Exercise 1

This exercise has total 5 points, of which one point is reserved for exceptional-quality code. Each point amounts to one percent of the total module mark. You are not allowed to use any external library in your code, doing so will receive a mark of 0.

All assessment deadlines in this module are strict. Late submissions will get a mark of 0. All submissions must work on the Linux lab machines as specified.

1.1 Welch's t-test

In statistics, Welch's t-test, or unequal variances t-test, is a two-sample location test which is used to test the hypothesis that two populations have equal means. Welch's t-test defines the statistic *t* by the following formula:

$$t = \frac{\overline{X_1} - \overline{X_2}}{S_{\bar{X}_1 - \bar{X}_2}}$$

where

$\overline{X_1}$ = Mean of data for group 1
$\overline{X_2}$ = Mean of data for group 2
S_{X_1} = Standard deviation of data for group 1
S_{X_2} = Standard deviation of data for group 2

$$\overline{X_1} = \frac{1}{n_1}\sum_{i=1}^{n_1} X_{1i}$$
$$\overline{X_2} = \frac{1}{n_2}\sum_{i=1}^{n_2} X_{2i}$$

$$S_{\bar{X}_1 - \bar{X}_2} = \sqrt{\frac{S_{X_1}^2}{n_1} + \frac{S_{X_2}^2}{n_2}}$$

$$S_{X_1}^2 = \frac{1}{n_1 - 1}\sum_{i=1}^{n_1}\left(X_{1i} - \overline{X_1}\right)^2$$
$$S_{X_2}^2 = \frac{1}{n_2 - 1}\sum_{i=1}^{n_2}\left(X_{2i} - \overline{X_2}\right)^2$$

If the value of t is less than 4.5 then the test concludes that the two populations do not differ significantly. Otherwise (i.e. t>4.5) it concludes that the two populations differ significantly.

1.2 Problem statement

A research lab has proposed a new medicine for treating a disease. We want to verify if the medicine has any real impact. Hence, an experiment is performed. Patients are split into two groups. The first group is treated with the new medicine, whereas the second group (also called the 'control group') is treated with a '*placebo'* medicine. Health conditions of the patients are measured and the measurements recorded in a file 'measurements.txt'.

1.2.1 Structure of the file measurements.txt

The contents of the file are as follows:

<n1: number of patients in group1>
<m1_g1: measurement of the 1st patient within group1>
<m2_g1: measurement of the 2nd patient within group1>
…
<mn1_g1: measurement of the n1th patient within group1>

<n2: number of patients in group2>
<m1_g2: measurement of the 1st patient within group2>
<m2_g2: measurement of the 2nd patient within group2>
…
<mn2_g2: measurement of the n2th patient within group2>

For example: Let us assume that the group1 and group2 have 3 and 4 patients respectively. Let the measurements be {45.0, 23.15, 11.98} and {2.45, 11.0, 12.98, 77.80} for group1 and group2 respectively.

The contents of measurements.txt will be:

3
45.0
23.15
11.98

4
2.45
11.0
12.98
77.80

1.2.2 Outliers present in the measured data

The instrument that was used to measure the health conditions, does not always produce correct measurements; some of the measurements can be abruptly low or high. These wrong values are called 'outliers.' The outliers must not be considered during the t-test as they could lead to wrong conclusion.

1.2.3 Detecting outlier values and discarding them

The measurements for a group are sorted and then the median value is obtained. If a measurement lies outside the range [0.5*median to 1.5*median], then it is an outlier and hence should be discarded.

You are provided the function

```
float sort_and_find_median(float *measurements , int size);
```

which receives the pointer '*measurements' pointing to the array containing the measurements and the number of measurements 'size'. The function sorts the array containing the measurements and additionally returns the median element.

2.1 Programming Task1 – Discarding Outliers (2 points):

Open the file 'task1.c' and define the function

```
float *discard_outliers(float *measurements, int size, float median, int *new_size);
```

which receives the pointer to the array of measurements in '*measurements', the number of measurements in 'size', and the median. Additionally, it receives the pointer '*new_size'. The steps to be followed are:

Step1: The function computes the number of outliers in the array of measurements, and then computes `*new_size = size – number_of_outliers;`

Step2: The function creates a new array of length *newsize using malloc as follows:

```
*measurements_wo_outliers = malloc( (*new_size) * sizeof(float) );
```

Step3: The function copies only those measurements that are not outliers in the array pointed by 'measurements_wo_outliers'.

[An example: Suppose, the array of initial measurements contains {1.0, 1.2, 0.9, 4.8}. After sorting the array becomes {0.9,1.0,1.2,4.8}. The median is 1.2. Hence, the allowed range for acceptable measurements is (0.5*1.2 to 1.5*1.2) = (0.6 to 1.8). Clearly, the value 4.8 is an outlier. When the function discard_outliers( ) receives the array of initial measurements, it creates a new array containing {0.9, 1.0, 1.2} of length 3. It returns a pointer to the new array and additionally assigns *new_size=3.]

Compile the code: gcc -Werror -Wall task1_main.c -o task1

Run the program: ./task1 <measurements.txt > measurements_wo_outliers.txt

This command takes data present in 'measurements.txt' as inputs and writes the result in the file 'measurements_wo_outliers.txt'.

We have provided a file 'expected_ measurements_wo_outliers.txt' containing the correct measurements after discarding the outliers.

If your program is correct, then the file 'measurements_wo_outliers.txt' will contain the same data as the file 'expected_ measurements_wo_outliers.txt'.

Use Valgrind to check memory leaks and errors.

2.2 Programming Task2 – Performing a t-test (2 points):

Open the file 'task2.c' and define the function

float t_test(float *measurements1, int size1, float *measurements2, int size2)

which performs the t-test on the measurements of the two groups. It computes the t-static and returns it.

The function receives the pointer '*measurements1' to the array of measurements for group1 along with the number of measurements in size1.

The function receives the pointer '*measurements2' to the array of measurements for group2 along with the number of measurements in size2.

The function performs the following steps:

Step1: Discards the outliers present in '*measurements1' and '*measurements2' and gets two new arrays of valid measurements. You should use your discard_outliers() to for this purpose.

Step2: Follows the equations for Welch's t-test on the two groups of measurements. You can use the sqrt() function available in 'math.h' library for computing the square-root required for computing

$$S_{\bar{X}_1 - \bar{X}_2} = \sqrt{\frac{S_{X_1}^2}{n_1} + \frac{S_{X_2}^2}{n_2}}$$

.

The computed t-static is returned to the main() function.

Compile the code: gcc -Werror -Wall task2_main.c -lm -o task2

Run the program: ./task2 <measurements_wo_outliers.txt

It will print the t-statistic. Check if your program reports the correct t-value.

Use Valgrind to check memory leaks and errors.

2.3 Code submission

Submit both task1.c and task2.c in a single .zip file 't_test.zip'

2.4 Marking scheme: We will test your program with arbitrary inputs.

- Your code must compile on the lab machines as above. If it fails to compile or produces any errors, it will be given 0 marks. Your program must not contain memory leaks nor memory errors.
- 2 points are given if your program correctly computes Task 1 for test vectors of our choice
- 2 points are given if your program correctly computes Task 2 for test vectors of our choice
- 1 point is reserved for exceptional quality code