Questions

printf

scanf

Precedence table

lvalue

rvalue

1. Write in the resulting output of printf, let i = 5, j = 23.461

Example:

printf	(	“	%	d	a	b		c	|	“	,	i	)	;											
output:	5	a	b		c	|																				

a.

printf	(	“	|	2	%	8	.	1	f	,	%	3	.	4	d	|	“	,	j	,	i	)	;			
output:																										

b.

printf	(	“	|	5	%	8	f	,	%	3	d	}	“	,	j	,	i	)	;							
output:																										

c.

printf	(	“	|	%	2	d	]	5	6	7	|	“	,	1	2	3	4	)	;							
output:																										

d.

printf	(	“	|	%	4	.	8	f	|	“	,	1	2	.	3	4	5	)	;							
output:																										

2a. Write down all the stages of compiling a program

2b. I have a file name “haato.c”, write down the command to generate an assembly file named “haachama”

2c. I have a file name “lulu.c”, write down the command to generate an object file named “suzuhara”

2d. I have a file name “hahaha.c”, I need a program named “peko”

3. Let i = 1, j = 2, k = 3.33

Write the NEWLY output of i and j, if applicable, after scanf is called together with INPUT values

a.

scanf	(	“		%	d	,	|	a	b	“		,	&	j	)	;										
input:		2	0	,	|	a	b																			

Current value of i	
Current value of j	

b.

scanf	(	“	%	d	%	d	“	,	&	i	,	&	j	)	;											
input:	3	0	0				4	4																		

Current value of i	
Current value of j	

c.

scanf	(	“	%	d	%	d	“	,	&	i	,	&	j	)	;											
input:	3	0	0	,			4	4																		

Current value of i	
Current value of j	

d.

scanf	(	“	%	d	_	%	d	“	,	&	i	,	&	j	)	;										
input:			3				4	4																		

Current value of i	
Current value of j	

e.

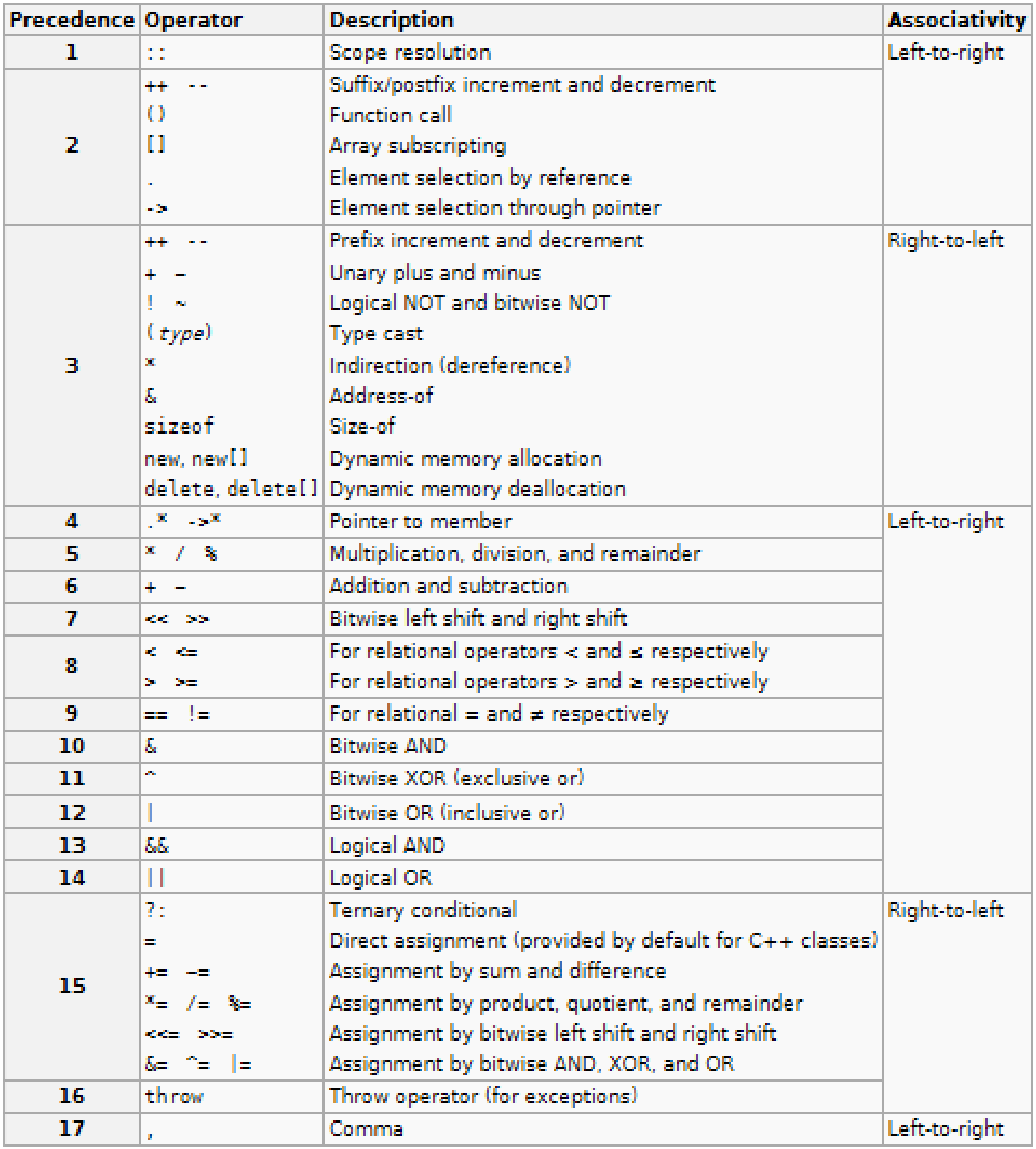
scanf	(	“		_	%	d	4	4	%	d	“	,	&	i	,	&	j	)	;							
input:		_	3			4	4	4																		

Current value of i	
Current value of j	

f.

scanf	(	“	*	_	%	d	4	4	%	d	“	,	&	i	,	&	j	)	;							
input:		*	_	6		4	4	4																		

Current value of i	
Current value of j	



Precedence	Operator	Description	Associativity
1	`::`	Scope resolution	Left-to-right

2	`++ --`
	`()`
	`[]`
	`.`
	`->`
		Suffix/postfix increment and decrement
		Function call
		Array subscripting
		Element selection by reference
		Element selection through pointer	

3	`++ --`
	`+ -`
	`! ~`
	`(` *type* `)`
	`*`
	`&`
	`sizeof`
	`new, new[]`
	`delete, delete[]`
		Prefix increment and decrement
		Unary plus and minus
		Logical NOT and bitwise NOT
		Type cast
		Indirection (dereference)
		Address-of
		Size-of
		Dynamic memory allocation
		Dynamic memory deallocation
			Right-to-left

4	`.* ->*`	Pointer to member	Left-to-right
5	`* / %`	Multiplication, division, and remainder	
6	`+ -`	Addition and subtraction	
7	`<< >>`	Bitwise left shift and right shift	

8	`< <=`
	`> >=`
		For relational operators < and ≤ respectively
		For relational operators > and ≥ respectively	

9	`== !=`	For relational = and ≠ respectively	
10	`&`	Bitwise AND	
11	`^`	Bitwise XOR (exclusive or)	
12	`|`	Bitwise OR (inclusive or)	
13	`&&`	Logical AND	
14	`||`	Logical OR	

15	`?:`
	`=`
	`+= -=`
	`*= /= %=`
	`<<= >>=`
	`&= ^= |=`
		Ternary conditional
		Direct assignment (provided by default for C++ classes)
		Assignment by sum and difference
		Assignment by product, quotient, and remainder
		Assignment by bitwise left shift and right shift
		Assignment by bitwise AND, XOR, and OR
			Right-to-left

16	`throw`	Throw operator (for exceptions)	
17	`,`	Comma	Left-to-right

4. Insert Parentheses to represent the precedence and associativity of the C expression:

Example:

a. a / b + c * d - f

b. a * b % c / d - - f

c. a + b ++ + c -- - d

d. a -- + b * ++ c / -- d

e. a - - b + + c

5. Evaluate the expression (s) if possible, if not, write ERROR

Let i = 5, j = 20, **EACH QUESTION IS INDEPENDENT** from each other!

++i - 4	
i + + - - -j	
i = j ++ = ++ i	
i = j + 5 + i	
i = j += 2 * ++j	
j = 1 - 2 + 3 * 2 + i++ = 2	
i / 5 * j	
j -= j / i -- * i	
++ i --	
+ + j ++ + - - i --	
i++i--	
i+++j	
i+ + + j	
i + ++ j	

Answers

1. **Write in the resulting output of printf, let i = 5, j = 23.461**

Example:

printf	(	“	%	d	a	b		c	|	“	,	i	)	;												
output:	5	a	b		c	|																				

a.

printf	(	“	|	2	%	8	.	1	f	,	%	3	.	4	d	|	“	,	j	,	i	)	;			
output:	|	2					2	3	.	4	,	0	0	0	5	|										

b.

printf	(	“	|	5	%	8	f	,	%	3	d	}	“	,	j	,	i	)	;							
output:	|	5			2	3	.	4	6	1	,			5	}											

c.

printf	(	“	|	%	2	d	]	5	6	7	|	“	,	1	2	3	4	)	;							
output:	|	1	2	3	4	]	5	6	7	|																

d.

printf	(	“	|	%	4	.	8	f	|	“	,	1	2	.	3	4	5	)	;							
output:	|	1	2	.	3	4	5	0	0	0	0	0	|													

2a. Write down all the stages of compiling a program

Pre-processing, Compiling, Assembling, Linking

2b. I have a file name “haato.c”, write down the command to generate an assembly file named “haachama”

```
gcc -S haato.c -o haachama.s
```

2c. I have a file name “lulu.c”, write down the command to generate an object file named “suzuhara”

```
gcc -c lulu.c -o suzuhara.o
```

2d. I have a file name “hahaha.c”, I need a program named “peko”

```
gcc hahaha.c -o peko.exe
```

3. Let i = 1, j = 2, k = 3.33

Write the NEWLY output of i and j, if applicable, after scanf is called together with INPUT values

a.

scanf	(	“		%	d	,	|	a	b	“		,	&	j	)	;										
input:		2	0	,	|	a	b																			

Current value of i	1
Current value of j	20

b.

scanf	(	“	%	d	%	d	“	,	&	i	,	&	j	)	;											
input:	3	0	0				4	4																		

Current value of i	300
Current value of j	44

c.

scanf	(	“	%	d	%	d	“	,	&	i	,	&	j	)	;											
input:	3	0	0	,			4	4																		

Current value of i	300
Current value of j	2

d.

scanf	(	“	%	d	_	%	d	“	,	&	i	,	&	j	)	;										
input:			3				4	4																		

Current value of i	3
Current value of j	2

e.

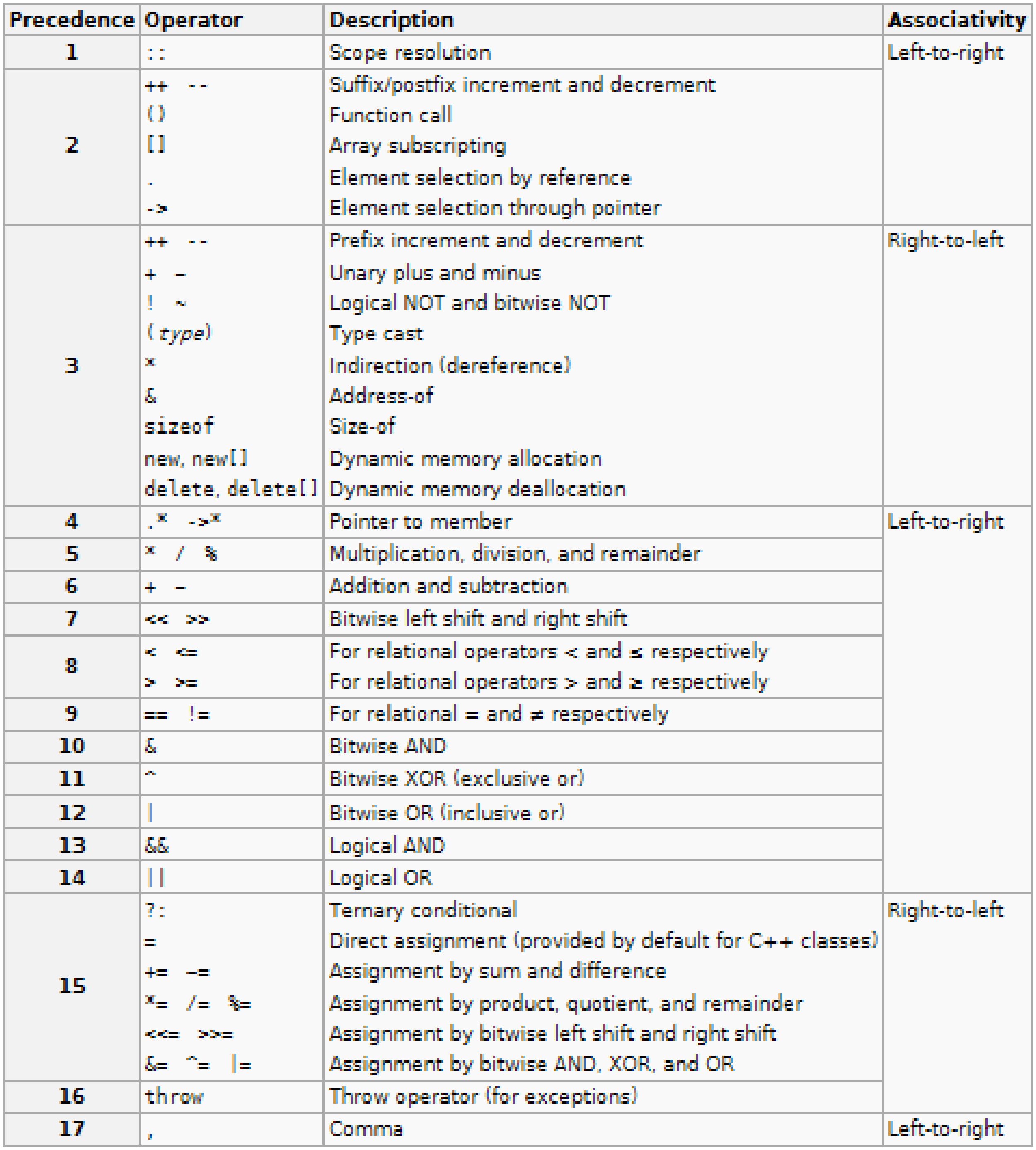
scanf	(	“		_	%	d	4	4	%	d	“	,	&	i	,	&	j	)	;							
input:		_	3			4	4	4																		

Current value of i	3
Current value of j	4

f.

scanf	(	“	*	_	%	d	4	4	%	d	“	,	&	i	,	&	j	)	;							
input:		*	_	6		4	4	4																		

Current value of i	1
Current value of j	2



Precedence	Operator	Description	Associativity
1	::	Scope resolution	Left-to-right
2	++ --	Suffix/postfix increment and decrement	
	()	Function call	
	[]	Array subscripting	
	.	Element selection by reference	
	->	Element selection through pointer	
3	++ --	Prefix increment and decrement	Right-to-left
	+ −	Unary plus and minus	
	! ~	Logical NOT and bitwise NOT	
	(*type*)	Type cast	
	*	Indirection (dereference)	
	&	Address-of	
	sizeof	Size-of	
	new, new[]	Dynamic memory allocation	
	delete, delete[]	Dynamic memory deallocation	
4	.* ->*	Pointer to member	Left-to-right
5	* / %	Multiplication, division, and remainder	
6	+ −	Addition and subtraction	
7	<< >>	Bitwise left shift and right shift	
8	< <=	For relational operators < and ≤ respectively	
	> >=	For relational operators > and ≥ respectively	
9	== !=	For relational = and ≠ respectively	
10	&	Bitwise AND	
11	^	Bitwise XOR (exclusive or)	
12	|	Bitwise OR (inclusive or)	
13	&&	Logical AND	
14	||	Logical OR	
15	?:	Ternary conditional	Right-to-left
	=	Direct assignment (provided by default for C++ classes)	
	+= −=	Assignment by sum and difference	
	*= /= %=	Assignment by product, quotient, and remainder	
	<<= >>=	Assignment by bitwise left shift and right shift	
	&= ^= |=	Assignment by bitwise AND, XOR, and OR	
16	throw	Throw operator (for exceptions)	
17	,	Comma	Left-to-right

4. Insert Parentheses to represent the precedence and associativity of the C expression:

Example:

a. a / b + c * d - f

```
( ( ( a / b ) + (c * d) ) - f )
```

b. a * b % c / d - - f

```
( ( ( ( ( a * b ) % ) c )  / d ) - ( - f ) )
```

c. a + b ++ + c -- - d

```
(a + ( b ++ ) + ( c -- ) - d
```

d. a -- + b * ++ c / -- d

```
( ( a -- ) + ( ( b * ( ++ c ) )  / ( -- d ) ) )
```

e. a - - b + + c

```
( ( a - ( - b ) ) + ( + c ) )
```

5. Evaluate the expression (s) if possible, if not, write ERROR

Let i = 5, j = 20, **EACH QUESTION IS INDEPENDENT** from each other!

++i - 4	2
i + + - - -j	-15
i = j ++ = ++ i	ERROR
i = j + 5 + i	30
i = j += 2 * ++j	63
j = 1 - 2 + 3 * 2 + i++ = 2	ERROR
i / 5 * j	20
j -= j / i -- * i	4
++ i --	ERROR
+ + j ++ + - - i --	25
i++i--	ERROR
i+++j	25
i+ + + j	25
i + ++ j	26