CSC 461 Semester Project Questions

Alex Murtha, Dylan Shaffer, Dustin Cook

Comparison to Python

1. Location it is used

Perl is used for quick fixes and quick scripts kind of like a “glue.”

Python is used in the same situations but also in larger productions as the main application

1. Where it excels

Perl is very good for text processing

Python is easy to pick up for beginners and a sort of jack of all trades

1. Where it fails

Perl fails when it comes to ease of use and maintainability

Python fails because it is slow relatively

Both languages are dynamically typed which can cause some confusion

1. Portability

Perl and Python are both portable by default with a wide array of supported systems

1. Simplicity

Perl syntax is incredibly difficult to follow and can get messy even in smaller applications. There is a steep learning curve with Perl’s syntax.

Python is extremely easy to read and very simple to pick up and start programming.

1. Orthogonality

Perl is not orthogonal by nature even though it is dynamically typed it is still strict on which operations can be performed on which data. Exceptions are common.

Python is built to have orthogonality to achieve its dynamic nature.

1. Reliability

Perl language maintenance is good and is updated frequently without any new bugs introduced.

Python is reliable due to its secure environment and availability of large support community.

Syntax, Parsing and Binding

1. Is the language compiled or interpreted or a hybrid?

Perl is technically compiled. Perl source is compiled into an internal form (parse tree) and then optimized. This intermediate representation is converted to C code and then compiled with the C compiler.

1. Write the EBNF for an expression in the language

<int> -> {1|2...|9}

<string> -> {a|b...|z}

<function> -> <string>()

<expression> -> <int> (\*|/|-|+) <expression> | <empty>

<assign> -> my ($|@)<expression>|<int>|<string>

<line> -> <assign>|<expression>|<function>

1. How does it handle module/namespace/packages, and what is the scope operator?

Packages are declared with the ‘package’ keyword, everything under the package statement belongs to that package until another package statement is used. The scope operator is ‘::’ for example Package::Module.

1. How is memory management handled (garbage collection, explicit, hints, etc) ?

Perl has automatic memory management using the reference counting method.

Control Flow, Type System, and Data Type Range

1. What the selection and repetition structures of the language, and what is their syntax?

Perl has an if..else if..else structure for selection and also an unless structure which is the inverse of an if. The if syntax is ‘if(<condition>) {…} else if (<condition>) {…} else {…}’ and ‘unless(<condition>) {…}’. Perl 5 has three repetition structures: while, for, and foreach loops, plus variants of each. The syntax for them are as follows:

* while(<condition>) {…}, and its opposite until(<condition>) {…}
* for ($i = 0; $i <= $max; $i++) {…}
* foreach (@array) { // current item in $\_}} or foreach my $key (keys %hash) {…}

1. Is the language static or dynamically typed? Give an example.

Perl is dynamically typed, it will attempt to convert variables to different types as it runs. An example is I can declare a variable with ‘my $var = “hello”’ and also reassign it with ‘$var = 5’.

1. Is the language static or dynamically scoped? Give an example.

Perl 5 has static scoping. An example of this is when a variable is declared inside a block, it is not available outside of that block.

1. What are the built in data types, and their ranges?(list a max of 10).

Perl 5 built in types are as follows:

* Scalars
  + Strings
  + Integer
  + Float
* Arrays
* Hashes

Function/OOP/specialties

1. Are function pass-by-value or pass-by reference, pass-by-value-result, pass-by-name, or a mix. give an example.

Perl 5 functions are pass-by-reference if a variable is declared in an outer scope and passed to a subroutine, if the subroutine changes the value of the variable it will stay changed in the outer scope.

1. How does the language handle modularity, and extensibility? For example, is it class based prototype based, or module based? Does is allow extension to function, classes, objects, files?

Perl 5 is module/package based and does support module inheritance, which allows the classes/objects to be extended.

1. Does the language allow operator overloading, function extension, or function overwriting? Given an example if it does.

Perl 5 does allow operator overloading and function overwriting, but not function extension.

Perl5 overloading is done with the ‘use overload …’ line where you can hypothetically override any sub-routine

1. Describe at least one of the languages specialties.

Perl 5 has regular expressions as a first-class data type, which allows Perl to be very good at text processing.