```c
// Created by hfwei on 2022/10/13.
//

#include <stdio.h>

#define LEN 10
int dictionary[LEN] = {1, 1, 2, 3, 5, 8, 13, 21, 34, 55};

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

  int index = -1;

  int low = 0;
  int high = LEN - 1;

  while (low <= high) {
    // int mid = low + (high - low) / 2
    int mid = (low + high) / 2;
    printf("low = %d\t high = %d\t mid = %d\n", low, high, mid);

    if (key < dictionary[mid]) {
      high = mid - 1;
    } else if (key > dictionary[mid]) {
      low = mid + 1;
    } else {
      index = mid;
      break;
      // search for the leftmost one
      //  high = index - 1;
    }
  }

  if (index == -1) {
    printf("Not found!\n");
  } else {
    printf("The index of %d is %d.\n", key, index);
  }

  return 0;
}
```

```c
//
// Created by hengxin on 10/19/22.
//
// For the usage of clock(),
// please refer to https://legacy.cplusplus.com/reference/ctime/clock/.
//

#include <stdlib.h>
#include <time.h>
#include <stdio.h>
#include <stdbool.h>

#define LEN 1000000
int numbers[LEN] = {0};

int main() {
  srand(time(NULL));
  for (int i = 0; i < LEN; i++) {
    numbers[i] = rand() % LEN;
  }

  // set a clock
  clock_t start = clock();

  bool swapped = true;
  for (int i = 0; i < LEN && swapped; i++) {
    swapped = false;
    for (int j = 0; j < LEN - 1 - i; j++) {
      if (numbers[j] > numbers[j + 1]) {
        int temp = numbers[j];
        numbers[j] = numbers[j + 1];
        numbers[j + 1] = temp;
        swapped = true;
      }
    }
  }

  // record the end clock
  clock_t end = clock();

  for (int i = 0; i < LEN; i++) {
    printf("%d ", numbers[i]);
  }

  long sec = (end - start) / CLOCKS_PER_SEC;
  printf("Sorting %d numbers in %ld seconds.\n", LEN, sec);

  return 0;
}
```

```c
//
// Created by hengxin on 10/19/22.
// Run it with "Terminal"
//

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define SIZE 6
const int board[SIZE][SIZE] = {
    {0},
    {0, 1, 1, 0, 0, 0},
    {0, 1, 1, 0, 0, 0},
    {0, 0, 0, 1, 1, 0},
    {0, 0, 0, 1, 1, 0},
    {0}
};

//const int board[SIZE][SIZE] = {
//    [1][1] = 1, [1][2] = 1,
//    [2][1] = 1, [2][2] = 1,
//    [3][3] = 1, [3][4] = 1,
//    [4][3] = 1, [4][4] = 1
//};

int main() {
  int old_board[SIZE + 2][SIZE + 2] = {0};

  for (int row = 0; row < SIZE + 2; row++) {
    for (int col = 0; col < SIZE + 2; col++) {
      if (row == 0 || row == SIZE + 1 || col == 0 || col == SIZE + 1 ) {
        old_board[row][col] = 0;
      } else {
        old_board[row][col] = board[row - 1][col - 1];
      }
    }
  }

  // print the original board
  for (int row = 1; row <= SIZE; row++) {
    for (int col = 1; col <= SIZE; col++) {
      printf("%c ", old_board[row][col] ? '*' : ' ');
    }
    printf("\n");
  }

  // clear the screen
  system("clear");

  int new_board[SIZE + 2][SIZE + 2] = {0};

  for (int round = 1; round < 10; round++) {
    for (int row = 1; row <= SIZE; row++) {
      for (int col = 1; col <= SIZE; col++) {
        // count the number of neighbours of old_board[row][col]
        int neighbours =
            old_board[row - 1][col - 1] +
                old_board[row - 1][col] +
                old_board[row - 1][col + 1] +
                old_board[row][col - 1] +
                old_board[row][col + 1] +
                old_board[row + 1][col - 1] +
                old_board[row + 1][col] +
                old_board[row + 1][col + 1];

        // evaluate the new board
        if (old_board[row][col]) {  // old_board[row][col] is alive
          new_board[row][col] = (neighbours == 2 || neighbours == 3);
        } else {  // old_board[row][col] is dead
          new_board[row][col] = (neighbours == 3);
        }
      }
    }

    // print the new board
    for (int row = 1; row <= SIZE; row++) {
      for (int col = 1; col <= SIZE; col++) {
        printf("%c ", new_board[row][col] ? '*' : ' ');
      }
      printf("\n");
    }

    // sleep for a while
    // Linux: #include <unistd.h>
    // Windows: #include <windows.h>: Sleep(ms)
    sleep(1);

    // clear the screen
    // Windows: #include <conio.h> clrscr();
    system("clear");

    // start the next round
    for (int row = 1; row <= SIZE; row++) {
      for (int col = 1; col <= SIZE; col++) {
        old_board[row][col] = new_board[row][col];
      }
    }
  }

  return 0;
}
```

```c
//
// Created by hfwei on 2022/9/21.
//

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main() {
  int high = 100;
  int number_of_tries = 2;

  /**
   * (1) print the rules of the game to players
   */
  printf("==========\n"
         "Let us play the Guess the Number game.\n"
         "The computer will generate a random number between 1 and %d.\n"
         "You have totally %d tries.\n"
         "==========\n",
         high, number_of_tries);

  /**
   * (2) generate a random number between 1 and high as the secret number
   */
  srand(time(NULL));
  int secret = rand() % high + 1;
  printf("=====Just for Test: secret = %d.=====\n", secret);

  /**
   * (3) ask the player to input a guess
   */
  while (number_of_tries > 0) {
    printf("You still have %d tries.\n"
           "Please input your guess.\n", number_of_tries);

    /**
     * (4) obtain the guessed number,
     * compare it with the secret number,
     * and inform the player of the result
     */
    int guess;
    scanf("%d", &guess);

    if (guess == secret) {
      printf("Congs! You win! \n");
      break;
    } else if (guess > secret) {
      printf("guess > secret.\n");
    } else {
      printf("guess < secret.\n");
    }

    /**
     * (5) repeat (3)-(4) until the player wins or loses
     */
    number_of_tries--;

    if (number_of_tries == 0) {
      printf("Sorry. You Lose!");
    }
  }

  return 0;
}
```

```c
//
// Created by hengxin on 10/19/22.
//

#include <stdio.h>
#define LEN_L 5
#define LEN_R 6

int L[LEN_L] = {1, 3, 5, 7, 9};
int R[LEN_R] = {0, 2, 4, 6, 8, 10};

int main() {
  int l = 0;
  int r = 0;

  while (l < LEN_L && r < LEN_R) {
    if (L[l] <= R[r]) {
      printf("%d ", L[l]);
      l++;
    } else { // L[l] > R[r]
      printf("%d ", R[r]);
      r++;
    }
  }

  while (l < LEN_L) {
    printf("%d ", L[l]);
    l++;
  }

  while (r < LEN_R) {
    printf("%d ", R[r]);
    r++;
  }

  return 0;
}
```

File - D:\cpl\cpl-coding-0\2022-CPL\5-function\leap.c

```c
// Created by hfwei on 2022/10/6.
//

#include <stdio.h>
int main() {
  int year = 0;
  scanf("%d", &year);

  int leap = 0;

  // test: year = 25
  // test: year = 80
  // test: year = 100
  // test: year = 400
  if ((year % 4 == 0 && year % 100 != 0) ||
      (year % 400 == 0)) {
    leap = 1;
  }

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

  printf("The year %d is a %s year.\n", year, leap? "leap" : "common");

  return 0;
}

```

```c
// Created by hfwei on 2022/10/13.
//

#include <stdio.h>
#include <string.h>
#include <stdbool.h>

#define LEN 21
char string[LEN] = "";

int main() {
  // example: nolemon,nomelon
  scanf("%20s", string);

//  int len = 0;
//  while (string[len] != '\0') {
//    len++;
//  }
  int len = strlen(string);
  printf("The length of \"%s\" is %d.\n", string, len);

  bool is_palindrome = true;
  for (int i = 0, j = len - 1; i < j; i++, j--) {
    if (string[i] != string[j]) {
      is_palindrome = false;
      break;
    }
  }

  printf("\"%s\" is %s a palindrome.\n", string,
         is_palindrome ? "" : "not");

  return 0;
}
```

```c
// Created by hfwei on 2022/10/13.
//

#include <stdio.h>
#include <stdbool.h>
#include <math.h>
#include <time.h>

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

  int count = 0;
  clock_t start = clock();
  for (int number = 2; number <= max; number++) {
    // why 1? why not 0?
//    int is_prime = 1;
    // since C99
    bool is_prime = true;

    for (int i = 2; i * i <= number; i++) {
      if (number % i == 0) {
//        is_prime = 0;
        is_prime = false;
        break;
      }
    }

    if (is_prime) {
      count++;
      printf("%d ", number);
    }
  }
  clock_t end = clock();

  printf("\ncount = %d in %f seconds\n",
         count, (double) (end - start) / CLOCKS_PER_SEC);

  return 0;
}
```

```c
// Created by hfwei on 2022/10/13.
//

#include <stdio.h>

#define LEN 20
int numbers[LEN] = {0};

int main() {
  /*
   * Input the array
   * Note: fails to run this program in "Run" (Ctrl + D)
   * See: https://youtrack.jetbrains.com/issue/CPP-5704
   * Use "Terminal" instead.
   */
  int len = -1;
  while (scanf("%d", &numbers[++len]) != EOF);
//  while (scanf("%d", &numbers[++len]) == 1);

  for (int i = 0; i < len; i++) {
    printf("%d ", numbers[i]);
  }
  printf("\n");

  printf("--------------------\n");
  for (int i = 0; i < len; i++) {
    // find the minimum of numbers[i .. len - 1]
    int min = numbers[i];
    int min_index = i;
    for (int j = i + 1; j < len; j++) {
      if (numbers[j] < min) {
        min = numbers[j];
        min_index = j;
      }
    }

    // swap numbers[i] and numbers[min_index]
    int tmp = numbers[i];
    numbers[i] = numbers[min_index];
    numbers[min_index] = tmp;

    // print it out
    for (int i = 0; i < len; i++) {
      printf("%d ", numbers[i]);
    }
    printf("\n");
    printf("--------------------\n");
  }
}
```

```c
// file: stars.c
// author: hfwei
// date: 2022/10/13
// description: print stars pyramid

#include <stdio.h>
int main() {
  int lines;
  scanf("%d", &lines);

  for (int i = 0; i < lines; ++i) {
    // print [lines - (i + 1)] spaces
    for (int j = 0; j < lines - (i + 1); ++j) {
      printf(" ");
    }

    // print [2 * i + 1] spaces
    for (int j = 0; j < 2 * i + 1; ++j) {
      printf("*");
    }

    if (i < lines - 1) {
      printf("\n");
    }
  }

  return 0;
}
```

```
# 5-function

## `static.c`
    - `random.c`

## `generic.c`
```