```c
//
// Created by hfwei on 2022/11/1.
//
// You can even call `main` in the `main` function.
// WARNING: Do not write code like this!!!
//

#include <stdio.h>

int main(int argc, char *argv[]) {
  if (argc == 1) {
    return 0;
  }

  printf("%s", argv[argc - 1]);

  main(argc - 1, argv);

  return 0;
}
```

```c
#include <stdio.h>
/**
 * file: gcd-euclid-iter.c
 *
 * Visualization (gcd(64, 18) for illustration):
 *  https://pythontutor.com/c.html#code=int%20main%28%29%20%7B%0A%0A%20%20int%20a%20%3D%2064%3B%0A%20%20int%20b%20%3D%2018%3B%0A%0A%20%20while%20%28a%20!%3D%20b%29%20%7B%0A%20%20%20%20if%20%28a%20%3E%20b%29%20%7B%0A%20%20%20%20%20%20a%20%3D%20a%20-%20b%3B%0A%20%20%20%20%7D%20else%20%7B%0A%20%20%20%20%20%20b%20%3D%20b%20-%20a%3B%0A%20%20%20%20%7D%0A%20%20%7D%0A%20%20%7D%0A%20%20%0A%20%20return%200%3B%0A%7D&curInstr=28&mode=display&origin=opt-frontend.js&py=c_gcc9.3.0&rawInputLstJSON=%5B%5D
 *
 * Created by hengxin on 11/01/22.
 */

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

  while (a != b) {
    if (a > b) {
      a = a - b;
    } else {
      b = b - a;
    }
  }

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

  return 0;
}

```

```c
/**
 * file: fib-iter.c
 *
 * Iteratively computing the first n Fibonacci numbers with an array.
 *
 * Created by hengxin on 11/13/21.
 */

#include <stdio.h>
#include <limits.h>

#define LEN 93

int main() {
  long long fibs[LEN] = {0, 1};

  int n;
  scanf("%d", &n);

  for (int i = 2; i < n; i++) {
    fibs[i] = fibs[i - 1] + fibs[i - 2];
    printf("%lld ", fibs[i]);
  }

//  for (int i = 0; i < n; i++) {
//    printf("%lld ", fibs[i]);
//  }
//
//  printf("\n%lld\n", LLONG_MAX);

  return 0;
}
```

```c
#include <stdio.h>
/**
 * file: fib-iter.c
 *
 * Iteratively computing the first n Fibonacci numbers without using an array.
 *
 * Created by hengxin on 11/13/21.
*/

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

  long long fib1 = 0;
  long long fib2 = 1;
  printf("%lld %lld ", fib1, fib2);

  long long fib3;
  for (int i = 3; i < n; i++) {
    fib3 = fib1 + fib2;
    printf("%lld ", fib3);

    fib1 = fib2;
    fib2 = fib3;
  }

  return 0;
}
```

```c
/**
 * file: gcd-euclid.c
 *
 * Euclid's algorithm:
 *
 * if a > b
 * then gcd(a, b) = gcd(a - b, b)
 * else gcd(a, b) = gcd(a, b - a)
 *
 * Created by hengxin on 11/01/22.
 */

#include <stdio.h>

int GCDEuclid(int a, int b);

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

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

  return 0;
}

int GCDEuclid(int a, int b) {
  if (a == b) {
    return a;
  }

  if (a > b) {
    return GCDEuclid(a - b, b);
  }

  if (a < b) {
    return GCDEuclid(a, b - a);
  }
}
```

```c
/**
 * file: merge-sort.c
 *
 * Created by hengxin on 11/14/21.
 */

#include <stdio.h>

// #define LEN 8
#define LEN 7

void MergeSort(int nums[], int left, int right);

/**
 * Merge two subarrays nums[left .. mid] and nums[mid + 1 .. right]
 *
 * @param nums
 * @param left
 * @param mid
 * @param right
 */
void Merge(int nums[], int left, int mid, int right);

int main() {
  // int numbers[LEN] = {6, 5, 3, 1, 8, 7, 2, 4};
  int numbers[LEN] = {38, 27, 43, 3, 9, 82, 10};
  MergeSort(numbers, 0, LEN - 1);

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

  return 0;
}

void MergeSort(int nums[], int left, int right) {
  if (left == right) {
    return;
  }

  int mid = (left + right) / 2;
  MergeSort(nums, left, mid);
  MergeSort(nums, mid + 1, right);

  Merge(nums, left, mid, right);
}

void Merge(int nums[], int left, int mid, int right) {
  // make a copy of nums[left .. right]
  int size = right - left + 1;
  // use VLA (variable-length arrays)
  // introduced since C99
  // but made optional since C11
```

```
54   int copy[size];
55   for (int i = 0, j = left; i < size; i++, j++) {
56     copy[i] = nums[j];
57   }
58
59   int left_index = left;
60   int right_index = mid + 1;
61   int copy_index = 0;
62
63   while (left_index <= mid && right_index <= right) {
64     if (nums[left_index] <= nums[right_index]) {
65       copy[copy_index] = nums[left_index];
66       left_index++;
67     } else {
68       copy[copy_index] = nums[right_index];
69       right_index++;
70     }
71
72     copy_index++;
73   }
74
75   while (left_index <= mid) {
76     copy[copy_index] = nums[left_index];
77     left_index++;
78     copy_index++;
79   }
80
81   while (right_index <= right) {
82     copy[copy_index] = nums[right_index];
83     right_index++;
84     copy_index++;
85   }
86
87   // copy back
88   for (int i = 0, j = left; i < size; i++, j++) {
89     nums[j] = copy[i];
90   }
91 }
```

```c
/**
 * file: sum-re.c
 *
 * Recursively computing the sum of an array of integers.
 *
 * Visualization: https://pythontutor.com/visualize.html#code=%23include%20%3Cstdio.h%3E%0A%0Aint%20Sum%28int%20numbers%5B%5D,%20int%20len%29%3B%0A%0Aint%20main%28%29%20%7B%0A%20%20%20%20int%20numbers%5B%5D%20%3D%20%7B1,%202,%203,%204,%205%7D%3B%0A%0A%20%20%20%20int%20sum%20%3D%20Sum%28numbers,%20sizeof%20numbers%20/%20sizeof%20numbers%5B0%5D%29%3B%0A%20%20%20%20printf%28%22sum%20%3D%20%25d%5Cn%22,%20sum%29%3B%0A%0A%20%20%20%20return%200%3B%0A%7D%0A%0Aint%20Sum%28int%20numbers%5B%5D,%20int%20len%29%20%7B%0A%20%20%20%20if%20%28len%20%3D%3D%200%29%20%7B%0A%20%20%20%20%20%20%20%20return%200%3B%0A%20%20%20%20%7D%0A%0A%20%20%20%20int%20partial_sum%20%3D%20Sum%28numbers,%20len%20-%201%29%3B%0A%0A%20%20%20%20int%20sum%20%3D%20numbers%5Blen%20-%201%5D%20%2B%20partial_sum%3B%0A%0A%20%20%20%20return%20sum%3B%0A%7D&cumulative=false&heapPrimitives=nevernest&mode=edit&origin=opt-frontend.js&py=c_gcc9.3.0&rawInputLstJSON=%5B%5D&textReferences=false
 *
 * Created by hengxin on 11/01/22.
 */

#include <stdio.h>

int Sum(const int numbers[], int len);

int main() {
  int numbers[] = {1, 2, 3, 4, 5};

  int sum = Sum(numbers, sizeof numbers / sizeof numbers[0]);
  printf("sum = %d\n", sum);

  return 0;
}

int Sum(const int numbers[], int len) {
  if (len == 0) {
    return 0;
  }

//  return numbers[len - 1] + Sum(numbers, len - 1);

  int partial_sum = Sum(numbers, len - 1);

  int sum = numbers[len - 1] + partial_sum;

  return sum;
}
```

```c
/**
 * Recursively computing the greatest common divisor of two integers
 *
 * Euclidean algorithm:
 * gcd(a, b) = gcd(b, a % b)
 *
 * Visualization (gcd(64, 48) for illustration):
 *   https://pythontutor.com/c.html#code=int%20GCD%28int%20a,%20int%20b%29%3B%0A%0Aint%20main%28%29%20%7B%0A%20%20int%20a%20%3D%2064%3B%0A%20%20int%20b%20%3D%2048%3B%0A%0A%20%20printf%28%22gcd%28%25d,%20%25d%29%20%3D%20%25d%5Cn%22,%20a,%20b,%20GCD%28a,%20b%29%29%3B%0A%0A%20%20return%200%3B%0A%7D%0A%0A//%20gcd%28130,%20124%29%20%3D%202%0A//%20gcd%28662,%20414%29%20%3D%202%0Aint%20GCD%28int%20a,%20int%20b%29%20%7B%0A%20%20if%20%28b%20%3D%3D%200%29%20%7B%0A%20%20%20%20return%20a%3B%0A%20%20%7D%0A%0A%20%20return%20GCD%28b,%20a%20%25%20b%29%3B%0A%7D&curInstr=17&mode=display&origin=opt-frontend.js&py=c_gcc9.3.0&rawInputLstJSON=%5B%5D
 *
 * Created by hengxin on 11/13/21.
 */

#include <stdio.h>

int GCD(int a, int b);

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

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

  return 0;
}

// gcd(130, 124) = 2
// gcd(414, 662) = 2
int GCD(int a, int b) {
  if (b == 0) {
    return a;
  }

  return GCD(b, a % b);
}
```

```c
/**
 * file: fib.c
 *
 * Recursively computing the n-th Fibonacci number
 *
 * Visualization (for n = 4): https://pythontutor.com/render.html#code=%23include%20%3Cstdio.h%3E%0A%0Along%20long%20Fib%28int%20n%29%3B%0A%0Aint%20main%28%29%20%7B%0A%20%20int%20n%20%3D%204%3B%0A%0A%20%20printf%28%22%25lld%5Cn%22,%20Fib%28n%29%29%3B%0A%7D%0A%0Along%20long%20Fib%28int%20n%29%20%7B%0A%20%20if%20%28n%20%3D%3D%200%29%20%7B%0A%20%20%20%20return%200%3B%0A%20%20%7D%0A%0A%20%20if%20%28n%20%3D%3D%201%29%20%7B%0A%20%20%20%20return%201%3B%0A%20%20%7D%0A%0A%20%20return%20Fib%28n%20-%201%29%20%2B%20Fib%28n%20-%202%29%3B%0A%7D&cumulative=false&curInstr=55&heapPrimitives=nevernest&mode=display&origin=opt-frontend.js&py=c_gcc9.3.0&rawInputLstJSON=%5B%5D&textReferences=false
 *
 * Created by hengxin on 11/01/22.
 */

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

long long Fib(int n);

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

  clock_t start = clock();

  printf("%lld\n", Fib(n));

  clock_t end = clock();
  double time = ((double) end - start) / CLOCKS_PER_SEC;
  printf("time = %f (sec)\n", time);

  return 0;
}

long long Fib(int n) {
  if (n == 0) {
    return 0;
  }

  if (n == 1) {
    return 1;
  }

  return Fib(n - 1) + Fib(n - 2);
}
```

```c
/**
 * Binary Search: the recursive version
 *
 * Created by hengxin on 11/14/21.
 */

#include <stdio.h>
#define LEN 10

int BinarySearch(int key, int dict[], int low, int high);

int main() {
  int dictionary[LEN] = {0, 1, 1, 2, 3, 5, 8, 13, 21, 34};

  int key;
  scanf("%d", &key);

  printf("The index of %d is %d.\n", key,
         BinarySearch(key, dictionary, 0, LEN - 1));

  return 0;
}

int BinarySearch(int key, int dict[], int low, int high) {
//  if (low == high) {
//    if (dict[low] == key) {
//      return low;
//    }
//    return -1;
//  }

  if (low > high) {
    return -1;
  }

  int mid = (low + high) / 2;

  if (dict[mid] == key) {
    return mid;
  }

  if (dict[mid] > key) {
    return BinarySearch(key, dict, low, mid - 1);
  }

  return BinarySearch(key, dict, low + 1, high);
}

```

```c
/**
 * file: min-re.c
 *
 * Recursively find the minimum of an array of integers
 *
 * Visualization: https://pythontutor.com/visualize.html#code=%23include%20%3Cstdio.h%3E%0A%0A%23define%20NUM%205%0Aint%20numbers%5BNUM%5D%20%3D%20%7B25,%2034,%2037,%2045,%2043%7D%3B%0A%0Aint%20Min%28const%20int%20nums%5B%5D,%20int%20len%29%3B%0A%0Aint%20main%28%29%20%7B%0A%20%20int%20min%20%3D%20Min%28numbers,%20NUM%29%3B%0A%20%20printf%28%22min%20%3D%20%25d%5Cn%22,%20min%29%3B%0A%0A%20%20return%200%3B%0A%7D%0A%0Aint%20Min%28const%20int%20numbers%5B%5D,%20int%20len%29%20%7B%0A%20%20if%20%28len%20%3D%3D%201%29%20%7B%0A%20%20%20%20return%20numbers%5B0%5D%3B%0A%20%20%7D%0A%0A%20%20int%20partial_min%20%3D%20Min%28numbers,%20len%20-%201%29%3B%0A%20%20return%20numbers%5Blen%20-%201%5D%20%3C%20partial_min%20%3F%20numbers%5Blen%20-%201%5D%20%3A%20partial_min%3B%0A%7D&cumulative=false&heapPrimitives=nevernest&mode=edit&origin=opt-frontend.js&py=c_gcc9.3.0&rawInputLstJSON=%5B%5D&textReferences=false
 *
 * Created by hengxin on 11/01/22.
 */

#include <stdio.h>

#define NUM 5
// Just for illustration. Avoid global variables.
int numbers[NUM] = {25, 34, 37, 45, 43};

int Min(const int nums[], int len);

int main() {
  int min = Min(numbers, NUM);
  printf("min = %d\n", min);

  return 0;
}

int Min(const int nums[], int len) {
  if (len == 1) {
    return nums[0];
  }

  int partial_min = Min(nums, len - 1);
  return nums[len - 1] < partial_min ? nums[len - 1] : partial_min;
}
```

```c
/**
 * file: gcd.c
 *
 * Iteratively computing the greatest common divisor of two integers.
 *
 * Euclidean algorithm:
 * gcd(a, b) = gcd(b, a % b)
 *
 * Created by hengxin on 11/01/22.
 */

#include <stdio.h>

int GCD(int a, int b);

int main() {
  int a = 130;
  int b = 124;

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

  return 0;
}

int GCD(int a, int b) {
  int tmp;
  while (b != 0) {
    tmp = b;
    b = a % b;
    a = tmp;
  }

  return a;
}
```

```
 1 # 6-recursion
 2
 3 ## Recursion
 4 - `sum.c`
 5 - `min.c`
 6 - `fib.c`
 7 - `gcd.c`
 8 - `binary-search.c`
 9 - `merge-sort.c`
10
11 ## Backup
12 - `hanoi.c`
13 - `quicksort.c`
```