```c
// Created by hengxin on 2024/12/04.

int main() {
  char **argv;

  int *names[10];

  int (*musician_score_table)[10];

  int *StrCpyStd(char *dest, const char *src);

  int (*comp)(const void *left, const void *right);

  // see https://en.cppreference.com/w/c/program/atexit
  int atexit(void (*func)(void));

  // see https://en.cppreference.com/w/c/program/signal
  void (*signal(int sig, void (*handler)(int)))(int);

  char (*(*func(int num, char *str))[3])(void);

  char (*(*arr[3])(void))[5];

  // Refer to https://cdecl.org/ for more practice.
  // See https://c-faq.com/decl/spiral.anderson.html for secrets!!!
}
```

```c
// Created by hengxin on 2024/12/04.

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

void PrintInts(const int integers[], size_t len);
void PrintStrs(const char *str[], size_t len);

int main() {
  int integers[] = {-2, 99, 0, -743, 2, INT_MIN, 4};
  int size_of_integers = sizeof integers / sizeof *
integers;

  PrintInts(integers, size_of_integers);
  // TODO: Sort integers using qsort
  PrintInts(integers, size_of_integers);

  const char *names[] = {
      "Luo Dayou",
      "Cui Jian",
      "Dou Wei",
      "Zhang Chu",
      "Wan Qing",
      "Li Zhi",
      "Yao",
      "ZuoXiao",
      "ErShou Rose",
      "Hu Mage",
  };
  size_t size_of_names = sizeof names / sizeof *names;

  PrintStrs(names, size_of_names);
  // TODO: Sort strings using qsort
  PrintStrs(names, size_of_names);
}

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

```c
}

void PrintStrs(const char *str[], size_t len) {
  printf("\n");
  for (int i = 0; i < len; i++) {
    printf("%s\n", str[i]);
  }
  printf("\n");
}
```

```c
// Created by hfwei on 2024/12/04.
// See https://en.cppreference.com/w/c/program/atexit

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

void f1(void) {
  puts("f1");
}

void f2(void) {
  puts("f2");
}

int main(void) {
  if (!atexit(f1) && !atexit(f2) && !atexit(f2)) {
    return EXIT_SUCCESS;
  }

  // atexit registration failed
  return EXIT_FAILURE;

}   // <- if registration was successful calls f2, f2, f1
```

```c
// Created by hfwei on 2024/12/04.

#include <stdio.h>
#include <signal.h>

void SIGSEGV_Handler(int sig) {
  printf("SIGSEGV %d is caught.\n", sig);
}

int main(void) {
  signal(SIGSEGV, SIGSEGV_Handler);
  raise(SIGSEGV);

  // int *p = NULL;
  // *p = 0;

  return 0;
}
```

```c
// Created by hfwei on 2024/12/04.

#include <stdio.h>

// See https://elixir.bootlin.com/linux/latest/source/include/linux/types.h#L245
typedef int (*cmp_func_t)(const void *a, const void *b);

// See https://elixir.bootlin.com/linux/latest/source/include/linux/bsearch.h#L8
void *bsearch(const void *key,
              const void *base,
              size_t num, size_t size, cmp_func_t cmp);

int main(void) {

  return 0;
}

void *bsearch(const void *key,
              const void *base, size_t num, size_t size,
              cmp_func_t cmp) {
  const char *pivot;
  int result;

  while (num > 0) {
    pivot = base + (num >> 1) * size;
    result = cmp(key, pivot);

    if (result == 0) {
      return (void *) pivot;
    }

    if (result > 0) {
      base = pivot + size;
      num--;
    }

    num >>= 1;
  }

  return NULL;
}
```

```
1 # `11-function-pointers`
2
3 ## `integrate.c`
4
5 ## `sort.c`
6
7 ## `bsearch-gnuc.c`
8
9 ## `decl.c`
```

```c
// Created by hfwei on 2024/12/04.
// A nice function pointer example on Riemann integration:
// https://en.wikipedia.org/wiki/Function_pointer

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

#define NUM_OF_PARTITIONS 1000000

int main() {
  double low = 0.0;
  double high = 1.0;
  double integration = 0.0;

  return 0;
}
```

```c
// Created by hfwei on 2024/12/04.
// Question: What if char key_name[] = "Zhang Chu"?

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

// See https://codebrowser.dev/glibc/glibc/stdlib/stdlib.h.html#__compar_fn_t
typedef int (*__compar_fn_t)(const void *, const void *);

// See https://codebrowser.dev/glibc/glibc/bits/stdlib-bsearch.h.html#19
void *bsearch(const void *__key,
              const void *__base, size_t __nmemb, size_t __size,
              __compar_fn_t __compar);

int CompareStrs(const void *left, const void *right);

const char *names[] = {
    "Cui Jian",
    "Dou Wei",
    "ErShou Rose",
    "Hu Mage",
    "Li Zhi",
    "Luo Dayou",
    "Wan Qing",
    "Yao",
    "Zhang Chu",
    "Zhang Chu",
    "Zhang Chu",
    "Zhang Chu",
    "ZuoXiao",
};

int main(void) {
  char *key_name = "Zhang Chu";

  return 0;
}

void *bsearch(const void *__key,
              const void *__base, size_t __nmemb, size_t
```

```
41 __size,
42                __compar_fn_t __compar) {
43   size_t __l, __u, __idx;
44   const void *__p;
45   int __comparison;
46   __l = 0;
47   __u = __nmemb;
48   while (__l < __u) {
49     __idx = (__l + __u) / 2;
50     __p = (const void *) (((const char *) __base) + (__idx
    * __size));
51     __comparison = (*__compar)(__key, __p);
52     if (__comparison < 0) {
53       __u = __idx;
54     } else if (__comparison > 0) {
55       __l = __idx + 1;
56     } else {
57       return (void *) __p;
58     }
59   }
60
61   return NULL;
62 }
63
64 // void *bsearch_leftmost(const void *__key, const void *
   __base,
65 //                        size_t __nmemb, size_t __size,
66 //                        __compar_fn_t __compar) {
67 //   size_t __l, __u, __idx;
68 //   const void *__p;
69 //   int __comparison;
70 //
71 //   __l = 0;
72 //   __u = __nmemb;
73 //   // added by ant
74 //   void *__index = NULL;
75 //
76 //   while (__l < __u) {
77 //     __idx = (__l + __u) / 2;
78 //     __p = (const void *) (((const char *) __base) + (
   __idx * __size));
79 //     __comparison = (*__compar)(__key, __p);
80 //     if (__comparison < 0) {
81 //       __u = __idx;
```

```
82 //     } else if (__comparison > 0) {
83 //       __l = __idx + 1;
84 //     } else {
85 //       // added by ant
86 //       __index = (void *) __p;
87 //       __u = __idx - 1;
88 //     }
89 //   }
90 //
91 //   // added by ant
92 //   return __index;
93 // }
```

```
add_executable(integrate integrate.c)

add_executable(sort sort.c)

add_executable(bsearch bsearch.c)
add_executable(bsearch-gnuc bsearch-gnuc.c)

add_executable(11-decl decl.c)
add_executable(atexit atexit.c)
add_executable(signal signal.c)
```