CIS 343 – Structure of Programming Languages
Homework Assignment #3, Winter 2019
Topic: Names, Bindings, and Scopes

Due Date (Scanned copy as pdf file by Email): __Friday, March 8th___

Note: My key to this homework will be posted on Blackboard on Saturday, March 9th. So, any submission received after March 8th will not receive any credit for turning in.

Name: Chandler Scott

1. The process of associating an attribute to a name/identifier is called _____.
   a) linking b) binding c) bonding d) chaining
2. The meaning of a name/identifier is determined by the set of _____ bound to that name.
   a) attributes b) values c) locations
3. _____ bindings occur during program execution.
   a) Dynamic b) Static
4. In Java, the range of values for the **short** data type is bound at _____.
   a) compile-time b) language definition time c) run-time d) load time
5. In Java, the location attribute of a local variable is bound at _____.
   a) compile-time b) language definition time c) run-time d) load time
6. If the lifetime of a binding is longer than the lifetime of the object that the binding refers to, then this results in which of the following situations?
   a) garbage b) a dangling reference c) aliasing
7. _____ is an object in the environment with no live bindings to it.
   a) garbage b) a dangling reference c) aliasing
8. In Java, the scope of a field/method (both class and instance) definition includes the entire class.
   a) True b) False
9. If the lifetime of an object is longer than the lifetime of attributes bound to that object, then this results in which of the following situations?
   a) garbage b) a dangling reference c) aliasing
10. If a language permits recursion, _____ allocation of local variables is no longer an option.
    a) stack b) static c) heap
11. _____ fragmentation occurs when a heap storage management algorithm allocates a block that is larger than required to hold a given object; the extra space is then unused.
    a) External b) Internal

12. To eliminate external fragmentation in a heap storage allocation, we must be prepared to do _____ at regular intervals.

    a) heap coalescing b) heap compaction

13. In a language that uses _____ scoping, the scope of an identifier can be determined by simple examination of the source code listing.

    a) Static b) Dynamic

14. _____ scoping is also known as lexical scoping.

    a) Static b) Dynamic

15. _____ scoping cannot be determined merely by studying the source code listing of a program.

    a) Static b) Dynamic

16. In order to determine the _____ scope of a variable, we must first know the sequence in which functions are called during execution of the program.

    a) static b) dynamic

17. The referencing environment (i.e., the set of active bindings) at any given point in a program is determined by the _____ rules applied in the language.

binding

18. The _____ link in a subroutine's frame points to the frame of the most recent invocation of the lexically surrounding subroutine.

    a) static b) dynamic

19. With _____ scoping, errors associated with the referencing environment may not be detected until run time.

    a) static b) dynamic

20. Languages with _____ scoping tend to be interpreted, rather than compiled.

    a) static b) dynamic

21. Which of the following languages does not allow dangling references?

    a) C b) Java c) C++

22. In Java language aliases to objects are created through _____.

    a) assignment by sharing b) deep cloning

23. Two or more names that refer to the same object within the same scope are known as _____.

    a) overloading b) aliasing

24. When a name refers to more than one object with the same scope, this is known as _____.

    a) overloading b) aliasing

25. A program with excessive _____ suffers from program readability.

    a) overloading b) aliasing

26. In a compiled language, when a reference to an overloaded name cannot be resolved, this results in the following type of error:

a) lexical error b) syntax error c) static semantic error d) dynamic semantic error

27. Space for local objects with unlimited extent (or lifetime) must generally be allocated on/in _____.

a) stack b) static area c) heap

28. Given the following overloaded function declarations in C++, the function call **max(5.5, 10)** is ambiguous. This is because C++ allows

```
int max(int x, int y);
double max(double x, double y);
```

a) only automatic widening conversions
b) both automatic widening and narrowing conversions
c) only automatic narrowing conversions

29. Determine output of the following C-like program using static scoping and dynamic scoping, respectively.

```
int a, b, c;

void p() {
   int a = 3;
   b = 1;
   c = a + b;
   q();
}

void print() { printf("%d %d %d\n", a, b, c); }

void q() {
   int b = 4;
   a = 5;
   c = a + b;
   print();
}

main() {
   int c = 5;
   p();
}
```

Output with static scoping: 519

Output with dynamic scoping: 549

30. Suppose the following function declarations/prototypes are available in a program. Write down the number of **func()** function called for each of the eight (8) calls in the table below for each of the languages shown. Write "error" if a call is ambiguous in the language or if certain data type conversion cannot be made.

```
(1)  int func(int, int);
(2)  int func(int, double);
(3)  int func(double, int);
(4)  double func(double, double);
```

Assume the following variable declarations :

```
int  x;
double y;
```

Function Call	Ada	C++	Java
x = func(2,3)	1	1	1
y = func(2,3)	error	1	error
x = func(2,3.2)	error	2	2
y = func(2,3.2)	Error	error	error
x = func(2.1,3)	error	error	3
y = func(2.1,3)	error	3	error
x = func(2.1,3.2)	error	error	4
y = func(2.1,3.2)	4	4	4

31. Consider the following pseudocode:

```
x : integer  -- global

procedure set_x(n : integer)
   x := n

procedure print_x
   write_integer(x)

procedure first
   set_x(1)
   print_x

procedure second
   x : integer
   set_x(2)
   print_x


// program starts here
set_x(0)
first()
print_x
second()
print_x
```

(a) What does this program print if the language uses static scoping?

output 1020

(b) What does it print if the language uses dynamic scoping?

output 1122