**CSE 465/565**

**Term Project**

**Spring 2014**

**Objectives**:

* Gain experience using a variety of programming languages to solve different types of problems.
* Be able to independently learn how to effectively use a new programming language.
* Understand the types of problems that are appropriate/inappropriate for different programming languages.
* Produce a clearly written document describing the pros and cons of using a language to solve a particular problem.

**Languages:** The following list shows the languages, and their classifications, that are to be used for the term project. If you are particularly interested in a language that is not on this list, you may ask for permission to do so.

* **OO languages**
  + Java
  + C++
  + C#
  + Objective C
* **Logic**
  + Prolog
* **Functional**
  + Scheme/Lisp
  + Erlang
  + ML
  + Miranda
  + Haskell
* **Scripting**
  + Perl
  + Ruby
  + JavaScript
  + Python
  + awk/sed/bash

Many of the languages are currently available on easlnx01– see the contents of ~zmudam/lang for samples and instructions on how to compile/invoke these languages.

**Problems and language requirements**: The following list of problems itemizes those that you must solve, in addition to language requirements. It is noted that some of these problems have already been solved by you earlier in the semester – you are allowed to resubmit that code. You are expected to correct problems that were noted in your original submission. Additional handouts will be distributed detailing the requirements of each problem.

|  |  |
| --- | --- |
| **Problem** | **Notes** |
| **Expression Tree** | **Same as problem 8 on HW2. Must create expression tree and be able to print it** |
| **Band Matrix** | **Same as problem problem 6 on HW4** |
| **Puzzle** |  |
| **NHL** |  |
| **Data file** |  |
| **Z+-** | **Same as problem problem 3 on HW1** |
| **Nth cousins M times removed** | **565 students only** |

In addition to the problem-specific language requirements, your final project must include

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Language** | **Min Uses** | **Expr Tree** | **Band Matrix** | **Puzzle** | **NHL** | **Data File** | **Z+-** | **Nth Cousin** |
| **OOL** | **Java** | 0 | At Least One | Must use C++ | - | - | Any 1 | Any 1 | - |
| **C++** | 1 |
| **C#** | 1 |
| **Objective C** | 0 |
| **Logic** | **Prolog** | 1 | - | - | Prolog | - | - | - | - |
| **Functional** | **Scheme** | 1 | Must use Scheme | - | - | - | - | - | - |
| **Erlang** | 0 |
| **ML** | 0 |
| **Miranda** | 0 |
| **Haskell** | 0 |
| **Scripting** | **Perl** | 0 | - | Python | - | Any 2 | Any 1 | Any 1 | - |
| **Ruby** | 0 |
| **Javascript** | 0 |
| **Python** | 1 |
| **awk/sed/bash** | 0 |
| **Number of Solutions** | | 12/13 | 2 | 2 | 2 | 2 | 2 | 2 | 1 |

**Coding**: When solving the problems, you should utilize language features and conventions that are appropriate for each problem. When appropriate:

* Use standard libraries/algorithms
* Use recursion
* Efficient algorithms
* Generic methods/classes
* Avoid memory leaks
* Use delegates/function pointers
* Write portable code
* Use standard stylistic conventions
* Write highly polished code

**Format and grading**: Your paper must be formatted into sections as follows:

* + - One chapter for each of the problems solved, where each chapter should be structured into two main sections.
      * **Language Analysis (50%).** This section will be scored based on writing style, cogency, completeness, content, and overall readability.
        + As a minimum, you should address how well each of the languages allowed you to solve the problem. Does each language provide features/libraries appropriate for the problem? Did you notice appreciable difference in development time using this language and the others? Does each language provide reasonable run-time performance for this problem? Is the language ideal, reasonable, or a poor choice for this problem? You should also address any other issue you feel is relevant to the comparison of the different languages. Be sure to support your statements clearly.
      * **Code (50%).** The grading of this section will be based on whether or not you have written near expert-level source code for each problem. (Place a nicely formatted version at the end of each chapter of your document).
        + Statement of Correctness.Clearly state which requirements you completed and those that were not. This may be as simple as: “All requirements were completed in all languages.”
        + Output. You should also include a copy of output produced by your program. Provide enough output to convince the reader that your program works properly.
        + Source Code.This should be a copy of the source code you developed to solve each problem.
        + Testing. You should include enough tests to demonstrate to the reader that yor program works properly.

.

**Additional Notes:**

* Submit an electronic copy of your report and all source code to the turnins directory.
* You should use only standard libraries installed with the standard programming language.
* Do not use libraries or source code developed by other people, with the following exception: you may use linux utilities if you script using bash, awk, sed.