```c
#include <stdio.h>

int main(void) {
  int year = 0;
  scanf("%d", &year);

  int leap = 0;

  // TODO: C operator precedence
  // URL: https://en.cppreference.com/w/c/language/operator_precedence

  // TODO (hfwei): order of evaluation

  // TODO: short-circuit evaluation
  // test: year = 25
  // test: year = 80
  // test: year = 100
  // test: year = 400
  // TODO: ! (year % 100 == 0)
  if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0)) {
    leap = 1;
  } else {
    leap = 0;
  }

  // int leap = (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);

  // TODO: leap = 0
  if (leap == 0) {
    printf("%d is a common year\n", year);
  } else {
    printf("%d is a leap year\n", year);
  }

  return 0;
}
```

```
 1 # `2-if-for-array`
 2
 3
 4 ## `min-of-two.c`
 5
 6 - `if-else`
 7 - code style
 8   - tab vs. space video
 9   - 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
40   - `Code => Format Code (Ctrl + Alt + L)`
41 - find the iff condition for leap
42
43 ## `leap.c`
44
```

```
45 - `&&`, `||` operator
46   - operator precedence (https://en.cppreference.com/w/c/
   language/operator_precedence)
47 - short-circuit
48   - test: 25, 80, 100, 400
49   - TODO: order of evaluation (https://en.cppreference.com
   /w/c/language/eval_order)
50     - `i = ++i + i++;`
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` (初始化)
60   - What if uninitialized? (garbage in, garbage out)
61   - designator (Since C99)
62   - `int n[5] = {[4]=5,[0]=1,2,3,4}; // holds 1,2,3,4,5`
63   ```C
64     int a[MAX] = { // starts initializing a[0] = 1, a[1
   ] = 3, ...
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;
    访问越界)
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
```

```
  - What if uninitialized?
- input an array
  - `&numbers[i]`: lvalue
  - what if `n (NUM)` is unknown???

## Additional

- `Settings` => `Code Style` (Google)
- `Settings` => `Action on Save` (Formatting Code)
- TODO (hfwei): CLion code template
```

```c
#include <stdio.h>

#define NUM 5

int main(void) {
  // TODO (hfwei): Variable length array (VLA) folded to constant array as an extension
  // const int NUM = 5;

  // index starting from 0
  // 5 elements (no '\0')
  // int[] numbers = {23, 56, 19, 11, 78};
  // variable-sized object may not be initialized
  // designator: from C99
  int numbers[NUM] = {23, 56, 19, 11, 78};

  // []: array subscripting operator
  int min = numbers[0];

  // syntax + semantics
  // syntax: for (init-clause; condition-expression; iteration-expression) loop-statement
  // semantics: debug!!!
  // (1): []
  // (2): i < NUM: not i <= NUM (accessing out-of-bounds; 访问越界)
  // (3): int i = 1; since C99 (declaration in for-loop); code in standard C library
  for (int i = 1;
                  i < NUM;
                          i++) {
    if (numbers[i] < min) {
      min = numbers[i];
    }
  }

  printf("min = %d\n", min);

  return 0;
}
```

```c
#include <stdio.h>

int main(void) {
  int a = 0;
  int b = 0;

  scanf("%d%d", &a, &b);

  // TODO: calculate the minimum of a and b
  // code style: space, {, newline, tab vs. spaces
  // do not ignore { } for single-line statements
  // google format, format on save
  // ?:
  int min = 0;
  if (a >= b) {
    min = b;
  } else {
    min = a;
  }

  // conditional operator, ternary operator
  // int min = a >= b ? b : a;

  printf("min(%d, %d) = %d\n", a, b, min);

  return 0;
}
```

```c
#include <stdio.h>

int main(void) {
  int year = 0;
  scanf("%d", &year);

  int leap = 0;

  // TODO (hfwei): repeated branch body in conditional chain
  if (year % 4 != 0) {
    leap = 0;
  } else if (year % 100 != 0) {
    leap = 1; // year % 4 == 0 and year % 100 != 0
  } else if (year % 400 != 0) {
    leap = 0;
  } else {
    leap = 1; // (year % 4 == 0 and year % 100 == 0 and) year % 400 == 0
  }

  if (leap == 0) {
    printf("%d is a common year\n", year);
  } else {
    printf("%d is a leap year\n", year);
  }

  return 0;
}
```

```
add_executable(min-of-two min-of-two.c)
add_executable(min-of-three min-of-three.c)
add_executable(min-array min-array.c)
add_executable(min-array-input min-array-input.c)

add_executable(leap-if-else leap-if-else.c)
add_executable(leap-else-if leap-else-if.c)
add_executable(leap-elseif leap-elseif.c)
add_executable(leap leap.c)
```

```c
#include <stdio.h>

int main(void) {
  int year = 0;
  scanf("%d", &year);

  int leap = 0;

  if (year % 4 != 0) {
    leap = 0;
  } else {
    if (year % 100 != 0) {
      leap = 1;
    } else {
      if (year % 400 != 0) {
        leap = 0;
      } else {
        leap = 1;
      }
    }
  }

  if (leap == 0) {
    printf("%d is a common year\n", year);
  } else {
    printf("%d is a leap year\n", year);
  }

  return 0;
}
```

```c
#include <stdio.h>

int main(void) {
  int year = 0;
  scanf("%d", &year);

  // TODO: leap year or not
  // boolean variable
  int leap = 0;
  // TODO (hfwei): arrow code
  if (year % 4 == 0) {
    if (year % 100 == 0) {
      if (year % 400 == 0) {
        leap = 1;
      } else {  // can be removed
        leap = 0;
      }
    } else {
      leap = 1;
    }
  } else {  // can be removed; // easier case goes first
    leap = 0;
  }

  if (leap == 0) {
    printf("%d is a common year\n", year);
  } else {
    printf("%d is a leap year\n", year);
  }

  return 0;
}
```

```c
#include <stdio.h>

int main(void) {
  int a = 0;
  int b = 0;
  int c = 0;

  scanf("%d%d%d", &a, &b, &c);

  // TODO: calculate the minimum of a, b and c
  int min = 0;

  if (a > b) {
    if (b > c) {
      min = c;
    } else {  // b <= c and b < a
      min = b;
    }
  } else {  // a <= b
    if (a < c) {
      min = a;
    } else {  // c <= a <= b
      min = c;
    }
  }

  printf("min(%d, %d, %d) = %d\n", a, b, c, min);

  return 0;
}
```

```c
#include <stdio.h>

#define NUM 5

int main(void) {
  int numbers[NUM] = { 0 };

  // int i = 0;
  // &: address-of operator (numbers[i] is an lvalue)
  for (int i = 0; i < NUM; i++) {
    scanf("%d", &numbers[i]);
  }

  // []: array subscripting operator
  int min = numbers[0];

  // syntax + semantics
  // syntax: for (init-clause; condition-expression; iteration-expression) loop-statement
  // semantics: debug!!!
  // (1): []
  // (2): i < NUM: not i <= NUM (accessing out-of-bounds; 访问越界)
  // (3): int i = 1; since C99 (declaration in for-loop); code in standard C library
  for (int i = 1;
       i < NUM;
       i++) {
    if (numbers[i] < min) {
      min = numbers[i];
    }
  }

  printf("min = %d\n", min);

  return 0;
}
```