Question 4

Write an R function called `polyopt()`, to find a local maximum or minimum of an arbitrary polynomial $h(x) = a_0 + a_1x + a_2x^2 + \ldots + a_px^p$ with $p > 1$ and $a_p \neq 0$, using the Newton-Raphson algorithm with starting value x_0. Your algorithm should terminate when at least one of the following conditions is met:

- $|h'(x_i)| < \varepsilon$;
- $|(h(x_{i+1})-h(x_i))/h(x_i)| < \varepsilon$;
- the number of iterations exceeds some fixed number *N* ;

where x_i is the value of x after the i^{th} iteration and ε and *N* are values to be supplied by the user. The arguments to your function should be: `a`, a vector of coefficients such that `a[1]` represents a_0, `a[2]` represents a_1, and so on; `x0`, the starting value x_0; `tol`, the value of ε; and `MaxIter`, the value of *N*. The default values of `tol` and `MaxIter` should be `1e-6` and `100`, respectively.

Your function should return a list, containing components `x` (the value of *x* when the algorithm terminates), `gradient` (the value of the gradient *h*'(*x*) when the algorithm terminates), `hessian` (the value of the second derivative *h*''(*x*) when the algorithm terminates) and `N.iter`, the number of iterations taken. Your function must not use any of R's own optimisation routines such as `nlm()`, `optim()` or anything similar.

Question 8

Introduction

Consider using data (x_1,y_1), ... (x_n,y_n) to estimate the coefficients β_0 and β_1 in the linear regression model $Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i$ (i=1, ..., n). You are aware that outlying observations can have a large influence on the least-squares fit of such a model. One reason for this is that the least-squares fit minimises the sum of *squared* residuals, and hence penalises large residuals proportionately more than small ones: a residual of magnitude 2 is considered 'four times worse' than a residual of magnitude 1. To reduce the influence of outlying observations therefore, one option is to modify the usual 'sum of squares' function: instead of minimising

$$SS(\beta_0, \beta_1) = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2 ,$$

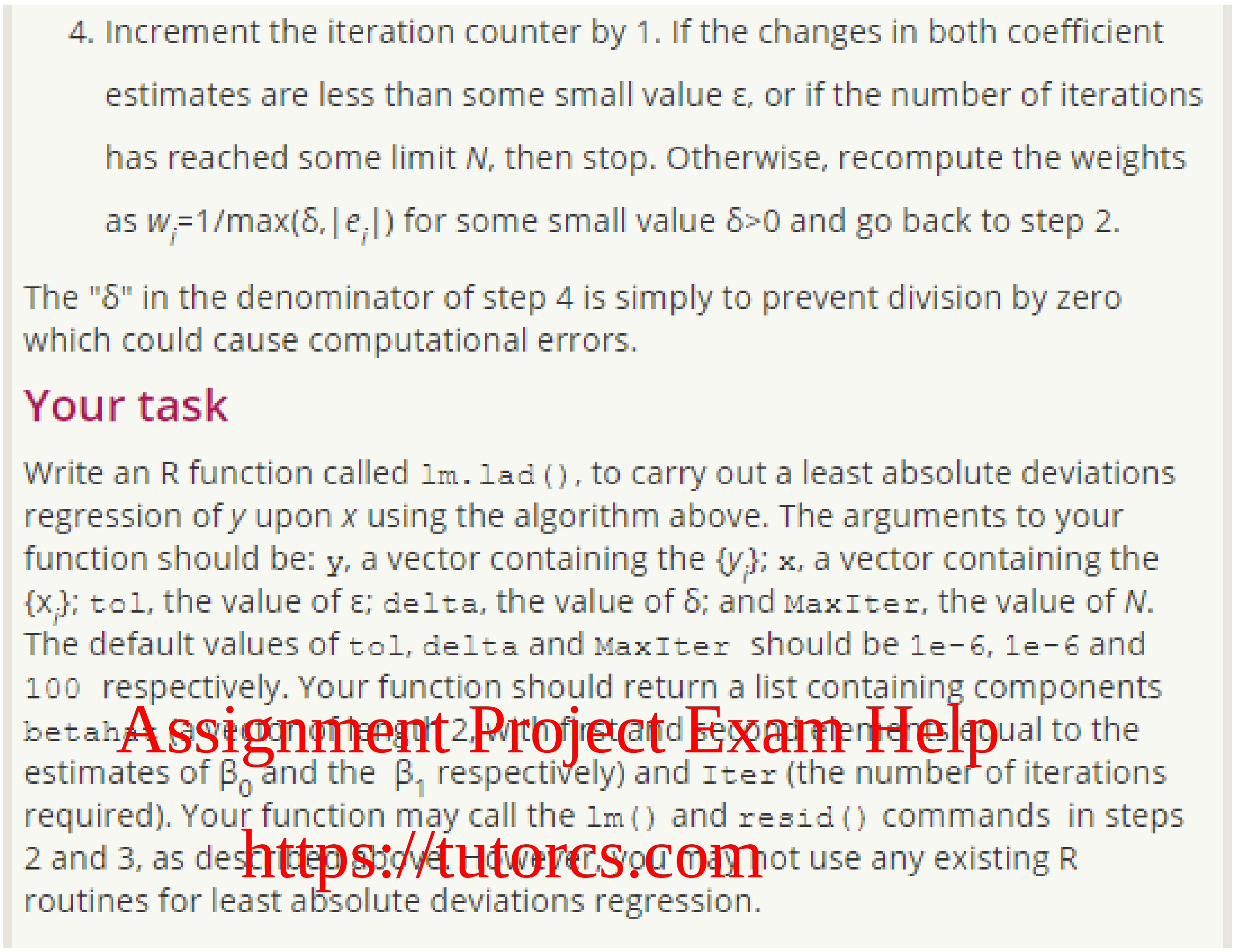
we can minimise the 'sum of absolute deviations' about the regression line:

$$SAD(\beta_0, \beta_1) = \sum_{i=1}^{n} |y_i - \beta_0 - \beta_1 x_i| .$$

This technique is called *least absolute deviations regression*.

Unfortunately, *SAD*(β_0,β_1) cannot be minimised in the usual way by differentiating with respect to β_0 and β_1, because it is not differentiable. However, it *can* be minimised using an iterative weighted least squares technique. The algorithm is as follows:

1. Initialise an iteration counter to 0, and initialise a set of 'weights' {w_i} say, one for each observation: w_i=1, for i=1,...n.
2. (Re-)estimate β_0 and β_1 by minimising the weighted sum of squares
   $$SS(\beta_0, \beta_1) = \sum_{i=1}^{n} w_i (y_i - \beta_0 - \beta_1 x_i)^2 .$$
   In R, this can be done using the `lm()` command, with a weights argument: for example `lm(y ~ x, weights=w)` where `y`, `x` and `w` are vectors containing the values of the {y_i}, {x_i} and {w_i} respectively.
3. Compute the residuals $\left\{e_i = y_i - \hat{\beta}_0 - \hat{\beta}_1 x_i : i = 1, \ldots, n\right\}$, where $\hat{\beta}_0$ and $\hat{\beta}_1$ are the current estimates of β_0 and β_1. If you used the `lm()` command in step 2 and stored the result in an object, you can use the `resid()` command here to compute the residuals from the stored object.



4. Increment the iteration counter by 1. If the changes in both coefficient estimates are less than some small value ε, or if the number of iterations has reached some limit *N*, then stop. Otherwise, recompute the weights as w_i=1/max(δ,|e_i|) for some small value δ>0 and go back to step 2.

The "δ" in the denominator of step 4 is simply to prevent division by zero which could cause computational errors.

Your task

Write an R function called `lm.lad()`, to carry out a least absolute deviations regression of *y* upon *x* using the algorithm above. The arguments to your function should be: `y`, a vector containing the {y_i}; `x`, a vector containing the {x_i}; `tol`, the value of ε; `delta`, the value of δ; and `MaxIter`, the value of *N*. The default values of `tol`, `delta` and `MaxIter` should be `1e-6`, `1e-6` and `100` respectively. Your function should return a list containing components `betahat` (a vector of length 2, with first and second elements equal to the estimates of β_0 and the β_1 respectively) and `Iter` (the number of iterations required). Your function may call the `lm()` and `resid()` commands in steps 2 and 3, as described above. However, you may not use any existing R routines for least absolute deviations regression.