```c
/**
 * Bubble Sort Algorithm
 * See https://en.wikipedia.org/wiki/Bubble_sort
 *
 * Created by hengxin on 10/30/21.
 */

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

#define LEN 100000
//int numbers[LEN] = {6, 5, 3, 1, 8, 7, 2, 4};
int numbers[LEN] = { 0 };

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

  clock_t start = clock();
  /**
   * double "for" version
   */
//  int swapped;
//  for (int i = 0; i < LEN; i++) {
//    swapped = 0;
//    for (int j = 0; j < LEN - 1 - i; j++) {
//      if (numbers[j] > numbers[j + 1]) {
//        int tmp = numbers[j];
//        numbers[j] = numbers[j + 1];
//        numbers[j + 1] = tmp;
//        swapped = 1;
//      }
//    }
//    if (! swapped) {
//      break;
//    }
//  }

  /**
   * The optimized version runs slower!!!
   */
  int swapped;
  int len = LEN;
  int new_len;
  do {
    swapped = 0;
    new_len = 0;
    for (int j = 0; j < len - 1; j++) {
      if (numbers[j] > numbers[j + 1]) {
        int tmp = numbers[j];
        numbers[j] = numbers[j + 1];
        numbers[j + 1] = tmp;
        swapped = 1;
        new_len = j + 1;
      }
    }
//    len--;
    len = new_len;
  } while (swapped);

  /**
   * toupper
   * see https://stackoverflow.com/a/459704
   */
  clock_t diff = clock() - start;
  long sec = diff / CLOCKS_PER_SEC;
  printf("Sorting time taken %d seconds.\n", sec);

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

  return 0;
}

```

```c
/**
 * Test whether two arrays A and B have an element in common
 *
 * Created by hengxin on 10/30/21.
 */

#include <stdio.h>
#define LEN_A 5
#define LEN_B 5

int a[LEN_A] = {1, 3, 5, 7, 9};
int b[LEN_B] = {2, 4};

int main() {
//  int found = 0;

  /**
   * "break" version
   */
//  for (int i = 0; i < LEN_A && (! found); i++) {
//    for (int j = 0; j < LEN_B; j++) {
//      if (a[i] == b[j]) {
//        found = 1;
//        break;
//      }
//    }
//  }
//
//  if (found) {
//    printf("Yes.\n");
//  } else {
//    printf("No.\n");
//  }

  /**
   * "goto" version
   */
  for (int i = 0; i < LEN_A; i++) {
    for (int j = 0; j < LEN_B; j++) {
      if (a[i] == b[j]) {
        printf("Yes.\n");
        goto found;
      }
    }
  }
  printf("No.\n");

found:

  return 0;
}

```

```c
/**
 * file: continue.c
 *
 * Created by hengxin on 10/30/21.
 */

#define LEN 5

int numbers[LEN] = {-2, 3, -4, 5, 6};

int main() {
  for (int i = 0; i < LEN; i++) {
    if (numbers[i] < 0) { // skip negative elements
      continue;
    }

    // if () // skip ...

    // do positive elements (which may be complicated)
  }
  return 0;
}

```

File - D:\cpl\cpl-coding-0\2021-CPL\4-loops\counting-if.c

```c
/**
 * Count the number of occurrences of each digit,
 * of white space characters, and of all other characters.
 *
 * Created by hengxin on 10/16/21.
 */

/**
 * Count the number of occurrences of each digit,
 * of white space characters, and of all other characters.
 */

#include <stdio.h>
#include <ctype.h>
#define LEN 10

int main() {
  int digit_count[LEN] = {0};
  int ws_count = 0;
  int other_count = 0;

  /**
   * "if" version
   * Note: fails to run this program in "Run"
   * See: https://youtrack.jetbrains.com/issue/CPP-5704
   * Use "Terminal" instead.
   * Or use the "input redirection" technique
   */
  char ch;
  while (scanf("%c", &ch) != EOF) {
    if (isdigit(ch)) {
      digit_count[ch - '0']++;
    } else if (isspace(ch)) {
      ws_count++;
    } else {
      other_count++;
    }
  }

  printf("digit_count:");
  for (int i = 0; i < LEN; i++) {
    printf("%d : %d\n", i, digit_count[i]);
  }
  printf("\nws_count: %d\n", ws_count);
  printf("other_count: %d\n", other_count);

  return 0;
}
```

File - D:\cpl\cpl-coding-0\2021-CPL\4-loops\counting-switch.c

```c
/**
 * Count the number of occurrences of each digit,
 * of white space characters, and of all other characters.
 *
 * Created by hengxin on 10/16/21.
 */

#include <stdio.h>
#define LEN 10

int main() {
  int digit_count[LEN] = {0};
  int ws_count = 0;
  int other_count = 0;

  /**
   * "switch-case" version
   * Note: fails to run this program in "Run"
   * See: https://youtrack.jetbrains.com/issue/CPP-5704
   * Use "Terminal" instead.
   * Or use the "input redirection" technique.
   */
  char ch;
  while (scanf("%c", &ch) != EOF) {
    switch (ch) {
      case '0': case '1': case '2': case '3': case '4':
      case '5': case '6': case '7': case '8': case '9':
        digit_count[ch - '0']++;
        break;
      case ' ': case '\n': case '\t':
        ws_count++;
        break;
      default:
        other_count++;
        break;
    }
  }

  printf("digit_count:");
  for (int i = 0; i < LEN; i++) {
    printf(" %d", digit_count[i]);
  }
  printf("\nws_count: %d\n", ws_count);
  printf("other_count: %d\n", other_count);

  return 0;
}


```

```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 SIZE 6
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}};

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

  for (int row = 1; row <= SIZE + 1; row++) {
    for (int col = 1; col <= SIZE + 1; col++) {
      printf("%c ", old_board[row][col] ? '*' : ' ');
    }
    printf("\n");
  }
  printf("\033c");

  int new_board[SIZE + 2][SIZE + 2];

  for (int tick = 1; tick < 10; tick ++) {
    for (int row = 1; row <= SIZE; row++) {
      for (int col = 1; col <= SIZE; 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];

        if (old_board[row][col]) {
          new_board[row][col] = (neighbours == 2 || neighbours == 3);
        } else {
          new_board[row][col] = (neighbours == 3);
        }
      }
    }

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

    /**
     * sleep
     * see https://stackoverflow.com/a/10923084
     */
    sleep(2);
    /**
     * Clear the console (in Linux)
     * Warning: This is not robust!!! Just for demonstration.
     * See https://stackoverflow.com/a/43884673/1833118
     *
     * In Windows, use
     *     #include <conio.h>
     *     clrscr();
     */
    printf("\033c");

    for (int row = 0; row < SIZE + 2; row++) {
      for (int col = 0; col < SIZE + 2; col++) {
        old_board[row][col] = new_board[row][col];
      }
    }

  }

  return 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};

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;
}

```