```c
//
// Created by hengxin on 10/23/21.
//

#include <stdio.h>

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

/**
 * Binary search for the KEY in the array DICT of length LEN.
 *
 * @param dict The array in which key is to be searched.
 * @param len The length of the array.
 * @param key The key to search for.
 * @return The index of the key in the array; -1 if the key is not found.
 */
int BinarySearch(const int dict[], int len, int key);

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

  int index = BinarySearch(dictionary, LEN, key);

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

  return 0;
}

int BinarySearch(const int dict[], int len, int key) {
  int low = 0;
  int high = len - 1;
  int mid = 0;

  while (low <= high) {
    mid = (low + high) / 2;
    if (key < dict[mid]) {
      high = mid - 1;
    } else if (key > dict[mid]) {
      low = mid + 1;
    } else {
      return mid;
    }
  }

  return -1;
}
```

```c
/**
 * file: game-of-life.c
 *
 * Simulate "Conway's Game of Life"
 * See https://en.wikipedia.org/wiki/Conway%27s_Game_of_Life
 * Play with it: https://playgameoflife.com/
 *
 * Created by hengxin on 10/30/21.
 */

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

#define ROUND 10
#define SIZE 6
int board[SIZE][SIZE] = {
    {0},
    {0},
    {0, 0, 1, 1, 1, 0},
    {0, 1, 1, 1, 0, 0},
    {0},
    {0}};

void ExtendBoard(const int origin_board[][SIZE],
                 int extended_board[][SIZE + 2]);
void PrintExtendedBoard(const int extended_board[][SIZE + 2]);
void GenerateNewBoard(const int old_extended_board[][SIZE + 2],
                      int new_extended_board[][SIZE + 2]);
void CopyExtendedBoard(const int src_board[][SIZE + 2],
                       int dest_board[][SIZE + 2]);
void ClearTerminal(int sec);

int main() {
  int old_board[SIZE + 2][SIZE + 2];
  ExtendBoard(board, old_board);
  PrintExtendedBoard(old_board);
  ClearTerminal(1);

  int new_board[SIZE + 2][SIZE + 2];
  for (int round = 0; round < ROUND; round++) {
    GenerateNewBoard(old_board, new_board);
    PrintExtendedBoard(new_board);
    ClearTerminal(1);
    CopyExtendedBoard(new_board, old_board);
  }

  return 0;
}

void ExtendBoard(const int origin_board[][SIZE],
                 int extended_board[][SIZE + 2]) {
  for (int row = 0; row < SIZE + 2; row++) {
    for (int col = 0; col < SIZE + 2; col++) {
```

```
 54       if (row == 0 || row == SIZE + 1 || col == 0 || col == SIZE + 1
    ) {
 55         extended_board[row][col] = 0;
 56       } else {
 57         extended_board[row][col] = origin_board[row - 1][col - 1];
 58       }
 59     }
 60   }
 61 }
 62
 63 void PrintExtendedBoard(const int extended_board[][SIZE + 2]) {
 64   for (int row = 1; row < SIZE + 1; row++) {
 65     for (int col = 1; col < SIZE + 1; col++) {
 66       printf("%c ", extended_board[row][col] ? '*' : ' ');
 67     }
 68     printf("\n");
 69   }
 70 }
 71
 72 void GenerateNewBoard(const int old_extended_board[][SIZE + 2],
 73                       int new_extended_board[][SIZE + 2]) {
 74   for (int row = 1; row < SIZE + 1; row++) {
 75     for (int col = 1; col < SIZE + 1; col++) {
 76       int neighours = old_extended_board[row - 1][col - 1]
 77           + old_extended_board[row - 1][col]
 78           + old_extended_board[row - 1][col + 1]
 79           + old_extended_board[row][col - 1]
 80           + old_extended_board[row][col + 1]
 81           + old_extended_board[row + 1][col - 1]
 82           + old_extended_board[row + 1][col]
 83           + old_extended_board[row + 1][col + 1];
 84
 85       new_extended_board[row][col] =
 86           (old_extended_board[row][col] && (neighours == 2 ||
    neighours == 3))
 87               || (!old_extended_board[row][col] && neighours == 3);
 88     }
 89   }
 90 }
 91
 92 void CopyExtendedBoard(const int src_board[][SIZE + 2],
 93                        int dest_board[][SIZE + 2]) {
 94   for (int row = 0; row < SIZE + 2; row++) {
 95     for (int col = 0; col < SIZE + 2; col++) {
 96       dest_board[row][col] = src_board[row][col];
 97     }
 98   }
 99 }
100
101 void ClearTerminal(int sec) {
102   sleep(sec);
103   printf("\033c");
104 }
```

```c
//
// Created by hengxin on 10/16/21.
//

#include <stdio.h>

int IsLeapYear(int year);

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

  printf("The year %d is%s a leap year.\n",
         year,
         IsLeapYear(year) ? "" : " not");

  return 0;
}

int IsLeapYear(int year) {
  return ((year % 4 == 0 && year % 100 != 0) || year % 400 == 0);
}
```

```c
/**
 * Merge two sorted arrays into one
 *
 * Created by hengxin on 10/30/21.
 */

#include <stdio.h>
#include <math.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};

/**
 * Please fill in the comment.
 *
 * @param left
 * @param len_left
 * @param right
 * @param len_right
 */
void Merge(const int left[], int len_left, const int right[], int len_right);

int main() {
  Merge(L, LEN_L, R, LEN_R);
  return 0;
}

void Merge(const int left[], int len_left, const int right[], int len_right) {
  int l = 0;
  int r = 0;

  while (l < len_left && r < len_right) {
    if (left[l] <= right[r]) {
      printf("%d ", left[l]);
      l++;
    } else { // left[l] > right[r]
      printf("%d ", right[r]);
      r++;
    }
  }

  while (l < len_left) {
    printf("%d ", left[l]);
    l++;
  }
  while (r < len_right) {
    printf("%d ", right[r]);
    r++;
```

```
52    }
53 }
54
```

```c
//
// Created by hengxin on 10/23/21.
//

#include <stdio.h>

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

int Len(const char str[]);
int IsParlindrome(const char str[]);

int main() {
  scanf("%20s", string);

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

  return 0;
}

int Len(const char str[]) {
  int len = 0;
  while (str[len] != '\0') {
    len++;
  }
  return len;
}

int IsParlindrome(const char str[]) {
  for (int i = 0, j = Len(str) - 1; i < j; i++, j--) {
    if (str[i] != str[j]) {
      return 0;
    }
  }

  return 1;
}
```

```c
//
// Created by hengxin on 10/23/21.
//

#include <stdio.h>

int IsPrime(int number);

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

  for (int number = 2; number <= max; number++) {
    if (IsPrime(number)) {
      printf("%d ", number);
    }
  }

  return 0;
}

int IsPrime(int number) {
  for (int i = 2; i < number; i++) {
    if (number % i == 0) {
      return 0;
    }
  }

  return 1;
}

```

```c
//
// Created by hengxin on 10/16/21.
//

#include <stdio.h>

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

void Swap(int left, int right);
void Print(const int arr[], int len);

/**
 * Sort the array ARR of length LEN using the selection sort algorithm.
 *
 * @param arr The array to be sorted.
 * @param len The length of the array.
 */
void SelectionSort(int arr[], int len);

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);

  SelectionSort(numbers, len);
  Print(numbers, len);

  return 0;
}

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

void SelectionSort(int arr[], int len) {
  for (int i = 0; i < len; ++i) {
    int min = arr[i];
    int min_index = i;

    for (int j = i + 1; j < len; j++) {
      if (min > arr[j]) {
        min = arr[j];
```

```
53          min_index = j;
54        }
55      }
56
57      /**
58       * swap arr[i] and arr[min_index]
59       */
60      int tmp = arr[i];
61      arr[i] = arr[min_index];
62      arr[min_index] = tmp;
63  //    Swap(arr[i], arr[min_index]);
64    }
65  }
66
67  /**
68   * Warning: This swap function does not work!!!
69   * You will know why when you learn pointers in C.
70   * Be patient.
71   */
72  void Swap(int left, int right) {
73    int tmp = left;
74    left = right;
75    right = tmp;
76  }
```

```c
//
// Created by hengxin on 10/16/21.
//

#include <stdio.h>

void Print(char ch, int count);

int main() {
  int lines;
  scanf("%d", &lines);

  for (int i = 0; i < lines; i++) {
    Print(' ', lines - (i + 1));
    Print('*', 2 * i + 1);
    Print(' ', lines - (i + 1));

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

  return 0;
}

void Print(char ch, int count) {
  for (int i = 0; i < count; i++) {
    printf("%c", ch);
  }
}

```

```c
//
// Created by hengxin on 9/27/21.
//

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

int main() {
  int high = 100;
  int number_of_tries = 7;

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

  /**
   * (2) generate a random number (name: r)
   * between 1 and high
   */
  // stdlib.h: standard library
  srand(time(NULL));
  int r = rand() % high + 1;
  printf("r = %d.\n", r);

  // rand(): 0 .. RAND_MAX
  int random = rand();
  printf("random = %d; RAND_MAX = %d. \n", random, RAND_MAX);
  /**
   * (3) ask the player to input a guess (name: guess)
   */
  while(number_of_tries > 0) {
    // number_of_tires--: number_of_tries = number_of_tries - 1
    printf("You still have %d tries.\n", number_of_tries);
    number_of_tries--;

    printf("Please input your guess.\n");

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

    // compare x and y: x == y
    // it is not x = y
```

```
53      if (guess == r) {
54        printf("Congs! You win! \n");
55        break;
56      } else if (guess > r) {
57        printf("guess > r.\n");
58      } else { // guess < r
59        printf("guess < r.\n");
60      }
61    }
62
63    /**
64     * (5) repeat (3)-(4) until the player wins or loses
65     * while number_of_tires > 0
66     */
67
68    return 0;
69  }
70
71
```

```
# 5-function

## Functions
- `leap.c`: function definition, function call, function declaration; pass by value
- `prime.c`: multiple return statements

- `stars.c`: void

- `binary-search.c`: array as parameter; const; smaller than 'len'; comment
  - `bsearch` in library (not ready!; but learn its comment style)
- `merge.c`: array as parameter; two arrays
- `selection-sort.c`: array as parameter; update the array
  - `qsort.c` in library (how to return an array; also learn its comment style)
- `swap.c`: pass by value

- `palindorm.c`: string as parameter; const; without length; two functions

- `game-of-life.c`: pass by value + array as parameter; multi-dimensional array
- `guess.c`: left as an exercise

## Backup
- `string.h`
- `random.c`
```