Programming Languages Midterm 2022

Name: Answer

**1. Prolog is an example of a logic or constraint programming language. What are some other examples of this kind of programming style.**

CSS and HTML are examples. These give layout constraints and the web browser has to place all the elements according to those constraints. Really all layout happens via

Constraints these days. (Android, React, etc….). SQL is a constraint language it specifies what data to get/add, etc, not how to do it. Makefiles don’t specify build order, just how to build each component. Make decides what needs to be built and in what order so that everything in the end is up to date.

In multi paradigm languages, these are showing up more and more. Java streams() foreach structure does not specify order. Basically saying less about how something is done, gives more freedom to the language/compiler/os to choose the best way to do it.

**2. C++ is called multi-paradigm, because it supports several styles of programming. Write a C++ function that computes kp for some double k and some non-negative int p in a procedural style using sums and multiplication (not pow or exp).**

k^p is k multiplied by itself p times:

double mypow(double k, int p) {

double ans = 1;

for (int i=0; i<p; ++i) {

ans = ans \* k;

}

return ans;

}

**3. Now write it in a functional style.**

double mypow(double k, int p) {

return (p == 0) ? 1 : k\*mypow(k,p-1);

}

double mypow(double k, int p) {

if (p == 0) {

return 1;

} else {

return k\*mypow(k,p-1);

}

}

double square(double k) { return k\*k; }

double mypow(double k, int p) {

If (p == 0) {

return 1;

} else {

If (p % 2 == 0) {

return square(mypow(k,p/2));

} else {

return k\*square(mypow(k,(p-1)/2));

}

}

}

**4. Not all languages are Turing complete. Give an example of why Turing-incomplete languages can be useful.**

Limiting the scope can help with simplicity for basic actions. hosts file in using has a simple format to specify hosts vs IP addresses. That makes processing the host file fast, so IP lookups can be fast.

Security is often the biggest reason. Sending a JSON document to a server, can be parsed in the amount of time roughly proportional to the length of the document. If you could send javascript, then you could break any server by sending some js that was an infinite loop or created an arbitrarily large document. This is the reason

json is the way it is. PostScript vs PDF had exactly this problem, which is why PDF was defined. By forcing a limited scope on a language you can guarantee runtimes and actions.

Bitcoin language is limited for this same reason. Transactions cannot stop the verifying nodes from working.

**5. Not all languages are multi-paradigm. Why can restricting the allowed styles of programming be useful?**

It makes it impossible to accidentally introduce another paradigm (and the consequences of that). In C++, you can have a global variable that makes a function call value change every time. So you can’t be sure that any function in C++ wouldn’t suddenly change behavior because of some global you weren’t aware of.

int g = 1;

int f(int x) {

if (g == 1) return x+ 1;

throw range\_error(x);

}

No amount of functional testing will make f(x) functional. There is no control over who/where g is changed.

**6. What is the difference between an object and an immutable object?**

Objects have state (running, fuelLevel) and methods (addFuel(), startCar()). In immutable objects, the state constant after construction (so makeCarWithMoreFuel() and makeStartedCar() would be ok methods, they make different cars with different constant state).

**7. Why do immutable objects and functional programming work well together?**

Functions need arguments. In functional programming, you should not modify those arguments. By having immutable objects you cannot accidentally do that.

Without objects, functions would become pointlessly

complicated. f(Car c) vs f(tire, wheel,fuel,running,….)

**8. Describe the regular expression (a|b)(-(a|b)+)\* in plain English.**

An “a” or “b”, followed by zero or more strings of the form “-ababaaabb” (one hyphen and one or more a/b’s).

Strings of a/b’s of at least length 1, but any length is ok. There has to be a hyphen between the first and second letter, and as many hyphens as you like between a/b substrings.

{a, b, a-a,a-b,b-a,b-b,a-abbababab-aba-b-ab-a-ba-b} etc.

**9. <e> := <n> | <e>+<e> | <e>\*<e>**

**Assume <n> matches a decimal number. Write a parse tree of 3+4\*5+6 that evaluates to 77, and another one that evaluates to 29.**

<e>

(<e>\*<e>)

((<e>+<e>)\*(<e>+<e>))

(((3)+(4))\*((5)+(6)))

So 77

<e>

(<e>+<e>)

((3)+(<e>+<e>))

((3)+((<e>\*<e>)+(6))

((3)+(((4)\*(5))+(6))

So 26

**10. Parsers that attempt to match every part of a grammar with recursive functions or methods is called what kind of parser?**

Top-down or recursive-descent parser.