

```
1 // Created by hfwei on 2024/10/6.
2
3 #include <stdio.h>
4
5 int main(void) {
6     int year = 0;
7     scanf("%d", &year);
8
9     int leap = 0;
10
11     // TODO: leap year or not (logical expressions)
12     if ((year % 4 == 0 && year % 100 != 0) ||
13         year % 400 == 0) {
14         leap = 1;
15     }
16
17     if (leap == 0) {
18         printf("%d is a common year\n", year);
19     } else {
20         printf("%d is a leap year\n", year);
21     }
22
23     return 0;
24 }
```

```
1 // Created by hfwei on 2024/10/16.
2
3 #include <stdio.h>
4
5 #define LEN_L 5
6 #define LEN_R 6
7
8 int L[LEN_L] = { 1, 3, 5, 7, 9 };
9 int R[LEN_R] = { 0, 2, 4, 6, 8, 10 };
10
11 int main(void) {
12     // TODO: merge L and R into a sorted array
13     int l = 0;
14     int r = 0;
15
16     while (l < LEN_L && r < LEN_R) {
17         if (L[l] < R[r]) {
18             printf("%d ", L[l]);
19             l++;
20         } else {
21             printf("%d ", R[r]);
22             r++;
23         }
24     }
25
26     while (r < LEN_R) {
27         printf("%d ", R[r]);
28         r++;
29     }
30
31     while (l < LEN_L) {
32         printf("%d ", L[l]);
33         l++;
34     }
35
36     return 0;
37 }
```

```
1 // Created by hfwei on 2024/10/10.
2
3 #include <stdio.h>
4
5 int main(void) {
6     int lines = 0;
7     scanf("%d", &lines);
8
9     // TODO: print stars pyramid
10    for (int i = 0; i < lines; i++) {
11        // print n - 1 - i spaces
12        for (int j = 0; j < lines - 1 - i; ++j) {
13            printf(" ");
14        }
15
16        // print 2 * i + stars
17        for (int j = 0; j < 2 * i + 1; ++j) {
18            printf("*");
19        }
20
21        if (i < lines - 1) {
22            printf("\n");
23        }
24    }
25
26    return 0;
27 }
```

```
1 // Created by hfwei on 2024/10/10.
2
3 #include <stdio.h>
4 #include <stdbool.h>
5
6 int main(void) {
7     int max = 0;
8     scanf("%d", &max);
9
10    int count = 0;
11
12    // TODO: print primes between 1 and max
13    for (int i = 2; i <= max; i++) {
14        bool is_prime = true;
15
16        for (int j = 2; j * j <= i; j++) {
17            // if j is a factor of i
18            if (i % j == 0) {
19                is_prime = false;
20                break;
21            }
22        }
23
24        if (is_prime) {
25            count++;
26        }
27    }
28
29    printf("\n %d ", count);
30
31    return 0;
32 }
```

```
1 // Created by hfwei on 2024/10/10.
2
3 #include <stdio.h>
4 #include <string.h>
5 #include <stdbool.h>
6
7 #define LEN 21
8 char string[LEN] = "";
9
10 int main() {
11     // example: nolemon,nomelon
12     printf("Input a string containing at most 20 characters.
13     \n");
14     scanf("%20s", string);
15
16     int len = strlen(string);
17     printf("The length of \"%s\" is %d.\n", string, len);
18
19     // TODO: test palindrome
20     bool is_palindrome = true;
21
22     for (int i = 0, j = len - 1; i < j; i++, j--) {
23         if (string[i] != string[j]) {
24             is_palindrome = false;
25             break;
26         }
27     }
28
29     printf("\"%s\" is %s a palindrome.\n", string,
30         is_palindrome ? "" : "not");
31
32     return 0;
33 }
```

```
1 # 2-if-for-array
2 add_executable(leap-func leap.c)
3
4 # 3-for-a-while
5 add_executable(primes-func primes.c)
6 add_executable(stars-func stars.c)
7 add_executable(binary-search-func binary-search.c)
8 add_executable(palindrome-func palindrome.c)
9 add_executable(selection-sort-func selection-sort.c)
10
11 # 4-loops
12 add_executable(insertion-sort-func insertion-sort.c)
13 add_executable(binary-insertion-sort-func binary-insertion
    -sort.c)
14
15 add_executable(merge-func merge.c)
16 add_executable(game-of-life-func game-of-life.c)
17 add_executable(game-of-life-transformed game-of-life-
    transformed.c)
```

```
1 // Created by hfwei on 2024/10/16.
2
3 #include <stdio.h>
4 #include <unistd.h>
5 #include <stdlib.h>
6
7 #define SIZE 6
8
9 const int board[SIZE][SIZE] = {
10     { 0 },
11     { 0, 1, 1, 0, 0, 0 },
12     { 0, 1, 1, 0, 0, 0 },
13     { 0, 0, 0, 1, 1, 0 },
14     { 0, 0, 0, 1, 1, 0 },
15     { 0 }
16 };
17
18 //const int board1[SIZE][SIZE] = {
19 //    [0] = { 0 },
20 //    [1] = { [1] = 1, [2] = 1 },
21 //    [2] = { [1] = 1, [2] = 1 },
22 //    [3] = { [3] = 1, [4] = 1 },
23 //    [4] = { [3] = 1, [4] = 1 },
24 //};
25
26 //const int board2[SIZE][SIZE] = {
27 //    [1][1] = 1, [1][2] = 1,
28 //    [2][1] = 1, [2][2] = 1,
29 //    [3][3] = 1, [3][4] = 1,
30 //    [4][3] = 1, [4][4] = 1
31 //};
32
33 //const int board3[SIZE][SIZE] = {
34 //    0, 0, 0, 0, 0, 0,
35 //    0, 1, 1, 0, 0, 0,
36 //    0, 1, 1, 0, 0, 0,
37 //    0, 0, 0, 1, 1, 0,
38 //    0, 0, 0, 1, 1, 0,
39 //};
40
41 int main(void) {
42     // TODO: play game-of-life
43
44     // expand this board
```

```
45  int old_board[SIZE + 2][SIZE + 2] = {0};
46  for (int i = 1; i <= SIZE; i++) {
47      for (int j = 1; j <= SIZE; j++) {
48          old_board[i][j] = board[i - 1][j - 1];
49      }
50  }
51
52  // print the old_board
53  for (int i = 1; i <= SIZE; i++) {
54      for (int j = 1; j <= SIZE; j++) {
55          printf("%c ", old_board[i][j] ? '*' : ' ');
56      }
57      printf("\n");
58  }
59  system("clear");
60
61  // count live neighbors for each cell
62  int new_board[SIZE + 2][SIZE + 2] = {0};
63
64  for (int i = 0; i < 10; ++i) {
65      for (int i = 1; i <= SIZE; i++) {
66          for (int j = 1; j <= SIZE; j++) {
67              int num_of_live_neighbors =
68                  old_board[i - 1][j - 1] +
69                  old_board[i - 1][j] +
70                  old_board[i - 1][j + 1] +
71                  old_board[i][j - 1] +
72                  old_board[i][j + 1] +
73                  old_board[i + 1][j - 1] +
74                  old_board[i + 1][j] +
75                  old_board[i + 1][j + 1];
76
77              if (old_board[i][j] == 1) {
78                  new_board[i][j] = (num_of_live_neighbors == 2
|| num_of_live_neighbors == 3);
79              } else {
80                  new_board[i][j] = num_of_live_neighbors == 3;
81              }
82          }
83      }
84
85      // print the new board
86      for (int i = 1; i <= SIZE; i++) {
87          for (int j = 1; j <= SIZE; j++) {
```



```
88         printf("%c ", new_board[i][j] ? '*' : ' ');
89     }
90     printf("\n");
91 }
92
93     // #include <unistd.h> (Linux, macOS)
94     // <windows.h> Sleep(1000)
95     sleep(1);
96
97     // <stdlib.h> (Linux)
98     // Windows: stdlib.h (system("cls"))
99     system("clear");
100
101     // copy new board onto old board
102     for (int i = 1; i <= SIZE; i++) {
103         for (int j = 1; j <= SIZE; j++) {
104             old_board[i][j] = new_board[i][j];
105         }
106     }
107 }
108
109 return 0;
110 }
```

```
1 // Created by hfwei on 2024/10/10.
2
3 #include <stdio.h>
4
5 #define LEN 10
6 int dictionary[LEN] = { 1, 1, 2, 3, 5, 8, 13, 21, 34, 55 };
7
8 int main(void) {
9     int key = 0;
10    scanf("%d", &key);
11
12    int low = 0;
13    int high = LEN - 1;
14
15    int index = -1;
16    // TODO: binary search: search for key in dictionary[]
17    while (low <= high) {
18        int mid = (low + high) / 2;
19        if (dictionary[mid] == key) {
20            index = mid;
21            break;
22        } else if (dictionary[mid] > key) {
23            high = mid - 1;
24        } else {
25            low = mid + 1;
26        }
27    }
28
29    if (index == -1) {
30        printf("Not found!\n");
31    } else {
32        printf("The index of %d is %d.\n", key, index);
33    }
34
35    return 0;
36 }
```

```
1 // Created by hfwei on 2024/10/16.
2 // Code generated by ChatGPT.
3
4 #include <stdio.h>
5 #include <stdlib.h>
6 #include <time.h>
7
8 #define MAX_LEN 10000
9 #define RANGE 10
10
11 int main(void) {
12     int numbers[MAX_LEN] = { 0 };
13
14     int size = 0;
15     scanf("%d", &size);
16
17     // generate an array of random integers between 0 and
18     RANGE - 1
19     srand(time(NULL));
20     for (int i = 0; i < size; i++) {
21         numbers[i] = rand() % RANGE;
22     }
23     // print the original array
24     for (int i = 0; i < size; i++) {
25         printf("%d ", numbers[i]);
26     }
27     printf("\n");
28
29     // TODO: insertion sort
30     for (int i = 1; i < size; i++) {
31         // numbers[0 .. i - 1] is already sorted
32         int key = numbers[i];
33
34         int j = i - 1;
35         while (j >= 0 && numbers[j] > key) {
36             numbers[j + 1] = numbers[j];
37             j--;
38         }
39         numbers[j + 1] = key;
40
41         // numbers[0 .. i] is already sorted
42         for (int i = 0; i < size; i++) {
43             printf("%d ", numbers[i]);
```

```
44     }
45     printf("\n");
46 }
47 // i = size
48 // numbers[0 .. size - 1] is already sorted
49
50 // print the sorted array
51 for (int i = 0; i < size; i++) {
52     printf("%d ", numbers[i]);
53 }
54 printf("\n");
55
56 return 0;
57 }
```

```
1 // Created by hfwei on 2024/10/10.
2
3 #include <stdio.h>
4
5 #define LEN 20
6 int numbers[LEN] = { 0 };
7
8 void SelectionSort(int arr[], int len);
9 void Print(const int arr[], int len);
10
11 int main(void) {
12     int len = -1;
13     while (scanf("%d", &numbers[++len]) != EOF);
14
15     Print(numbers, len);
16     SelectionSort(numbers, len);
17     Print(numbers, len);
18
19     return 0;
20 }
21
22 void SelectionSort(int arr[], int len) {
23     for (int i = 0; i < len; i++) {
24         // find the minimum value of numbers[i .. n-1]
25         int min = arr[i];
26         int min_index = i;
27
28         for (int j = i + 1; j <= len - 1; ++j) {
29             if (arr[j] < min) {
30                 min = arr[j];
31                 min_index = j;
32             }
33         }
34
35         // swap arr[i] and arr[min_index]
36         int temp = arr[i];
37         arr[i] = arr[min_index];
38         arr[min_index] = temp;
39     }
40 }
41
42 void Print(const int arr[], int len) {
43     printf("\n");
44     for (int i = 0; i < len; i++) {
```

```
45     printf("%d ", arr[i]);  
46 }  
47 printf("\n");  
48 }
```

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <time.h>
4
5 #define MAX_LEN 10000
6 #define RANGE 10
7
8 int main() {
9     int numbers[MAX_LEN] = { 0 };
10
11     int size = 0;
12     scanf("%d", &size);
13
14     srand(time(NULL));
15     for (int i = 0; i < size; i++) {
16         numbers[i] = rand() % RANGE;
17     }
18
19     // print the original array
20     for (int i = 0; i < size; i++) {
21         printf("%d ", numbers[i]);
22     }
23     printf("\n");
24
25     // TODO
26     for (int i = 1; i < size; i++) {
27         int key = numbers[i];
28         int low = 0;
29         int high = i - 1;
30
31         while (low <= high) {
32             int mid = (low + high) / 2;
33             if (key >= numbers[mid]) {
34                 low = mid + 1;
35             } else {
36                 high = mid - 1;
37             }
38         }
39
40         for (int j = i - 1; j >= low; j--) {
41             numbers[j + 1] = numbers[j];
42         }
43         numbers[low] = key;
44     }
```

```
45     for (int i = 0; i < size; i++) {
46         printf("%d ", numbers[i]);
47     }
48     printf("\n");
49 }
50
51 // Print the sorted array
52 for (int i = 0; i < size; i++) {
53     printf("%d ", numbers[i]);
54 }
55 printf("\n");
56
57 return 0;
58 }
```



```
1 //
2 // Created by hengxin on 10/19/22.
3 // Run it with "Terminal"
4 //
5
6 #include <stdio.h>
7 #include <stdlib.h>
8 #include <unistd.h>
9
10 #define SIZE 6
11
12 void ExtendBoard(const int origin_board[][SIZE],
13                 int extended_board[][SIZE + 2]);
14 void PrintExtendedBoard(const int extended_board[][SIZE +
15 2]);
16 void GenerateNewBoard(const int old_board[][SIZE + 2],
17                      int new_board[][SIZE + 2]);
18 void CopyExtendedBoard(const int src_board[][SIZE + 2],
19                       int dest_board[][SIZE + 2]);
20 void SleepAndClear(int sec);
21
22 int main() {
23     const int board[SIZE][SIZE] = {
24         { 0 },
25         { 0, 1, 1, 0, 0, 0 },
26         { 0, 1, 1, 0, 0, 0 },
27         { 0, 0, 0, 1, 1, 0 },
28         { 0, 0, 0, 1, 1, 0 },
29         { 0 }
30     };
31
32     int old_board[SIZE + 2][SIZE + 2] = { 0 };
33     ExtendBoard(board, old_board);
34     PrintExtendedBoard(old_board);
35     SleepAndClear(1);
36
37     int new_board[SIZE + 2][SIZE + 2] = { 0 };
38     for (int round = 0; round < 10; round++) {
39         GenerateNewBoard(old_board, new_board);
40         SleepAndClear(1);
41         PrintExtendedBoard(new_board);
42         CopyExtendedBoard(new_board, old_board);
43     }
```

```

44     return 0;
45 }
46
47 void ExtendBoard(const int origin_board[][SIZE],
48                 int extended_board[][SIZE + 2]) {
49     for (int row = 1; row <= SIZE; row++) {
50         for (int col = 1; col <= SIZE; col++) {
51             extended_board[row][col] = origin_board[row - 1][col
52             - 1];
53         }
54     }
55
56 void PrintExtendedBoard(const int extended_board[][SIZE +
57 2]) {
58     for (int row = 1; row <= SIZE; row++) {
59         for (int col = 1; col <= SIZE; col++) {
60             printf("%c ", extended_board[row][col] ? '*' : ' ');
61         }
62         printf("\n");
63     }
64
65 void GenerateNewBoard(const int old_board[][SIZE + 2],
66                      int new_board[][SIZE + 2]) {
67     for (int row = 1; row <= SIZE; row++) {
68         for (int col = 1; col <= SIZE; col++) {
69             // count the number of neighbours of old_board[row][
col]
70             int neighbours =
71                 old_board[row - 1][col - 1] +
72                 old_board[row - 1][col] +
73                 old_board[row - 1][col + 1] +
74                 old_board[row][col - 1] +
75                 old_board[row][col + 1] +
76                 old_board[row + 1][col - 1] +
77                 old_board[row + 1][col] +
78                 old_board[row + 1][col + 1];
79
80             // evaluate the new board
81             if (old_board[row][col]) { // old_board[row][col]
is alive
82                 new_board[row][col] = (neighbours == 2 ||

```

```
82 neighbours == 3);
83     } else { // old_board[row][col] is dead
84         new_board[row][col] = (neighbours == 3);
85     }
86 }
87 }
88 }
89
90 void CopyExtendedBoard(const int src_board[][SIZE + 2],
91                        int dest_board[][SIZE + 2]) {
92     for (int row = 1; row <= SIZE; row++) {
93         for (int col = 1; col <= SIZE; col++) {
94             dest_board[row][col] = src_board[row][col];
95         }
96     }
97 }
98
99 void SleepAndClear(int sec) {
100     sleep(sec);
101     system("clear");
102 }
```