1) Compare the triplets:

```
#include <assert.h>
#include <ctype.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char* readline();
char* ltrim(char*);
char* rtrim(char*);
char** split_string(char*);

// Complete the compareTriplets function below.

/*
* To return the integer array from the function, you should:
*     - Store the size of the array to be returned in the
result_count variable
*     - Allocate the array statically or dynamically
*
* For example,
* int* return_integer_array_using_static_allocation(int*
result_count) {
*     *result_count = 5;
*
*     static int a[5] = {1, 2, 3, 4, 5};
*
*     return a;
* }
```

```
*
* int* return_integer_array_using_dynamic_allocation(int*
result_count) {
*     *result_count = 5;
*
*     int *a = malloc(5 * sizeof(int));
*
*     for (int i = 0; i < 5; i++) {
*         *(a + i) = i + 1;
*     }
*
*     return a;
* }
*
*/
int* compareTriplets(int a_count, int* a, int b_count, int* b, int*
result_count) {

     int i;
     *result_count = 2;
     static int arr[2] = {0, 0};
     for(i=0; i<3; i++){
         if(a[i] > b[i]) {
             arr[0]++;
         }
         if(a[i] < b[i]) {
             arr[1]++;
         }
     }
     return arr;
}

int main()
{
   FILE* fptr = fopen(getenv("OUTPUT_PATH"), "w");
```

```
    char** a_temp = split_string(rtrim(readline()));

    int* a = malloc(3 * sizeof(int));

    for (int i = 0; i < 3; i++) {
        char* a_item_endptr;
        char* a_item_str = *(a_temp + i);
        int a_item = strtol(a_item_str, &a_item_endptr, 10);

        if (a_item_endptr == a_item_str || *a_item_endptr != '\0') {
exit(EXIT_FAILURE); }

        *(a + i) = a_item;
    }

    int a_count = 3;

    char** b_temp = split_string(rtrim(readline()));

    int* b = malloc(3 * sizeof(int));

    for (int i = 0; i < 3; i++) {
        char* b_item_endptr;
        char* b_item_str = *(b_temp + i);
        int b_item = strtol(b_item_str, &b_item_endptr, 10);

        if (b_item_endptr == b_item_str || *b_item_endptr != '\0') {
exit(EXIT_FAILURE); }

        *(b + i) = b_item;
    }

    int b_count = 3;

    int result_count;
```

```
   int* result = compareTriplets(a_count, a, b_count, b,
&result_count);

   for (int i = 0; i < result_count; i++) {
       fprintf(fptr, "%d", *(result + i));

       if (i != result_count - 1) {
           fprintf(fptr, " ");
       }
   }

   fprintf(fptr, "\n");

   fclose(fptr);

   return 0;
}

char* readline() {
   size_t alloc_length = 1024;
   size_t data_length = 0;
   char* data = malloc(alloc_length);

   while (true) {
       char* cursor = data + data_length;
       char* line = fgets(cursor, alloc_length - data_length, stdin);

       if (!line) {
           break;
       }

       data_length += strlen(cursor);

       if (data_length < alloc_length - 1 || data[data_length - 1] ==
'\n') {
           break;
```

```
        }

        alloc_length <<= 1;

        data = realloc(data, alloc_length);

        if (!data) {
            data = '\0';

            break;
        }
    }

    if (data[data_length - 1] == '\n') {
        data[data_length - 1] = '\0';

        data = realloc(data, data_length);

        if (!data) {
            data = '\0';
        }
    } else {
        data = realloc(data, data_length + 1);

        if (!data) {
            data = '\0';
        } else {
            data[data_length] = '\0';
        }
    }

    return data;
}

char* ltrim(char* str) {
    if (!str) {
```

```c
        return '\0';
    }

    if (!*str) {
        return str;
    }

    while (*str != '\0' && isspace(*str)) {
        str++;
    }

    return str;
}

char* rtrim(char* str) {
    if (!str) {
        return '\0';
    }

    if (!*str) {
        return str;
    }

    char* end = str + strlen(str) - 1;

    while (end >= str && isspace(*end)) {
        end--;
    }

    *(end + 1) = '\0';

    return str;
}

char** split_string(char* str) {
    char** splits = NULL;
    char* token = strtok(str, " ");

    int spaces = 0;

    while (token) {
        splits = realloc(splits, sizeof(char*) * ++spaces);

        if (!splits) {
            return splits;
        }

        splits[spaces - 1] = token;

        token = strtok(NULL, " ");
    }

    return splits;
}
```

2) A very Big Sum

```
#include <assert.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
```

```c
#include <stdlib.h>
#include <string.h>

char* readline();
char** split_string(char*);

// Complete the aVeryBigSum function below.
long aVeryBigSum(int ar_count, long* ar) {
  long long int sum=0;

    for(int i=0;i<ar_count;i++)
    {
        sum=sum+ar[i];
    }

    return sum;

}

int main()
{
   FILE* fptr = fopen(getenv("OUTPUT_PATH"), "w");

   char* ar_count_endptr;
   char* ar_count_str = readline();
   int ar_count = strtol(ar_count_str, &ar_count_endptr, 10);

   if (ar_count_endptr == ar_count_str || *ar_count_endptr != '\0') {
exit(EXIT_FAILURE); }

   char** ar_temp = split_string(readline());

   long* ar = malloc(ar_count * sizeof(long));

   for (int i = 0; i < ar_count; i++) {
       char* ar_item_endptr;
```

```
        char* ar_item_str = *(ar_temp + i);
        long ar_item = strtol(ar_item_str, &ar_item_endptr, 10);

        if (ar_item_endptr == ar_item_str || *ar_item_endptr != '\0')
{ exit(EXIT_FAILURE); }

        *(ar + i) = ar_item;
    }

    long result = aVeryBigSum(ar_count, ar);

    fprintf(fptr, "%ld\n", result);

    fclose(fptr);

    return 0;
}

char* readline() {
    size_t alloc_length = 1024;
    size_t data_length = 0;
    char* data = malloc(alloc_length);

    while (true) {
        char* cursor = data + data_length;
        char* line = fgets(cursor, alloc_length - data_length, stdin);

        if (!line) { break; }

        data_length += strlen(cursor);

        if (data_length < alloc_length - 1 || data[data_length - 1] ==
'\n') { break; }

        size_t new_length = alloc_length << 1;
        data = realloc(data, new_length);
```

```c
        if (!data) { break; }

        alloc_length = new_length;
    }

    if (data[data_length - 1] == '\n') {
        data[data_length - 1] = '\0';
    }

    data = realloc(data, data_length);

    return data;
}

char** split_string(char* str) {
    char** splits = NULL;
    char* token = strtok(str, " ");

    int spaces = 0;

    while (token) {
        splits = realloc(splits, sizeof(char*) * ++spaces);
        if (!splits) {
            return splits;
        }

        splits[spaces - 1] = token;

        token = strtok(NULL, " ");
    }

    return splits;
}
```