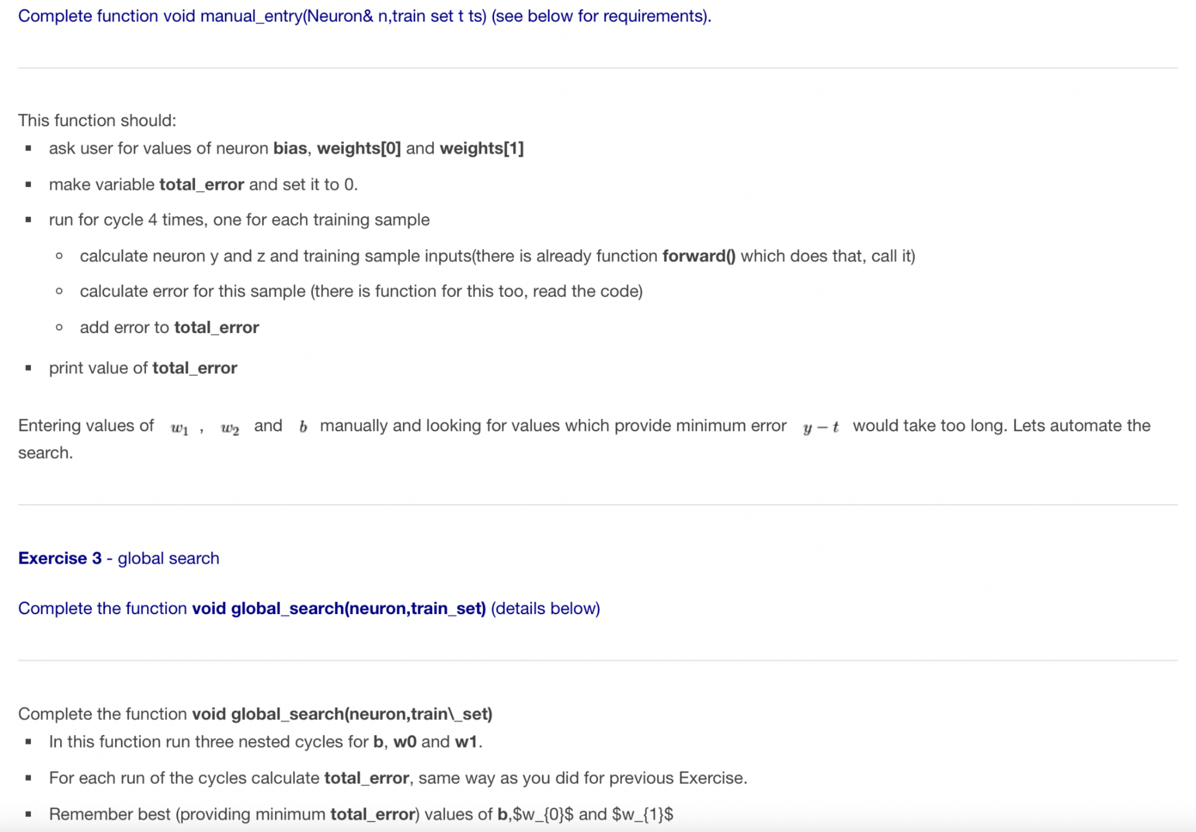
Exercise 2:



Complete function void manual_entry(Neuron& n,train set t ts) (see below for requirements).

This function should:

- ask user for values of neuron **bias**, **weights[0]** and **weights[1]**
- make variable **total_error** and set it to 0.
- run for cycle 4 times, one for each training sample
    - calculate neuron y and z and training sample inputs(there is already function **forward()** which does that, call it)
    - calculate error for this sample (there is function for this too, read the code)
    - add error to **total_error**
- print value of **total_error**

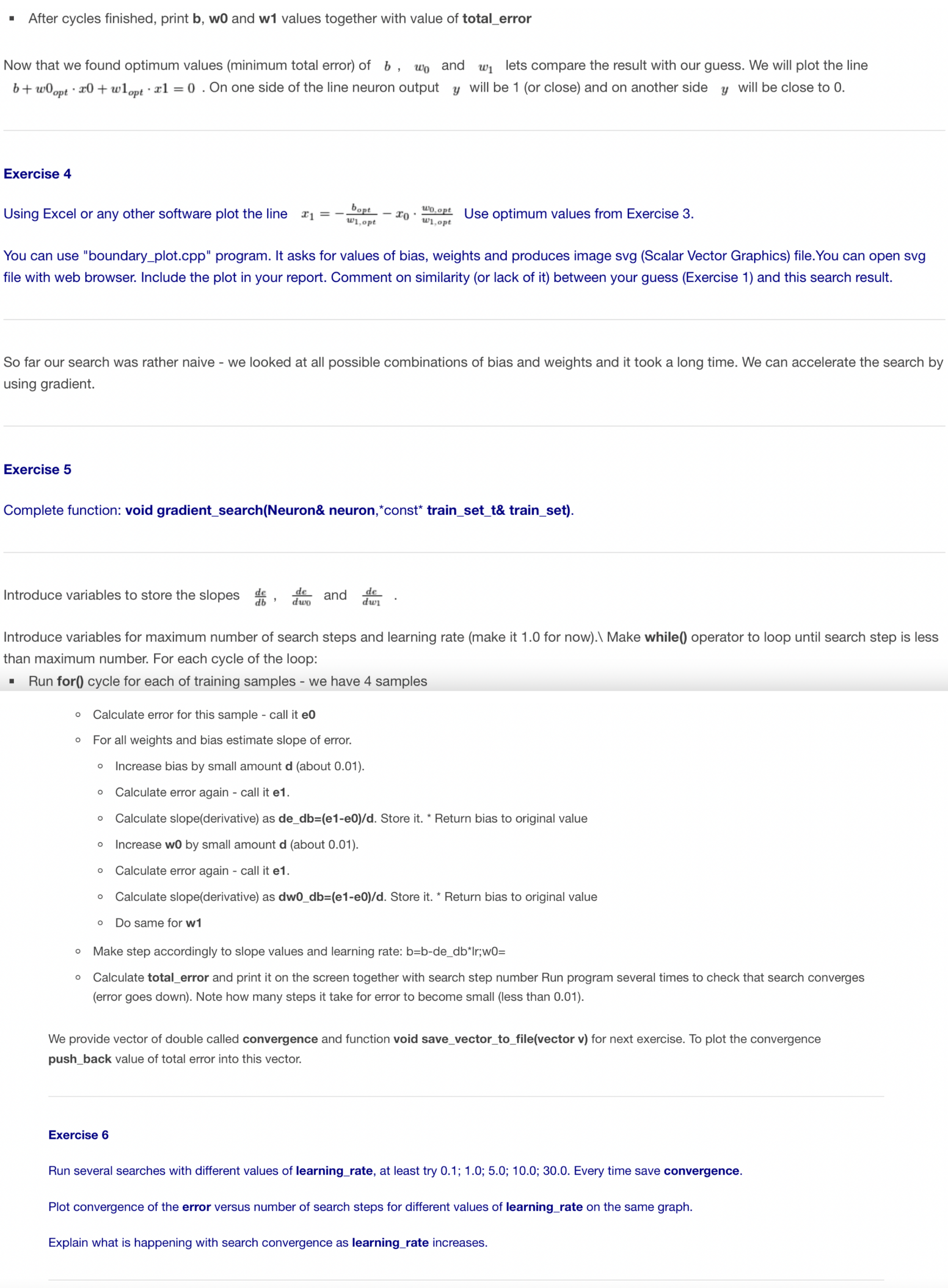
Entering values of w_1 , w_2 and b manually and looking for values which provide minimum error $y-t$ would take too long. Lets automate the search.

Exercise 3 - global search

Complete the function **void global_search(neuron,train_set)** (details below)

Complete the function **void global_search(neuron,train_set)**

- In this function run three nested cycles for **b**, **w0** and **w1**.
- For each run of the cycles calculate **total_error**, same way as you did for previous Exercise.
- Remember best (providing minimum **total_error**) values of **b**,w_{0} and w_{1}



- After cycles finished, print **b**, **w0** and **w1** values together with value of **total_error**

Now that we found optimum values (minimum total error) of b , w_0 and w_1 lets compare the result with our guess. We will plot the line $b + w0_{opt} \cdot x0 + w1_{opt} \cdot x1 = 0$. On one side of the line neuron output y will be 1 (or close) and on another side y will be close to 0.

Exercise 4

Using Excel or any other software plot the line $x_1 = -\frac{b_{opt}}{w_{1,opt}} - x_0 \cdot \frac{w_{0,opt}}{w_{1,opt}}$ Use optimum values from Exercise 3.

You can use "boundary_plot.cpp" program. It asks for values of bias, weights and produces image svg (Scalar Vector Graphics) file.You can open svg file with web browser. Include the plot in your report. Comment on similarity (or lack of it) between your guess (Exercise 1) and this search result.

So far our search was rather naive - we looked at all possible combinations of bias and weights and it took a long time. We can accelerate the search by using gradient.

Exercise 5

Complete function: **void gradient_search(Neuron& neuron**,*const* **train_set_t& train_set)**.

Introduce variables to store the slopes $\frac{de}{db}$, $\frac{de}{dw_0}$ and $\frac{de}{dw_1}$.

Introduce variables for maximum number of search steps and learning rate (make it 1.0 for now).\ Make **while()** operator to loop until search step is less than maximum number. For each cycle of the loop:

- Run **for()** cycle for each of training samples - we have 4 samples
  - Calculate error for this sample - call it **e0**
  - For all weights and bias estimate slope of error.
    - Increase bias by small amount **d** (about 0.01).
    - Calculate error again - call it **e1**.
    - Calculate slope(derivative) as **de_db=(e1-e0)/d**. Store it. * Return bias to original value
    - Increase **w0** by small amount **d** (about 0.01).
    - Calculate error again - call it **e1**.
    - Calculate slope(derivative) as **dw0_db=(e1-e0)/d**. Store it. * Return bias to original value
    - Do same for **w1**
  - Make step accordingly to slope values and learning rate: b=b-de_db*lr;w0=
  - Calculate **total_error** and print it on the screen together with search step number Run program several times to check that search converges (error goes down). Note how many steps it take for error to become small (less than 0.01).

We provide vector of double called **convergence** and function **void save_vector_to_file(vector v)** for next exercise. To plot the convergence **push_back** value of total error into this vector.

Exercise 6

Run several searches with different values of **learning_rate**, at least try 0.1; 1.0; 5.0; 10.0; 30.0. Every time save **convergence**.

Plot convergence of the **error** versus number of search steps for different values of **learning_rate** on the same graph.

Explain what is happening with search convergence as **learning_rate** increases.

- Exercise 2 :

  1) what is the criteria to decide which combination of $bias, w_0, w_1$ is better

2) why neuron is passed into the function by reference?

○ Exercise 4

submit the picture

○ Exercise 6

submit the picture with explanation of how search is influenced by learning rate