```
1 #include <stdio.h>
3 int main(void) {
     int year = 0;
 5
     scanf("%d", &year);
 6
7
     int leap = 0;
8
9
    // TODO: C operator precedence
    // URL: https://en.cppreference.com/w/c/language/
10
   operator_precedence
11
12
    // TODO (hfwei): order of evaluation
13
14
    // TODO: short-circuit evaluation
15
    // test: year = 25
   // test: year = 80
16
    // test: year = 100
17
    // test: year = 400
18
19
    // TODO: ! (year % 100 == 0)
20
    if ((year % 4 == 0 && year % 100 != 0) || (year % 400
    == 0)) {
21
      leap = 1;
22
     } else {
23
      leap = 0;
     }
24
25
26
    // int leap = (year % 4 == 0 && year % 100 != 0) || (
   year % 400 == 0);
27
     // TODO: leap = 0
28
29
     if (leap == 0) {
30
       printf("%d is a common year\n", year);
31
     } else {
32
       printf("%d is a leap year\n", year);
     }
33
34
35
     return 0;
36 }
```

```
1 # `2-if-for-array`
2
3
4 ## `min-of-two.c`
 5
6 - `if-else`
7 - code style
8 - tab vs. space video
    - google format
10
    - format on save
11 - `?:`: conditional operator; ternary operator
12 - `fmin, fmax` for doubles
13
14 ## `min-of-three.c`
15
16 - nested `if-else`
17 - `if-else` template
18 - comment for `else`
19
20 ## `leap-if-else.c`
21
22 - flowchart
23 - `leap`: 0/1 integer as a flag
24 - `if-else`
25 - easier cases go first
26 - code style
27 - spaces
28 - `==`: 0 == leap
29 - `if (leap == 0)` vs `if (leap != 0)`
30
31 ## `leap-else-if.c`
32
33 - easier cases go first (Flatten Arrow Code)
34
35 ## `leap-elseif.c`
36
37 - `else if` (Cascading If Statements)
38 - `{ }` removed
39 - `if` and `else` in the same line
    - `Code => Format Code (Ctrl + Alt + L)`
41 - find the iff condition for leap
42
43 ## `leap.c`
44
```

```
45 - `&&`, `||` operator
     - operator precedence (<a href="https://en.cppreference.com/w/c/">https://en.cppreference.com/w/c/</a>
46
   language/operator_precedence)
47 - short-circuit
     - test: 25, 80, 100, 400
48
     - TODO: order of evaluation (<a href="https://en.cppreference.com">https://en.cppreference.com</a>
49
   /w/c/language/eval_order)
     - `i = ++i + i++;`
50
51 - Code improvements
52 - `if`: without `else`
53 - `int leap = (year % 4 == 0 && year % 100 != 0) || (year
    % 400 == 0);
54
    - `?:` in `printf`
55
56 ## `min-array.c`
57
58 - `array`
59
     - `array initializer` (被網
60
     - What if uninitialized? (garbage in, garbage out)
61
     - designator (Since C99)
     - `int n[5] = {[4]=5,[0]=1,2,3,4}; // holds 1,2,3,4,5`
62
63
       int a[MAX] = { // starts initializing a[0] = 1, a[1
   ] = 3, \dots
65
         1, 3, 5, 7, 9, [MAX-5] = 8, 6, 4, 2, 0
66
67
68 - `const int NUM`
69 - `#define NUM 5`
70 - `for`
71
     - syntax
72
       - `for (init-clause; condition-expression; iteration-
   expression) loop-statement`
73
     - semantics (CLion debug!!!)
74
       - (1): []
75
       - (2): i < NUM: not i <= NUM (accessing out-of-bounds;
    训奶!
76
       - (3): int i = 1; since C99 (declaration in for-loop);
    code in standard C library
77
78 ## `min-array-input.c`
79
80 - `array` initializer
81
     - designator
```

```
82
    - What if uninitialized?
83 - input an array
84 - `&numbers[i]`: lvalue
85 - what if `n (NUM)` is unknown???
86
87 ## Additional
88
89 - `Settings` => `Code Style` (Google)
90 - `Settings` => `Action on Save` (Formatting Code)
91 - TODO (hfwei): CLion code template
```

```
1 #include <stdio.h>
2
3 #define NUM 5
4
5 int main(void) {
   // TODO (hfwei): Variable length array (VLA) folded to
   constant array as an extension
7
    // const int NUM = 5;
8
9
    // index starting from 0
    // 5 elements (no '\0')
10
    // int[] numbers = {23, 56, 19, 11, 78};
11
12
    // variable-sized object may not be initialized
    // designator: from C99
13
14
     int numbers[NUM] = {23, 56, 19, 11, 78};
15
16
    // []: array subscripting operator
17
     int min = numbers[0];
18
19
    // syntax + semantics
    // syntax: for (init-clause; condition-expression;
20
  iteration-expression) loop-statement
21
    // semantics: debug!!!
22
    // (1): []
    // (2): i < NUM: not i <= NUM (accessing out-of-bounds;</pre>
23
   THE P
24
    // (3): int i = 1; since C99 (declaration in for-loop);
   code in standard C library
25
     for (int i = 1;
26
                     i < NUM;
27
                              i++) {
28
       if (numbers[i] < min) {</pre>
29
         min = numbers[i];
30
       }
31
     }
32
33
     printf("min = %d\n", min);
34
35
     return 0;
36 }
```

```
1 #include <stdio.h>
2
3 int main(void) {
4 int a = 0;
5
     int b = 0;
6
     scanf("%d%d", &a, &b);
7
8
9
     // TODO: calculate the minimum of a and b
    // code style: space, {, newline, tab vs. spaces
10
     // do not ignore { } for single-line statements
11
12
    // google format, format on save
13
     // ?:
    int min = 0;
14
     if (a >= b) {
15
16
     min = b;
17
     } else {
18
       min = a;
19
     }
20
21
     // conditional operator, ternary operator
     // int min = \alpha >= b ? b : \alpha;
22
23
     printf("min(%d, %d) = %d\n", a, b, min);
24
25
26
     return 0;
27 }
```

```
1 #include <stdio.h>
2
3 int main(void) {
    int year = 0;
5
     scanf("%d", &year);
6
7
     int leap = 0;
8
9
    // TODO (hfwei): repeated branch body in conditional
   chain
     if (year % 4 != 0) {
10
11
      leap = 0;
12
     } else if (year % 100 != 0) {
13
      leap = 1; // year % 4 == 0 and year % 100 != 0
     } else if (year % 400 != 0) {
14
15
      leap = 0;
16
     } else {
17
      leap = 1; // (year % 4 == 0 and year % 100 == 0 and)
   year % 400 == 0
18
    }
19
     if (leap == 0) {
20
21
       printf("%d is a common year\n", year);
22
     } else {
23
       printf("%d is a leap year\n", year);
     }
24
25
26
     return 0;
27 }
```

```
1 add_executable(min-of-two min-of-two.c)
2 add_executable(min-of-three min-of-three.c)
3 add_executable(min-array min-array.c)
4 add_executable(min-array-input min-array-input.c)
5
6 add_executable(leap-if-else leap-if-else.c)
7 add_executable(leap-else-if leap-else-if.c)
8 add_executable(leap-elseif leap-elseif.c)
9 add_executable(leap leap.c)
```

```
1 #include <stdio.h>
2
3 int main(void) {
     int year = 0;
 5
     scanf("%d", &year);
 6
 7
     int leap = 0;
8
9
     if (year % 4 != 0) {
10
       leap = 0;
     } else {
11
12
       if (year % 100 != 0) {
         leap = 1;
13
14
       } else {
         if (year % 400 != 0) {
15
16
           leap = 0;
17
         } else {
18
           leap = 1;
19
         }
20
       }
21
     }
22
23
     if (leap == 0) {
       printf("%d is a common year\n", year);
24
25
     } else {
       printf("%d is a leap year\n", year);
26
27
     }
28
29
     return 0;
30 }
```

```
1 #include <stdio.h>
2
3 int main(void) {
     int year = 0;
5
     scanf("%d", &year);
 6
7
     // TODO: leap year or not
8
    // boolean variable
9
     int leap = 0;
    // TODO (hfwei): arrow code
10
     if (year % 4 == 0) {
11
       if (year % 100 == 0) {
12
13
         if (year % 400 == 0) {
14
           leap = 1;
         } else { // can be removed
15
16
           leap = 0;
17
         }
18
       } else {
19
         leap = 1;
20
21
     } else { // can be removed; // easier case goes first
22
      leap = 0;
23
     }
24
25
     if (leap == 0) {
26
       printf("%d is a common year\n", year);
27
     } else {
28
       printf("%d is a leap year\n", year);
29
     }
30
31
     return 0;
32 }
```

```
1 #include <stdio.h>
3 int main(void) {
    int a = 0;
5
    int b = 0;
6
    int c = 0;
7
    scanf("%d%d%d", &a, &b, &c);
8
9
10
    // TODO: calculate the minimum of a, b and c
11
     int min = 0;
12
13
     if (a > b) {
       if (b > c) {
14
15
         min = c;
16
       } else { // b <= c and b < a
17
         min = b;
       }
18
19
     } else { // α <= b
20
       if (a < c) {
21
         min = a;
       } else { // c <= α <= b
22
23
         min = c;
24
       }
25
     }
26
     printf("min(%d, %d, %d) = %d\n", a, b, c, min);
27
28
29
     return 0;
30 }
```

```
1 #include <stdio.h>
2
3 #define NUM 5
4
5 int main(void) {
     int numbers[NUM] = { 0 };
7
8
     // int i = 0;
    // &: address-of operator (numbers[i] is an lvalue)
9
     for (int i = 0; i < NUM; i++) {
10
       scanf("%d", &numbers[i]);
11
     }
12
13
14
     // []: array subscripting operator
     int min = numbers[0];
15
16
17
     // syntax + semantics
    // syntax: for (init-clause; condition-expression;
18
   iteration-expression) loop-statement
19
    // semantics: debug!!!
     // (1): []
20
    // (2): i < NUM: not i <= NUM (accessing out-of-bounds;</pre>
21
   THE P
22
     // (3): int i = 1; since C99 (declaration in for-loop);
  code in standard C library
     for (int i = 1;
23
24
          i < NUM;
25
          i++) {
       if (numbers[i] < min) {</pre>
26
27
         min = numbers[i];
28
       }
29
     }
30
31
     printf("min = %d\n", min);
32
33
     return 0;
34 }
```