Problem Set C

Due Tuesday, November 4, 2025 at 6:00pm

(Tripp) Milton Lamb

# [20 points] Briefly explain each of the following programming language terms in your own words. Give one example for each (from any programming language you choose).

## Structured data type

A structured/derived/composite/compound data type is a collection of other data types. The collection consists of native/basic types and other structured data types. Structured data types often (but not always) offer additional functionality past the basic interactions of storage and retrieval. Some examples of common structured data types are arrays, lists, dictionaries, structs, classes, and tuples.

struct Rectangle { // programming language is Rust

x: f64,

y: f64,

}

## Runtime (calling) stack

The runtime calling stack (or simply the “call stack”) is a data structure which is (almost always) autogenerated by the compiler and used by the runtime to keep track of function calls during program execution.

const C = () => ””; // programming language is JavaScript

const B = () => C();

const A = () => B();

A();

run stack example of above program

[global]

[global] -> [A]

[global] -> [A] -> [B]

[global] -> [A] -> [B] -> [C]

[global] -> [A] -> [B] // C returns

[global] -> [A] // B returns

[global] // A returns

# [30 points] For each of the following pairs of programming language concepts, briefly describe an important difference between the two. Do NOT merely define each term – differentiate them!

## Coercion vs. casting

Coercion and casting are both methods of type conversion. Coercion is handled implicitly while casting is explicit. If a language supports coercion it is normally only applied when the new type has more bits than the current type (called a widening conversion). Casting is normally a method called explicitly by the programmer.

## Pass-by-value vs pass-by-reference

Pass-by-value and pass-by-reference are two methods of passing variables into a function. Pass-by-reference passes a pointer to the actual variable while pass-by-value passes a copy of the variable value into the function. If you change a pass-by-reference argument in the function call it will change the value of the original variable, while changing a pass-by-value argument will not affect the original.

## Parameter (formal parameter) vs. argument (actual parameter)

A parameter is the placeholder value in a function interface, while an argument is the actual value sent to a function when the function is called.

# [25 points] Consider a runtime environment with a calling stack.

## What is contained in a stack frame / activation record? Briefly explain the content and purpose of each of the main parts.

## How does the stack represent (part of) the state of the computation, in the denotational semantics sense of defining semantics as state changes caused by a program? It may be helpful to go back and look at the denotational semantics notes.

# [25 points] Assume C/C++ semantics for the following program.

int d, e, f;

void W(int x, int y) {

bool e, j;

Y(d);

}

void Y(int p) {

int k, m;

e = 2 \* p;

p = p + 1;

}

int main() {

int b;

d = 5; b = 2;

W(d, b);

}

## Show what the program call (runtime) stack looks like just **after** Y has been **called**. Identify each stack frame / activation record including (at least) any globals, locals, parameters, static pointers and dynamic pointers. You do not have to include sizes or specific offsets.