```c
// Created by hengxin on 11/27/2024.

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);

  int atexit(void (*func)(void));

  void (*signal(int sig, void (*handler)(int)))(int);

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

  char (*(*arr[3])())[5];
}
```

```c
/**
 * Echo program (command-line) arguments.
 *
 * Created by hengxin on 11/27/2024.
 */

#include <stdio.h>

// arg: argument
// c: count
// v: vector

// argv[0]: the name of the program
// argv[1 .. argc - 1]: command line arguments
// argv[argc]: NULL
int main(int argc, char *argv[]) {
  // for version with argv
  // for (int i = 1; i < argc; ++i) {
  //   printf("argv[%d] = %s\n", i, argv[i]);
  // }

  // for version with pointers
  // for (char **ptr = argv + 1; *ptr != NULL; ptr++) {
  //   printf("%s\n", *ptr);
  // }

  // while version
  // char **ptr = argv + 1;
  // while (*ptr != NULL) {
  //   printf("%s\n", *ptr);
  //   ptr++;
  // }

  // WRONG
  // char **ptr = argv + 1;
  // while (*ptr++ != NULL) {
  //   printf("%s\n", *ptr);
  // }

  char **ptr = argv;
  // *++ptr
  // *ptr++
  // *--ptr
  // *ptr--
```

```c
  while (*++ptr != NULL) {
    printf("%s\n", *ptr);
  }

  return 0;
}
```

```c
// Created by hengxin on 11/27/2024.

// Python Tutor Visualization:
// (1) https://tinyurl.com/scores-of-musicians
// (2) https://tinyurl.com/scores-of-musicians-malloc
// https://tinyurl.com/

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

#define NUM_OF_MUSICIANS 4
#define NUM_OF_SCORES 3

void Print(const int table[][NUM_OF_SCORES], int num_of_musicians);

int main() {
  // C, Java, Python scores of several musicians

  // TODO: (1) initialize scores with a 2D array
  // const int scores[NUM_OF_MUSICIANS][NUM_OF_SCORES] = {
  //     { 0, 10, 20 },
  //     { 10, 20, 30 },
  //     { 20, 30, 40 },
  //     { 30, 40, 50 },
  // };

  // TODO: (2) Dynamically allocate memory for scores
  // malloc here
  int (*scores)[NUM_OF_SCORES] = malloc(NUM_OF_MUSICIANS * sizeof(*scores));
  if (scores == NULL) {
    return 0;
  }

  // fill in data here
  for (int i = 0; i < NUM_OF_MUSICIANS; ++i) {
    for (int j = 0; j < NUM_OF_SCORES; ++j) {
      scores[i][j] = i * j;
    }
  }

  // print it here
  Print(scores, NUM_OF_MUSICIANS);
```

```c

  // ptr_scores here
  int (*ptr_scores)[NUM_OF_SCORES] = scores;
  printf("ptr_scores[3][2] = %d\n",
         (*(ptr_scores + 3))[2]);

  // do not forget to free it
  free(scores);

  return 0;
}

// table: int table[][COL]
// int table[]: int *table
// int table[][COL]: int (*table)[COL]
void Print(const int table[][NUM_OF_SCORES], int num_of_musicians) {
  for (int i = 0; i < num_of_musicians; i++) {
    for (int j = 0; j < NUM_OF_SCORES; j++) {
      // table[i][j]: *(*(table + i) + j)
      // table: int (*)[COL]
      // table + i: int (*)[COL]
      // *(table + i): int *
      // *(table + i) + j: int *
      // *(*(table + i) + j): int
      printf("table[%d][%d]: %d\n\n",
             i, j, table[i][j]);

      printf("table: %p\n", table);
      printf("table + %d: %p\n", i, table + i);
      printf("*(table + %d): %p\n", i, *(table + i));
      printf("*(table + %d) + %d: %p\n", i, j, *(table + i) + j);
      printf("*(*(table + %d) + %d): %d\n", i, j, *(*(table + i) + j));
    }
    printf("\n\n");
  }
}
```

```
 1 # 10-double-pointers
 2
 3 ## `selection-sort-strings.c`
 4
 5 - `const`
 6
 7 ## `echo.c`
 8
 9 - Linux `echo`
10 - C standard
11 - `printf("%s\n", argv[i])`: printf the nullptr
12
13 ## `scores.c`
14
15 - `student_score_table`: as a 2D array
16 - `Print`
17   - `int table[][COLS]` vs. `int (*table)[COLS]`
18 - `malloc`
19   - `int *`
20   - `int (*)[COLS]`
```

```
 1 add_executable(selection-sort-ints selection-sort.c)
 2 add_executable(selection-sort-strings selection-sort-
   strings.c)
 3
 4 add_executable(scores scores.c)
 5
 6 add_executable(echo echo.c)
 7
 8 add_executable(pointers-malloc pointers-malloc.c)
 9
10 add_executable(decl decl.c)
```

```c
// Created by hfwei on 2024/11/27.

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

#define LEN 5

void SelectionSort(int *arr, int len);
int GetMinIndex(const int *arr, int begin, int end);
void Swap(int *left, int *right);
void Print(const int *arr, int len);

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

  int *numbers = malloc(len * sizeof(*numbers));

  if (numbers == NULL) {
    return EXIT_FAILURE;
  }

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

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

  free(numbers);
}

void SelectionSort(int *arr, int len) {
  for (int i = 0; i < len; i++) {
    int min_index = GetMinIndex(arr, i, len);
    Swap(arr + i, arr + min_index);
  }
}

int GetMinIndex(const int *arr, int begin, int end) {
  int min = arr[begin];
  int min_index = begin;

```

```c
  for (int i = begin + 1; i < end; ++i) {
    if (arr[i] < min) {
      min = arr[i];
      min_index = i;
    }
  }

  return min_index;
}

void Swap(int *left, int *right) {
  int temp = *left;
  *left = *right;
  *right = temp;
}

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

```c
// Created by hfwei on 2024/11/27.

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

#define LEN 3
#define ROW 3
#define COL 4

int main(void) {
  // (1) One-Dimensional Array
  // Visualization: https://tinyurl.com/pointers-malloc-int

  // malloc
  int *array = malloc(LEN * sizeof *array);
  if (array == NULL) {
    printf("malloc failed\n");
    return EXIT_FAILURE;
  }

  // fill in
  for (int i = 0; i < LEN; ++i) {
    array[i] = i * i;
  }

  // print
  for (int i = 0; i < LEN; ++i) {
    printf("%d ", array[i]);
  }
  printf("\n\n");

  // free
  free(array);

  // (2) Two-Dimensional Array
  // Visualization: https://tinyurl.com/pointers-malloc-int-array

  // malloc
  int (*table)[COL] = malloc(ROW * sizeof *table);

  // fill in
  for (int i = 0; i < ROW; ++i) {
```

```c
    for (int j = 0; j < COL; ++j) {
      table[i][j] = i * j;
    }
  }

  // print
  for (int i = 0; i < ROW; ++i) {
    for (int j = 0; j < COL; ++j) {
      printf("%d ", table[i][j]);
    }
    printf("\n");
  }
  printf("\n");

  // free
  free(table);

  // (3) One-Dimensional Array of Pointers
  // Visualization: https://tinyurl.com/pointers-malloc-arraypointers

  // malloc and fill in
  int *array_of_pointers[LEN];
  for (int i = 0; i < LEN; ++i) {
    array_of_pointers[i] = malloc((i + 1) * sizeof *array_of_pointers[i]);
    for (int j = 0; j < i + 1; ++j) {
      array_of_pointers[i][j] = i * j;
    }
  }

  // print
  for (int i = 0; i < LEN; ++i) {
    for (int j = 0; j < i + 1; ++j) {
      printf("%d ", array_of_pointers[i][j]);
    }
    printf("\n");
  }
  printf("\n");

  // free
  for (int i = 0; i < LEN; ++i) {
    free(array_of_pointers[i]);
  }
```

```c

  // (4) Two-Dimensional Array with Potentially Non-Contiguous Memory
  // Visualization: https://tinyurl.com/pointers-malloc-int-pp

  // malloc
  int **matrix = malloc(ROW * sizeof *matrix);
  for (int i = 0; i < ROW; ++i) {
    matrix[i] = malloc(COL * sizeof *matrix[i]);
  }

  // fill in
  for (int i = 0; i < ROW; ++i) {
    for (int j = 0; j < COL; ++j) {
      matrix[i][j] = i * j;
    }
  }

  // print
  for (int i = 0; i < ROW; ++i) {
    for (int j = 0; j < COL; ++j) {
      printf("%d ", matrix[i][j]);
    }
    printf("\n");
  }
  printf("\n");

  // free
  for (int i = 0; i < ROW; ++i) {
    free(matrix[i]);
  }
  free(matrix);

  // (5) Two-Dimensional Array with Contiguous Memory
  // Visualization: https://tinyurl.com/pointers-malloc-int-p

  // malloc
  int *matrix_contiguous = malloc(ROW * COL * sizeof *matrix_contiguous);

  // fill in
  for (int i = 0; i < ROW; ++i) {
```

```c
    for (int j = 0; j < COL; ++j) {
      matrix_contiguous[i * COL + j] = i * j;
    }
  }

  // print
  for (int i = 0; i < ROW; ++i) {
    for (int j = 0; j < COL; ++j) {
      printf("%d ", matrix_contiguous[i * COL + j]);
    }
    printf("\n");
  }

  // free
  free(matrix_contiguous);

  return 0;
}
```

```c
// Created by hfwei on 2024/11/27.

// Python Tutor Visualization: https://tinyurl.com/array-of-musicians (LEN 3)
// https://tinyurl.com/

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

#define LEN 10

void SelectionSort(const char *arr[], int len);
int GetMinIndex(const char *arr[], int begin, int end);
void Swap(const char **left, const char **right);
void Print(const char *arr[], int len);

int main(void) {
  const char *musicians[LEN] = {
      "Luo Dayou",
      "Cui Jian",
      "Dou Wei",
      "Zhang Chu",
      "Wan Qing",
      "Li Zhi",
      "Yao",
      "ZuoXiao",
      "ErShou Rose",
      "Hu Mage",
  };

  Print(musicians, LEN);
  SelectionSort(musicians, LEN);
  Print(musicians, LEN);

  return 0;
}

void SelectionSort(const char *arr[], int len) {
  for (int i = 0; i < len; i++) {
    int min_index = GetMinIndex(arr, i, len);

    // arr: char *[]
    // char**
    // const char **
```

```c
    Swap(arr + i, arr + min_index);
  }
}

int GetMinIndex(const char *arr[], int begin, int end) {
  const char *min = arr[begin];
  int min_index = begin;

  for (int i = begin + 1; i < end; ++i) {
    if (strcmp(arr[i], min) < 0) {
      min = arr[i];
      min_index = i;
    }
  }

  return min_index;
}

void Swap(const char **left, const char **right) {
  const char *temp = *left;
  *left = *right;
  *right = temp;
}

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