Assignment #1

Sample Solutions

Question 1: C++ Warm-Up

#include <iostream>
#include <string>

using namespace std;

/*
   node struct with each input string held
   in the nodes' content field
*/

struct node
{
   string content;
   node *left;
   node *right;
};

/*
   initially passed the root node of the tree,
   this method traverses it recursively,
   appending L or R as it hits vertices,
   and creating new nodes when necessary.
   The "tail" holds Ls, Rs and parentheses,
   to maintain the original string's purity.
*/
void place(string inpt, string tail, node * root) {
    if (root->content == "") {
        root->content = inpt;
        inpt = inpt + tail + "\n";
        cout << inpt;
    } else if (inpt < root->content) {
        if (root->left == NULL)
            root->left = new node();
        place(inpt, tail + "L", root->left);
    } else {
        if (root->right == NULL)
            root->right = new node();
        place(inpt, tail + "R", root->right);
    }
}

void freeTree(node *root) {
    if (root->left != NULL)
        freeTree(root->left);
    if (root->right != NULL)
        freeTree(root->right);
    delete root;
}

/*
 * main creates the root, outsources the growth to place, and finally deletes it all
 */

int main() {
    node *root = new node();
    string inpt;

    while (cin >> inpt)
        place(inpt, " (", root);
    freeTree(root);
    return 0;
}
Question 2:

Critics of agile software development suggest that the process leads to systems that lack coherence in overall design. How do agile teams ensure that they avoid that outcome?

There needs to be an appropriate balance between the autonomy of individual teams and the overall mission of the company or the goals of the current project. For instance, the management at Spotify aims at clearly conveying the project goals and priorities to the individual teams to maintain the overall coherence. At the same time, they allow flexibility by largely leaving it up to the teams how to achieve these goals.

Question 3:

Which of the twelve principles in the agile manifesto do you think is the least compelling? Why?

Well, this is really a matter of opinion…

Question 4:

In his blog post, Dan Milstein cites the research of Daniel Kahneman in support of his claim that software engineers are bad at making estimates. Under what conditions, however, does Kahneman's research suggest that our ability to make estimates improves?

Kahneman’s research shows that both familiarity with a task through repetition and shorter timescales improve our ability to make accurate estimates. Therefore, it is a good idea to break down large projects into many small, elementary chunks that are similar across projects and can be completed quickly.