Write a program to calculate the range and size- estimating parameters for a set of n programs where historical object LOC and new and changed LOC data are available. Remember you need to calculate the Linear regression and all the required formulas as explained in previous exercises.

Enhance the linked list of program 1A to hold the n data records, where each record holds two real numbers.

Testing: Thoroughly test the program. At a minimum, use this program to calculate the parameters for three cases, one is the used in the examples table1. The resulting value for Yk and the range should be those delivered in the previous activity of this course.

Work around your data. Calculate the and parameters for the regressions fit of estimated new and changed LOC to actual new changed LOC for the programs that you have developed. Prepare a report of your tests that includes a table of the planned and actual results from these tests.

|  |  |  |
| --- | --- | --- |
| Program Number i | Estimated Object LOC: xi | Actual Object LOC yi |
| 1 | 130 | 186 |
| 2 | 650 | 699 |
| 3 | 99 | 132 |
| 4 | 150 | 272 |
| 5 | 128 | 291 |
| 6 | 302 | 331 |
| 7 | 95 | 199 |
| 8 | 945 | 1890 |
| 9 | 368 | 788 |
| 10 | 961 | 1601 |
| Sum | 3828 | 6389 |
| Average | 382.8 | 638.9 |

Normally your checklist for testing must look like this (NOT HAVING THE WHOLE SET OF TESTS WOULD INVALIDATE YOUR WORK ):

· Equivalence Partitions. The checklist is to prepare evaluations related to:

– Have valid input values proofs been defined?

– Have not valid input values proofs been defined?

– Has a proof been defined for every single identified proof?

· Boundary Analysis.

– Have proofs been defined for valid values at the boundaries? For example, if a value can accept a 1-25 characters long input, proofs must be defined for exactly 1 and 25 characters.

– Have proofs been defined for not valid values at the boundaries? For example, if a value can accept a 1-25 characters long input, proofs must be defined for exactly 0 and 26 characters.

– Have proofs been defined for input values way above the reasonable value for an input? For example, a 1000 kilos person weight.

– Have indexes been extensively proved? For example, when the beginning and the end of a cycle are the same, or if the ending is larger than the beginning, if values are negative or have a larger than permitted value, or do not correspond to the real size of the element to be manipulated.

– Have proofs for inconsistent values been defined? For example, if a numerical input has no consistent format, or if it includes text.

– Have proofs been defined for data with an invalid format? For example an input consisting of text for an e-mail that does not have the proper format, such as “my@mail”.

– Have proofs been defined for input with complex data with a valid/not valid format? Bearing in mind that complex data imply a combination of simple data, we know that there can be multiple combinations of the same data. Hence, we must define individual rules for each data and, from that point on, define the proper rules for the complex data.

– Have proofs been defined for empty or non-existent values? For example, 0, empty string, null object.

– Whenever necessary, if the system points to external entities, have proofs been defined to identify if we have the correct/incorrect reference to the external entity? For example, if such entity saves any particular required state (a non-empty data structure), which other dependencies exist in order to be able to operate.

– Have proofs been made for duplicate entries in the case that such duplicity is or is not permitted?

– Have proofs been made with sorted data? For example, a sorted list. Besides, the order of the objects should be considered, for example, the list of activities required to have a party.

– Have proofs been done regarding the cardinality? For example, what happens if there is no data in the system? Can we generate reports with few data? What is the minimum amount required?

– Have proofs been done to determine whether the response time is within the acceptable frame for the user?

– Whenever necessary, have proofs been done taking in consideration shifts in time zones, hour time, leap-years?

– Have proofs been done modifying the sequence of the execution of the system’s functions or methods? For example, trying to access system functions that require to be logged on, without having previously logged in.

· Proof Quality Test

– Has there been a revision that the output results are correct? We need to be sure that the proofs are well designed, or else we risk having bad code or wasting too much time looking for solutions to problems that don’t exist.

– Whenever possible, have proofs been designed for the case of using the inverse operation to validate results? For example, a Select for an Insert in case of a database, or rising to the power of two the result of a square root.

– Whenever possible, have proofs been defined for different ways to do the calculation in order to check if the results are correct? For example, if we are sorting using the optimum algorithm for computer performance, using another method just to check that the result is correct.

– Whenever necessary, have proofs been done forcing hardware or software errors not related to the system? For example, full disk space, Internet not working, unexpected system crash.

– The application usage trend data analysis has real data reference? Have we conducted proofs to guarantee the volume of operations supported by our system?

USability

– Time

– Performance

– Learnability

– Memorability

– User Preference

– 10 heuristics from Nielsen

Design

– Design patterns