CS4080 Assignment

Sebastian Hernandez

1. Which category of C++ reference variables is always aliases?
   Pointers are used in C++ unions.

2. What is the l-value of a variable? What is the r-value?
   The **l-value** is the address of a variable. The **r-value** is the value of a variable.

3. Define binding and binding time.
   Binding is the association between an entity and one of it's attributes. **Binding time** is the time at which the binding takes place.

4. After language design and implementation, what are the four times bindings can take place in a program?
   1. Compile Time
   2. Load Time
   3. Link Time
   4. Run Time

5. Define static binding and dynamic binding.
   Static binding- first occurs before run time and remains unchanged while the program executes
   Dynamic binding- first occurs during execution or can change during execution

6. What are the advantages and disadvantages of implicit declarations?
   Advantages- writability
   Disadvantages- reliability

7. What are the advantages and disadvantages of dynamic type binding?
   Advantages- flexibility
   Disadvantages- high cost and type error detection is difficult

8. Define static, stack-dynamic, explicit heap-dynamic, and implicit heap-dynamic variables. What are their advantages and disadvantages?
   Static- bound to memory cells before execution begins and remains bound to the same memory cell through execution
   -Advan- efficiency, history-sensitive subprogram support
   -Disdvan- lack of flexibility (no recursion)
   Static-dynamic- storage bindings are created for variables when their declaration statements are elaborated
   -Advan- allows recursion, conserves storage
   -Disdvan- overhead, subprograms can't be history sensitive, inefficient references
   Explicit heap-dynamic- allocated and deallocated by explicit directives which take effect during execution
   -Advan- provides for dynamic storage management
   -Disdvan- inefficient and unreliable

Implicit heap-dynamic- allocation and deallocation caused by assignment statements
-Advan- flexibility
-Disdvan- inefficient (all dynamic attributes), loss of error detection

9. Define lifetime, scope, static scope, and dynamic scope.
   Lifetime- time during which it is bound to a specific memory cell
   Scope- range of statements over which it is visible
   Static scope- method of binding names to nonlocal variables
   Dynamic scope- references to variables are connected to declarations by searching back through the chain of subprogram calls that forced execution to this point

10. What is the general problem with static scoping?
    It is possible to give too much access, and the local variables and subprograms often become global.

11. What is the referencing environment of a statement?
    It is the collection of all names that are visible in the statement.

12. What is a static ancestor of a subprogram? What is a dynamic ancestor of a subprogram?
    The static parent of a subprogram, and its static parents, and so forth up to and including the largest enclosing subprogram are the static ancestors of a subprogram. The dynamic ancestors are the dynamic parents of a subprogram and its dynamic parents and so forth. A dynamic parent is the calling function of a subprogram.

Programming Assignment #2

For my test, I had the three methods create arrays of size 1,000 and ran each declaration 1,000,000 times. After a few test runs, I found an average for the time it took each operation. The static array declaration took the shortest amount of time, with an average of 3242250 nanoseconds. The second shortest time was the declaration onto the stack, with an average of 6985750 nanoseconds. The longest time was by declaring the array onto the heap, which took an average of 6616530500 nanoseconds, or about 6.6 seconds. It makes sense that the static declaration took the shortest amount of time, because the array was bounded to memory cells before the program was even started. This meant that the execution time did not include the time it took to allocate memory cells. Declaring the array on the stack forces the array to wait until execution time to allocate memory for the array. The amount of time that this took was not too long however and was only 3743500 nanoseconds (0.004 seconds) longer. Declaring the array on the heap took much longer, with an average of over six seconds. This is most likely due to the fact that the heap uses pointers, which is much more cumbersome than directly referencing a variable.