Midterm 1

CSE 3411 Spring 2016

Danilo Barros Mendes

1 - Assuming we have a password for the minimum of x characters and maximum of y characters, including only letters and/or numbers, it leads us to

SUM (36 ^ (x+i)) i = 0, x+i <= y

Where, 36 is the number of valid characters and i goes from 0 to the extent where x plus i is equal to y. That way we include every possible valid entries.

2 - Popper explain that designing an experiment to confirm something already known is not right. When you experiment something you do it to test what you believe. So demonstration is to show that something you believe is actually true, but when you experiment something it is to challenge what you believe to be true.

3 - Because when you multiply a slightly-wrong number to another you propagate the bit with an error much forward than when a sum its made. When you sum 5 with 2 (0101 + 0010) it gives 7 (0111), but when it is multiplied by 2 it gives 10 (1010), which propagated the bit with an error much further.

4 -

- Strings
  - Using the reference program oracle and test the result in Calc against the original string stored in a file, that was used in Excel spreadsheet.
- Fonts
  - Check the accessed libraries in both programs and see what library is called for the specific font.
- Colors
  - Use the reference program oracle. Using a third program to check what color is used in which part of the imported spreadsheet and assign a code to that color, and test it against the code produced by the original spreadsheet.
- Formulas
  - Use the self-verifying data oracle, and test the result from the formula with the expected result. If needed, test further and check the values in the cells used in the formula.
- Table size
  - Could use the constrain oracle to check the size of the table imported to Calc, see if its size is in between the acceptable size. But for this is necessary to know first the size of the table inside Excel.

5 -

- Rules of thumb and order of magnitude
  - These are considered heuristics because neither can guarantee the correct answer to a problem. This is because these are estimations on what something should probably behave or measure.

  - In software testing we can glance it in occasions such as estimating the number of bugs per X lines of code, that some business use. This is an estimation to get a number for something that they are trying to measure.

- Factors of safety
  - The factor of safety is a value that its added/multiplied in order to reduce the effort to achieve a satisfactory answer and it depends on the context. Is much used in scenarios that involve the well being of other people, like the thickness of a steel beam in a bridge.
  - In software testing this heuristic can be seeing in the context of oracles that need a boundary to check whether the object in test has passed or not, many of the times creating a range of safety for the test being accepted

- Resource alocation
  - Alocate new resources as long as the cost go not knowing exceeds the cost of finding out, and the appropriate point in the project is freezing the design.
  - In software testing we can witness this heuristic in the context of memory being the resource, where it is more convenient and cheaper to assign more memory or give the minimum requirements (games) with more memory to the program.

- Risk controlling
  - This try to give the best solution, brushing-up the final state, even in situations that are marginal decidable, although some risk of failure are unavoidable. This heuristic is complemented with other heuristics to determine if it is acceptable and to control the size of the risk.
  - This can be seen in software testing all over the tests, trying to brush up the tests and give the best final result in the software, until the resources are empty.

6 -

- Memory coverage
  - Gain statistics on how much memory is used for a average document on google docs and see which ones are the most memory consuming. This could show what types of data are discrepant compared to the average.
  - The big data in the google docs server would bring a difficulty to the analysis of this.
- Browser compatibility coverage
  - Gain a bigger percentage of the computer users, trying to cover every browser that is commonly used by the population.
  - There are many browsers that are used by only a few, that could not be the main user target for the google docs.
- Import file coverage
  - Check if the most commonly used types of files for documents of the same type are being put as a option of import. This would make the experience much richer and cover a big population of users.
  - Its hard to come up with a library that can parser every type of file to text and in the right format.
- Multiuser coverage
  - Check if the program can handle dozens of users editing at the same time the spreadsheet, without interfering with the experience of one another. This could generate a much more proactive environment with a much higher productivity.

  - Its hard to test and cover the tests that need to be handled at different environments (different computers) and with different people, all at the same time.
- Section of complements coverage
  - Test and cover the complements that can be installed in the Google Docs, to further improve the experience of the user. This could bring a much more effective sense of security in the program and give a richer experience for the user.
  - Its hard to test a library of complements that grows every time a programmer uploads a application to the google store. And once it has been tested its hard to predict how it could behave in the updates to come.