```
1958 1961 1969 1954 1969
```

```c
// Created by hfwei on 2024/10/10.

#include <stdio.h>

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

  // TODO: print stars pyramid
  for (int i = 0; i < lines; ++i) {
    // print lines - 1 - i spaces
    for (int j = 0; j < lines - 1 - i; ++j) {
      printf(" ");
    }

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

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

  return 0;
}
```

```c
// Created by hfwei on 2024/10/10.

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

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

  // TODO: print primes between 1 and max

  int count = 0;

  for (int number = 2; number <= max; number++) {
    // decide whether number is a prime
    // Since C99: bool (macro extended to _Bool; with macros true and false)
    // Since C23: will become keywords (bool, true, false); do not need stdbool.h
    bool is_prime = true;
    for (int factor = 2; factor * factor <= number; factor++) {
      if (number % factor == 0) {
        is_prime = false;
        break;  // test: number = 18
      }
    }

    if (is_prime) { // TODO: is_prime == 1; is_prime != 0
      count++;
      printf("%d ", number);  // TODO: comment this for performance
    }
  }

  printf("\ncount = %d\n", count);

  return 0;
}
```

```markdown
# `3-for-a-while`

## `stars.c`

- double loops
- `for (int i = 0) + for (int j = 0)`

## `primes.c`

- double loops
- `int is_prime = 1;`: why 1? why not 0?
- `if (is_prime)` vs. `if (is_prime != 0)` vs. `if (is_prime == 1)`
- testing
  - https://www.wolframalpha.com/input?i=+primes+less+than+100000
  - mma: `PrimePi[100000]`
  - `number = 2`
- `break`
- `i * i <= number` vs. `i * i < number`
- `stdbool.h`
  - C89, C99, C23
  - `bool b = 5`
  - `(bool) 3.5`
- [x] timing
  - `clock_t start = clock(); clock_t end = clock(); (end - start) / CLOCKS_PER_SEC`

# `binary-search.c`

- already sorted array
  - Fib
- `int index = -1;`
  - `printf`
- `break`
- testing
  - `1`: the leftmost/rightmost one
  - search for the leftmost/rightmost one
- [ ] learn from the standard library???
  - `(low + high) / 2`
    - `low + (high - low) / 2`
    - [ ] try it???

## `digits.c`
```

```markdown

- testing
- `do-while`

## `selection-sort.c`

- preparation: scanf
- with comments
- `swap`
- `while (scanf ...)`
  - https://en.cppreference.com/w/c/io/fscanf
    - Number of receiving arguments successfully assigned (which may be zero in case a matching failure occurred before
      the first receiving argument was assigned)
    - or `EOF` if input failure occurs before the first receiving argument was assigned
  - How to run this?
    - Linux: `Ctrl + D` at the beginning of a line
    - Mac: `Cmd + D` at the beginning of a line
    - Windows: `Ctrl + Z` at the beginning of a line
- more `printf` (after each iteration)
- `sizeof`
- Input&Output indirection
  - Linux/Windows Cmd

## `palindrome.c`

- `#define`: pre-processing
- `scanf("%20s", string);`
- `strlen`
- comma expression
- `for` version
- `while` version
```

```c
// Created by hfwei on 2024/10/10.

#include <stdio.h>

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

  // Initialize the number of digits to 1
  int num_of_digits = 1;

  // For numbers other than 0, adjust number of digits
  for (; number / 10 != 0; num_of_digits++) {
    number /= 10;
  }

  printf("Number of digits is %d\n", num_of_digits);

  return 0;
}
```

```c
// Created by hfwei on 2024/10/10.

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

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

int main() {
  // example: nolemon,nomelon
  printf("Input a string containing at most 20 characters.\n");
  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);

  // TODO: test palindrome

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

  // TODO: the while version
  bool is_palindrome = true;

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

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

  return 0;
}
```

```
# for
add_executable(stars stars.c)
add_executable(primes primes.c)

# while (do-while)
add_executable(binary-search binary-search.c)
add_executable(binary-search-for binary-search-for.c)

add_executable(digits-while digits-while.c)
add_executable(digits-do-while digits-do-while.c)
add_executable(digits-for digits-for.c)

# for-a-while
add_executable(selection-sort selection-sort.c)
add_executable(palindrome palindrome.c)
```

```c
// Created by hfwei on 2024/10/10.

#include <stdio.h>

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

  // TODO: number of digits
  int num_of_digits = 0;

  // TODO: >= (infinite loop)
  if (number == 0) {
    num_of_digits = 1;
  } else {
    while (number > 0) {
      number /= 10;
      num_of_digits++;
    }
  }

  printf("Number of digits is %d\n",
         num_of_digits);

  return 0;
}
```

```c
// Created by hfwei on 2024/10/10.

#include <stdio.h>

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

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

  // TODO: binary search: search for key in dictionary[]
  int low = 0;
  int high = LEN - 1;

  int index = -1;

  while (low <= high) {
    int mid = (low + high) / 2;

    if (key > dictionary[mid]) {
      low = mid + 1;
    } else if (key < dictionary[mid]) {
      high = mid - 1;
    } else {  // key == dictionary[mid]
      index = mid;
      // break;  // what if `break` is removed
      high = mid - 1; // find the leftmost index of the key
    }
  }

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

  return 0;
}
```

```c
// Created by hfwei on 2024/10/10.

#include <stdio.h>

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

int main(void) {
  /*
   * Input the array
   *
   * Note: fails to run this program in "Run" (Ctrl + D)
   * See: https://youtrack.jetbrains.com/issue/CPP-5704
   * Use "Terminal" instead.
   *
   * TODO: CLion; Terminal
   * Linux: Ctrl + D (works now; in the new line, or Ctrl + D twice)
   *   See https://stackoverflow.com/a/21365313/1833118 (send and clear the buffer)
   * Windows: Ctrl + Z (does not work on my platform)
   * TODO: Input&Output redirection
   *   See https://stackoverflow.com/a/11788475/1833118
   */
  int len = -1;
  while (scanf("%d", &numbers[++len]) != EOF);

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

  // TODO: selection sort
  for (int i = 0; i < len; i++) {
    // find the minimum value of numbers[i .. n-1]
    int min = numbers[i];
    int min_index = i;

    for (int j = i + 1; j <= len - 1; ++j) {
      if (numbers[j] < min) {
        min = numbers[j];
        min_index = j;
      }
    }

    // swap numbers[i] and numbers[min_index]
    int temp = numbers[i];
    numbers[i] = numbers[min_index];
    numbers[min_index] = temp;
  }

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

  return 0;
}
```

```c
// Created by hfwei on 2024/10/10.

#include <stdio.h>

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

  // TODO: number of digits
  int num_of_digits = 0;

  do {
    number /= 10;
    num_of_digits++;
  } while (number > 0);

  printf("Number of digits is %d\n",
         num_of_digits);

  return 0;
}
```

```c
// Created by hfwei on 2024/10/10.

#include <stdio.h>

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

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

  int index = -1;

  for (int low = 0, high = LEN - 1; low <= high; ) {
    int mid = (low + high) / 2;

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

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

  return 0;
}
```