Oscar Galindo, Russell Gehan
HW 2

1. (5 points) Variables in PHP are represented by a dollar sign ($) followed by the name of the variable. Discuss the advantages and disadvantages of preceding a PHP variable name with a dollar sign.

The dollar signs improve the readability of the code in PHP, making it easier to determine a program's variables and to differentiate them from keywords. On the other hand, it does slightly hinder writability, since coders have to remember to add the dollar sign to the front of every call to a variable. It can also be a little confusing at times when using the keyword "this" in PHP and when giving variables more than a single dollar sign.

2. (5 points) Define static binding and dynamic binding.

Static: Binding that occurs before run time begins.
Dynamic: Binding that occurs during run time or can change in the course of execution.

3. (5 points) A programming language can be typeless. What are the obvious advantages and disadvantages of having no type in a language?

Some advantages of typeless programming languages falls under writability. It's easier to write code without worrying about the types of your variables. You don't have to typecast for comparisons or operations with variables of different types. One major disadvantage of typeless programming languages falls under reliability. It's more expensive to check types during run time as opposed to compile time, and errors in the program might not be detected until run time as well.

4. (5 points) Dynamic type binding is closely related to implicit heap-dynamic variables. Explain this relationship.

Since dynamic binding involves binding a variable to a type until a value has being given in an assignment statement, implicit heap-dynamic variables are the best option for dynamic binding because, indeed, this type of variables are bound to a storage when they receive a value.

5. (10 points) Define what binding time is. List five different binding times and give an example of each.

Binding time is the time at which a binding between an attribute and variable takes place. The different times as which a binding time can take place is during design time, implementation time, compile time, load time, or run time. For example, declaring a variable as an integer in Java occurs during implementation time. The plus symbol (+) is often bound to addition during design time. Variables are bound to a given type during at compile time in Java and during run time in PHP. Finally, a variable may be bound to a memory cell during load time.

6. (15 points) Consider the following Java assignment statement with one arithmetic operator: x = y + 1.0; For each component of the statement (e.g., variable, operators, and constant), list the various bindings that are required to determine the semantics (meaning) when the statement is executed. For each binding, indicate the binding time
used.

Element	Binding	Binding Time
X	Name: Type: Lifetime: L-value: R-value: Scope	Coding Time Compile time Run Time assuming the the lifetime of the variable is span of time between its memory allocation is allocated and deallocated Run Time Run Time Compile Time
Y	Name: Type: Lifetime: L-value: R-value: Scope	Coding Time Compile Time Run Time assuming the the lifetime of the variable is span of time between its memory allocation is allocated and deallocated Run Time Run Time Compile Time
(=)	Meaning:	Language Design Time

Element	Binding	Binding Time
1.0	Value:	Language Implementation Time
+	Meaning	Language Design Time

7. (15 points) Define static, stack-dynamic, explicit heap-dynamic and implicit heap dynamic variables. What are the advantages and disadvantages of these variables?

Static variables are those that are bound to the same memory cells throughout the entirety of the program's execution. These can be used whenever you want a program to "remember" something, or whenever you want a variable that can be accessed globally. Since static variables are addressed directly, they can also make the program more efficient. However, static variables don't share storage and don't support recursive subprograms.

Stack-dynamic variables are those whose types are statically bound and whose storage bindings are created when program execution reaches their code. These variables allow for recursive subprograms and allow local subprograms to be storage within the same memory space. However, they're not history sensitive and the time required to allocate and reallocate stack-dynamic variables can slow run-time by a very small margin.

Explicit heap-dynamic variables are abstract memory cells that can only be referenced by pointer or reference variables, and they are allocated and deallocated by explicit run-time instructions. These variables can be used to create dynamic structures, such as linked lists or trees, which need to grow/ shrink during program execution. However, pointers and references for these variables can be confusing, and the required storage management can be complex.

Implicit heap-dynamic variables, and all of their attributes, are bound to heap storage when they are assigned values. The advantage of implicit heap-dynamic variables is that they have the highest degree of flexibility, allowing highly generic

code to be written. However, they can slow run-time when maintaining all their dynamic attributes, and the compiler might overlook some errors within the code.

8. (10 points) Java does not support a history sensitive variable in a method. Explain how you can simulate or implement a history sensitive variable in Java, and describe the advantages of supporting a history sensitive variable as a built-in language feature (e.g., in PHP).

In order to implement a history sensitive (static) variable in a method we can declare a static variable within the scope of the class that can be used, accessed, and modified at any scope of the class as long as it is not static. Or, we could write continuously in such a way that we update a specific variable stored in a file outside of our program. This trait is important because it helps to keep track of important information such as number of times a method is called in a class.

9. (10 points) There are several different approaches for determining the type of a variable, including type inference (e.g., Haskell), explicit type declarations (e.g., Java), and dynamic typing (e.g., PHP). Compare type inference and the other two approaches and describe its advantages and disadvantages.

Type inference is an implicit type declaration which uses context to determine the type of a variable. In the simplest case, the context is the type of the value assigned to the variable. This can save programmers some time from cast variables as certain types, when the type can more easily be given with the context of given values. However, being an implicit type declaration does mean that the type is statically bound, which doesn't allow for type flexibility.

An explicit type declaration occurs when a particular type is explicitly given to a variable, whether during declaration or type casting. Explicit type declarations are simple and easy to understand. They also make it easier for the compiler to detect errors before program execution. However, explicit type declarations lack flexibility, rarely ever allowing the type of a variable to change since their types are statically bound.

Dynamic type binding occurs when a variable is assigned a particular value. Since any variables can be assigned any type of value, the type of the program's variables can change any amount of times. Granted, the types of these variables are dynamically bound, which means that they're only temporary. Additionally, it can be harder for the compiler to catch errors for dynamically typed variables before program execution.

10. (10 points) Consider the following Java-like program that uses static scoping.

```
void fun() {
    int a, b, c; /* definition 1 */
    ...
    while (...) {
      int a, c, d; /* definition 2 */
      ... <------------------------ 1
      while (...) {
        int d, e, f; /* definition 3 */
        ... <---------------------- 2
      }
      ... <------------------------- 3
    }
    ... <--------------------------- 4
  }
```

For each of the four statements labeled 1-4 in this function, list all visible variables along with the definition statements (1-3) that define them.

1-. Visible: a, c, d defined at point definition 2 and b defined at point definition 1.

2-. Visible: b, defined at point definition 1, a and c, defined at point definition 2
d, e, f defined at point definition 3

3-. Visible: a, c, d defined at point definition 2 and b defined at point definition 1.

4-. Visible: a, b, c defined at point definition 1.

11. (10 points) Consider the following Java-like program:

```
void main() {
  int a, b, c;
  ...
}
void fun1() {
  int b, c, d;
  ...
}
void fun2() {
  int c, d, e;
  ...
}
void fun3() {
  int d, e, f;
  ...
}
```

Given the following calling sequences and assuming that dynamic scoping is used, what variables are visible during execution of the last function called? Include with each visible variable the name of the function in which it was defined.

(a) main calls fun1; fun1 calls fun2; fun2 calls fun3.
(b) main calls fun1; fun1 calls fun3.
(c) main calls fun2; fun2 calls fun3; fun3 calls fun1.
(d) main calls fun3; fun3 calls fun1.
(e) main calls fun1; fun1 calls fun3; fun3 calls fun2.
(f) main calls fun3; fun3 calls fun2; fun2 calls fun1.

Problem	Visible Variables	Function in which variables are defined
(a)	d, e, f	fun3
	c	fun2
	b	fun1

	a	main
(b)	d, e, f	fun3
	b, c	fun1
	a	main
(c)	b, c, d	fun1
	e, f	fun3
	a	main
(d)	b, c, d	fun1
	e, f	fun3
	a	main
(e)	c, d, e	fun2
	f	fun3
	b	fun1
	a	main
(f)	b, c, d	fun1
	e	fun2
	f	fun3
	a	main