical expression 1, logical expression 2, and all other logical expressions specified are true. Otherwise, it returns false.
OR(logical expression 1, logical expression 2, ...)[2)]	Returns true when the value of at least one expression out of logical expression 1, logical expression 2, and all other logical expressions specified is true. Otherwise, it returns false.
NOT(logical expression)	Returns false when the value of a specified logical expression is true, and returns true when the value of the logical expression is false.
ROUNDUP(arithmetic expression, position) ROUNDOFF(arithmetic expression, position) ROUNDDOWN (arithmetic expression, position)	Returns the value of a specified arithmetic expression that is rounded to the position specified. The function ROUNDUP rounds the value up, ROUNDOFF rounds the value off, and ROUNDDOWN rounds the value down. Here, the position is the number of places as counted from left to right with the first position being 0. Example 1: ROUNDUP(−314.159, 2) returns −314.16. Example 2: ROUNDUP(314.159, −2) returns 400. Example 3: ROUNDUP(314.159, 0) returns 315.
CONCATENATE (expression 1, expression 2, ...) [2)]	Treats the value of expression 1, expression 2, and all other expressions as a character string, and returns these as a single character string joined in order of the arguments. Example: CONCATENATE('Chicago', 'Washington', 123, 456) returns the character string "ChicagoWashington123456".
RANK(arithmetic expression, cell range[1)], order option)	Returns the rank order of the value of a specified arithmetic expression in a cell range in ascending order when the order option is 0, and in descending order when the order option is 1. Here, if the same value occurs more than once in the cell range, these values are given the same rank order and the subsequent rank order is set as this rank order plus the number of cells in the same rank order.
RANDOM( )	Returns a uniform random number (real value) that is greater than or equal to 0 and less than 1.
TLOOKUP(cell range, row position, column position)	Table lookup: Counts each row and column from the top left position of a specified cell range as 1, 2, ..., and returns the value of a cell at the location specified using a row position and column position. Example: TLOOKUP(A3:H11, 2, 5) returns the value of cell E4.

VLOOKUP (expression, cell range, column position, search option)	Vertical lookup: Scans the leftmost column of a specified cell range vertically from top to bottom, and searches for the first row with a cell that meets the search option. For the row found, counts each column from the leftmost column of the cell range as 1, 2, ..., and returns the value of the cell in the column specified by the column position. • When the search option is 0: searches for the value that matches the value of the expression. • When the search option is 1: searches for the maximum value that is less than or equal to the value of the expression. Here, the leftmost column must be sorted in ascending order from top to bottom in advance. Example: VLOOKUP(15, A2:E10, 5, 0) searches the leftmost column in the cell range from cell A2 through A10. Under this condition, if the value 15 is first found in cell A6, the value in cell E6 is returned, which is located at the 5th column E in the same row as cell A6.
HLOOKUP (expression, cell range, row position, search option)	Horizontal lookup: Scans the top row of a cell range horizontally from left to right, and searches for the first column with a cell that meets the search option. For the column found, counts each row from the top row of the cell range as1, 2, ..., and returns the value of the cell in the row specified by the row position. • When the search option is 0: searches for the value that matches the value of the expression value. • When the search option is 1: searches for the maximum value that is less than or equal to the value of the expression. Here, the top row must be sorted in ascending order from the left to right in advance. Example: HLOOKUP(15, A2:G6, 5, 1) searches the top row in the cell range from cell A2 to G2. Under this condition, if the maximum value less than or equal to 15 is first found in cell D2, the value in cell D6 is returned, which is located at the 5th row 6 in the same column as cell D2,.

Notes: [1] Values other than numeric values contained in the cell range that is passed as an argument are not processed.

[2] One or more expressions can be passed as a list of arguments.

ANNEX 3 Spreadsheet Software Functions and Terminology (for Fundamental Information Technology Engineers Examination)

The following defines the basic functions and technical terms of spreadsheet software.
In addition, when worksheet functions not described here, such as save, read, print, line creation, and graph creation, are used, they are described in the exam question text.

1. Worksheets

(1) The work area that is composed of a grid of rows and columns is called a worksheet. A worksheet has 256 columns and 10,000 rows.

(2) The position of each column and row in a worksheet is represented by a column letter and a row number. The leftmost column letter is A, and all column letters are represented as A, B, ..., Z, AA, AB, ..., AZ, BA, BB, ..., BZ, ..., IU, and IV. The uppermost row number is 1, and all row numbers are represented as 1, 2, …, 10000.

(3) Multiple worksheets can be used. In this case, a unique name is assigned to each worksheet in order to distinguish worksheets from one another.

2. Cells and cell ranges

(1) Each box-shaped area in the grid that constitutes a worksheet is called a cell. The position of a cell is represented by a pair of the column letter and row number of the cell, and this is called a cell address.

Example: The cell address at column A and row 1 is A1.

(2) Where a group of all cells included in certain rectangular area of a worksheet is handled, the cell addresses of the top left and bottom right cells of the group are used with a semicolon " : ", and the group is represented as "top left cell address:bottom right cell address". This is called a cell range.

Example: The cell range with the top left cell at A1 and the bottom right cell at B3 is represented as A1:B3 .

(3) When a cell address or a cell range in another worksheet is specified, the worksheet name and an exclamation mark "!" are used, and they are represented as "worksheet name!cell address" or "worksheet name!cell range" respectively.

Example: When the cell range B5:G10 in the worksheet "sheet1," is specified from another worksheet, it is represented as sheet1!B5:G10.

3. Values and expressions

(1) A cell holds a value, and this value can be referenced with the cell address. A value can be a numeric value, a character string, a logical value, or a null value.

(2) A character string is enclosed in a pair of single quotation marks ' '.

Example: The character strings "A" and "BC" are represented as 'A' and 'BC' respectively.

(3) A logical value is represented as true or false.

(4) A null value is represented as null, and a cell that has a null value is called a blank cell. The initial state of a cell is null.

(5) An expression can be entered into a cell. A cell holds the resulting value of an evaluated expression.

(6) An expression is comprised of constants, cell addresses, operators, parentheses, and functions. A constant is a notation that represents a numeric value, a character string, a logical value, or a null value. A cell address in an expression refers to the value held in the cell at that cell address.

(7) An expression can be an arithmetic expression, a character expression, or a logical expression. An expression whose evaluation generates a numeric value is called an arithmetic expression, an expression whose evaluation generates a character value is called a character expression, and an expression whose evaluation generates a logical value is called a logical expression.

(8) When an expression is entered into a cell, it is evaluated immediately. When the value in a cell that is referenced by an expression is changed, the expression is immediately re-evaluated.

4. Operators

(1) Unary operators are the plus sign "+" and the minus sign "−".

(2) Arithmetic operators are addition "+", subtraction "−", multiplication "*", division "/", and exponentiation "^".

(3) Comparison operators are greater than ">", less than "<", greater than or equal to "≥", less than or equal to "≤", equal to "=", and not equal to "≠".

(4) For grouping symbols, a pair of parentheses "( )" are used.

(5) When there are multiple operations and parentheses in an expression, the order of calculation is prioritized as shown in the table below.

Type of operation	Operator	Priority
Parentheses	( )	High
Exponentiation	^	↕
Unary operator	+, −	
Multiplication and division	*, /	
Addition and subtraction	+, −	
Comparison operators	>, <, ≥, ≤, =, ≠	Low

5. Copying of cells

(1) A value or expression in a cell can be copied into other cells.

(2) In a cell copy operation, when the source cell holds an expression containing a cell address, the cell reference method that changes the cell address in the expression so that the difference in the cell addresses of the source cell and destination cell can be maintained is called a relative reference. In this case, the expression that is entered into the destination cell is the expression modified by

adding the difference in column letter and row number between the source cell and destination cell to each cell address in the expression that is entered into the source cell.

Example: When the expression A1 + 5 in cell A6 is copied into cell B8, the expression B3+5 is entered into cell B8.

(3) In a cell copy operation, when the source cell holds an expression containing a cell address, the cell reference method that does not change either of or both the column letter and row number of the cell address is called an absolute reference. A dollar symbol "$" is placed immediately before either of or both the column letter and row number to which an absolute reference is applied.

Example: When the expression A1+$A2+A$5 in cell B1 is copied into cell C4, the expression A1+$A5+B$5 is entered into cell C4.

(4) In a cell copy operation, when the source cell holds an expression that refers to another worksheet, the name of the referenced worksheet does not change at the copy destination.

Example: When the expression Sheet1!A1 in cell A6 of the worksheet "Sheet2" is copied into cell B8 of the worksheet "Sheet3", the expression Sheet1!B3 is entered into cell B8 of "Sheet3."

6. Functions

The functions defined in the table below are available for use in an expression.

Format	Description
SUM(cell range[1)])	Returns the sum of the values in a specified cell range. Example: SUM(A1:B5) returns the sum of the values in the cell range A1:B5.
AVG(cell range[1)])	Returns the average of the values in a specified cell range.
SAMPLE_STD_DEV (cell range[1)])	Sample standard deviation: Returns the standard deviation of the sample values in a cell range.
POPULATION_STD_ DEV(cell range[1)])	Population standard deviation: Returns the standard deviation of the population values in a specified cell range.
MAX(cell range[1)])	Returns the maximum value in a specified cell range.
MIN(cell range[1)])	Returns the minimum value in a specified cell range.
IF(logical expression, expression 1, expression 2)	Returns the value of expression 1 when the value of the logical expression is true, and returns the value of expression 2 when it is false. Example: IF(B3 > A4, 'Chicago', C4) returns the character string "Chicago", if the value in cell B3 is greater than the value in cell A4. Otherwise, the value in cell C4 is returned.
COUNT(cell range)	Returns the number of non-blank cells in a specified cell range.
COUNTIF(cell range, search condition)	Returns the number of cells within a specified cell range that meet the specified search condition. The search condition is described using a combination of a comparison operator and an expression, and the value of each cell contained in a cell range is evaluated using the specified comparison operator. Example 1: COUNTIF(H5:L9, > A1) returns the number of cells in the cell range H5:L9 that hold a greater value than the value in cell A1 . Example 2: COUNTIF(H5:L9, = 'ABC') returns the number of cells in the cell range H5:L9 that hold the character string "ABC".

INT(arithmetic expression)	Integer: Rounds a value down to the maximum integer that is less than or equal to the value of a specified arithmetic expression. Example 1: INTEGER(3.9) returns 3. Example 2: INTEGER(−3.9) returns −4.
MOD(arithmetic expression 1, arithmetic expression 2)	Modulo arithmetic: Returns the remainder after the value of arithmetic expression 1 (dividend) is divided by the value of arithmetic expression 2 (divisor) . The equation below holds for functions MOD and INT. $MOD(x, y) = x - y * INT(x / y)$ Example 1: MOD(10, 3) returns 1. Example 2: MOD(−10, 3) returns 2.
SQRT(arithmetic expression)	Square root: Returns the non-negative square root of the value of a specified arithmetic expression. The value of the arithmetic expression must be a non-negative numeric value.
AND(logical expression 1, logical expression 2, ...)[2)]	Returns true only when the values of logical expression 1, logical expression 2, and all other logical expressions specified are true. Otherwise, it returns false.
OR(logical expression 1, logical expression 2, ...)[2)]	Returns true when the value of at least one expression out of logical expression 1, logical expression 2, and all other logical expressions specified is true. Otherwise, it returns false.
NOT(logical expression)	Returns false when the value of a specified logical expression is true, and returns true when the value of the logical expression is false.
ROUNDUP(arithmetic expression, position) ROUNDOFF(arithmetic expression, position) ROUNDDOWN (arithmetic expression, position)	Returns the value of a specified arithmetic expression that is rounded to the position specified. The function ROUNDUP rounds the value up, ROUNDOFF rounds the value off, and ROUNDDOWN rounds the value down. Here, the position is the number of places as counted from left to right with the first position being 0. Example 1: ROUNDUP(−314.159, 2) returns −314.16. Example 2: ROUNDUP(314.159, −2) returns 400. Example 3: ROUNDUP(314.159, 0) returns 315.
CONCATENATE (expression 1, expression 2, ...) [2)]	Treats the value of expression 1, expression 2, and all other expressions as a character string, and returns these as a single character string joined in order of the arguments. Example: CONCATENATE('Chicago', 'Washington', 123, 456) returns the character string "ChicagoWashington123456".
RANK(arithmetic expression, cell range[1)], order option)	Returns the rank order of the value of a specified arithmetic expression in a cell range in ascending order when the order option is 0, and in descending order when the order option is 1. Here, if the same value occurs more than once in the cell range, these values are given the same rank order and the subsequent rank order is set as this rank order plus the number of cells in the same rank order.
RANDOM( )	Returns a uniform random number (real value) that is greater than or equal to 0 and less than 1.
TLOOKUP(cell range, row position, column position)	Table lookup: Counts each row and column from the top left position of a specified cell range as 1, 2, ..., and returns the value of a cell at the location specified using a row position and column position. Example: TLOOKUP(A3:H11, 2, 5) returns the value of cell E4.

VLOOKUP (expression, cell range, column position, search option)	Vertical lookup: Scans the leftmost column of a specified cell range vertically from top to bottom, and searches for the first row with a cell that meets the search option. For the row found, counts each column from the leftmost column of the cell range as 1, 2, ..., and returns the value of the cell in the column specified by the column position. • When the search option is 0: searches for the value that matches the value of the expression. • When the search option is 1: searches for the maximum value that is less than or equal to the value of the expression. Here, the leftmost column must be sorted in ascending order from top to bottom in advance. Example: VLOOKUP(15, A2:E10, 5, 0) searches the leftmost column in the cell range from cell A2 through A10. Under this condition, if the value 15 is first found in cell A6, the value in cell E6 is returned, which is located at the 5th column E in the same row as cell A6.
HLOOKUP (expression, cell range, row position, search option)	Horizontal lookup: Scans the top row of a cell range horizontally from left to right, and searches for the first column with a cell that meets the search option. For the column found, counts each row from the top row of the cell range as1, 2, ..., and returns the value of the cell in the row specified by the row position. • When the search option is 0: searches for the value that matches the value of the expression value. • When the search option is 1: searches for the maximum value that is less than or equal to the value of the expression. Here, the top row must be sorted in ascending order from the left to right in advance. Example: HLOOKUP(15, A2:G6, 5, 1) searches the top row in the cell range from cell A2 to G2. Under this condition, if the maximum value less than or equal to 15 is first found in cell D2, the value in cell D6 is returned, which is located at the 5th row 6 in the same column as cell D2,.
RLOOKUP(expression, search cell range, extraction cell range)	Referential lookup: Scans a specified search cell range from the leftmost or uppermost cell for the search cell range and extraction cell range of the same size that cover one row or one column, and searches for the first cell that matches the value of the expression specified. Returns the value of the corresponding cell in the extraction cell range, which is located at the same relative position as the cell found in the search cell range. Example: RLOOKUP(15, A1:A8, C6:C13) searches the cells in the cell range A1:A8 sequentially from A1. Under this condition, if 15 is first found in cell A5, the value in cell C10 is returned, which is the 5th cell from the uppermost cell of the cell range C6:C13.

MLOOKUP(expression, cell range, search option)	Matching lookup: Scans a specified search cell range that covers one row or one column from the leftmost or uppermost cell, and searches for the first cell that matches the value of the expression specified. Returns the position of the cell found, which is counted from the leftmost or uppermost cell of the cell range with the first cell being 1. • When the search option is 0: searches for the value that matches the value of the expression. • When the search option is 1: searches for the maximum value that is less than or equal to the value of the expression. Here, the cell range must be sequentially sorted in ascending order from the leftmost or uppermost cell in advance. • Condition when the search option is −1: searches for the minimum value that is greater than or equal to the value of the expression. Here, the cell range must be sequentially sorted in descending order from the leftmost or uppermost cell in advance. Example: MLOOKUP(15, B2:B12, −1) searches cells sequentially from cell B2 in the cell range B2:B12. Under this condition, if the minimum value greater than or equal to 15 is first found in cell B9, the number of cells from cell B2 is counted and the value 8 is returned.
SUMIF(search cell range, search condition, sum cell range[1)])	Performs search and summation for a specified search cell range and a sum cell range which both have the same number of rows and the same number of columns. Searches for all cells in the search cell range that meet the condition specified in the search condition. Returns the total value of the corresponding cells in the sum cell range, which are located at the same relative positions as the cells found in the search cell range. The search condition is described using a combination of a comparison operator and an expression, and the values of the expression and each cell in the search cell range are evaluated using the specified comparison operator. Example 1: SUMIF(A1:B8, > E1, C2:D9) searches for all cells in the search cell range A1:B8 that have a value greater than the value of cell E1. Under this condition, if cells A2, B4, and B7 are found, the sum of the values of cells C3, D5, and D8 is calculated and returned, which are located at the same positions from the top left in the sum cell range C2:D9. Example 2: SUMIF(A1:B8, = 160, C2:D9) searches for all cells in the search cell range A1:B8 that have a value that is equal to 160 . Under this condition, if cells A2, B4, and B7 are found, the sum of the values of cells C3, D5, and D8 is calculated and returned, which are located at the same positions from the top left in the sum cell range C2:D9.

Notes: 1) Values other than numeric values contained in the cell range that is passed as an argument are not processed.

2) One or more expressions can be passed as a list of arguments.

7. Macros

(1) Worksheets and macros

Multiple macros can be stored in a worksheet.

Macros are declared with a unique macro name. Macros are executed using the execution function for macros in spreadsheet software.

Example: ○Macro:Pro

The example is a declaration of the macro `Pro`.

(2) Variables and cell variables

There are three types of variables: numeric type, character string type, and logical type. Each type can be used by declaration. A cell address cannot be used for a variable name.

Example: ○Numeric type: row, col

The example is a declaration of numeric type variables row and col.

The cell can be used as a variable, and this is called a cell variable. A cell variable can be used without declaring it. Expressions for cell variables can be an absolute expression or a relative expression.

An absolute expression for a cell variable uses a cell address.

A relative expression for a cell variable uses the format below.

Format	Description
RELATIVE(cell variable, row position, column position)	The cell specified with the cell variable is used as a base cell. The row position and column position of that cell are set to 0, and cells below and to the right are counted using positive numbers, and this variable represents the cell at the position that matches the numbers specified using the row position and the column position.

Examples: 1. RELATIVE(B5, 2, 3) is a variable that represents cell E7.

2. RELATIVE(B5, −2, −1) is a variable that represents cell A3.

(3) Arrays

A numeric array, a character string array, or a logical array can be used by declaration. The index is enclosed in a pair of brackets "[" and "]", and multiple indices are separated by commas. Indices start at 0.

In addition, if a variable and an array are numeric type or character string type, a null value can be stored in those elements.

Example: ○Character string type: table[100, 200]

This example is a declaration of a two dimensional array called table that contains 100×200 character string elements.

(4) Declaration, comments, and processes

The description of declaration, comments, and processes follows the "Common Description Format for Pseudo-Languages."

When an expression or a function is used in the description of a process, variables, cell variables, and elements of an array can be used in the description.

Example: ○ Numeric type: row

row: 0, row < 5, 1

• RELATIVE(B5, row, 0) ← RANKING(RELATIVE(C5, row, 0), G5:G9, 0)

In this example, for each value in cells C5 through C9, the rank order in the cell range G5:G9 is checked, and that each rank order is substituted into cells B5 through B9 in sequence.

May 22, 2012

■Outline of Information Technology Engineers Examination■

Information-technology Promption Agency, Japan

Address
Green Court Center Office 15F
2-28-8 Hon-Komagome, Bunkyo, Tokyo 113-6591 Japan