```c
//
// Created by hengxin on 10/19/22.
//

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

#define LEN_L 4
#define LEN_R 5

int *Merge(const int L[], int llen, const int R[], int rlen);

int main() {
  int L[LEN_L] = {2, 4, 6, 8};
  int R[LEN_R] = {1, 3, 5, 7, 9};

  int *merge = Merge(L, LEN_L, R, LEN_R);

  for (int i = 0; i < LEN_L + LEN_R; i++) {
    printf("%d ", merge[i]);
  }

  return 0;
}

int *Merge(const int L[], int llen, const int R[], int rlen) {
  int *merge = malloc((llen + rlen) * sizeof *merge);

  int l = 0;
  int r = 0;
  int m = 0;

  while (l < llen && r < rlen) {
    if (L[l] <= R[r]) {
      merge[m++] = L[l];
      l++;
    } else { // L[l] > R[r]
      merge[m++] = R[r];
      r++;
    }
  }

  while (l < llen) {
    merge[m++] = L[l];
    l++;
  }

  while (r < rlen) {
    merge[m++] = R[r];
    r++;
  }

  return merge;
}
```

```c
//
// Created by hfwei on 2022/12/8.
//

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

#define NUM_OF_PARTITIONS 10000

typedef double (*fp)(double);

double Integrate(double low, double high, double (*fp)(double));
double Square(double x);

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

  integration = Integrate(low, high, Square);
  printf("Integrate(%f, %f, Square) = %f\n", low, high, integration);

  integration = Integrate(low, high, sin);
  printf("Integrate(%f, %f, sin) = %f\n", low, high, integration);

  integration = Integrate(low, high, cos);
  printf("Integrate(%f, %f, cos) = %f\n", low, high, integration);

  // double (*functions[3])(double) = {Square, sin, cos};
  fp functions[3] = {Square, sin, cos};
  for (int i = 0; i < 3; i++) {
    integration = Integrate(low, high, functions[i]);
    printf("integration = %f\n", integration);
  }

  return 0;
}

double Integrate(double low, double high, double (*fp)(double)) {
  double interval = (high - low) / NUM_OF_PARTITIONS;
  double sum = 0.0;

  for (int i = 0; i < NUM_OF_PARTITIONS; i++) {
    double x = low + interval * i;
    double y = fp(x);
    sum += y * interval;
  }

  return sum;
}

double Square(double x) {
  return x * x;
}
```

```c
/**
 * file: sort.c
 *
 * Created by hengxin on 12/01/22.
 *
 * A nice function pointer example on Riemann integration:
 * https://en.wikipedia.org/wiki/Function_pointer
 */

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

int CompareInts(const void *left, const void *right);
void PrintInts(const int *integers, int len);

int CompareStrs(const void *left, const void *right);
void PrintStrs(const char *str[], int len);

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

  /**
   * void qsort( void *ptr, size_t count, size_t size,
   *          int (*comp)(const void *, const void *) );
   */
  int (*comp)(const void *, const void *) = CompareInts;
  // you should not do this!!!
  // printf("sizeof comp : %zu\n", sizeof comp);
  printf("comp : %p\n", comp);
  printf("*comp : %p\n", *comp);
  printf("CompareInts : %p\n", CompareInts);
  printf("&CompareInts : %p\n", &CompareInts);

  qsort(integers, size_of_integers, sizeof *integers, comp);
  PrintInts(integers, size_of_integers);

  // Call functions indirectly via function pointers.
  int a = 10;
  int b = 20;
  printf("%d %s %d\n", a, comp(&a, &b) > 0 ? ">" : "<=", b);

  // Sorting an array of strings
  const char *names[] = {
      "Luo Dayou",
      "Cui Jian",
      "Dou Wei",
      "Zhang Chu",
      "He Yong",
      "Wan Qing",
      "WuTiaoRen",
      "ZuoXiao",
      "Hu Mage",
      "Li Zhi"
  };
  int size_of_names = sizeof names / sizeof *names;

  comp = CompareStrs;
  qsort(names, size_of_names, sizeof *names, comp);
  PrintStrs(names, size_of_names);
}

int CompareInts(const void *left, const void *right) {
  int int_left = *(const int *) left;
  int int_right = *(const int *) right;

  if (int_left < int_right) {
    return -1;
  }

  if (int_left > int_right) {
    return 1;
  }

  return 0;

//   return int_left - int_right; // erroneous shortcut (fails if INT_MIN is present)
}

// actual arguments: char *const *
int CompareStrs(const void *left, const void *right) {
  char *const *pp1 = left;
  char *const *pp2 = right;
  return strcmp(*pp1, *pp2);
}

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

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