Arab Academy for Science and Technology and Maritime Transport
College of Computing and Information Technology

Course	Structure of Programming Languages (CS445)
Lecturer	Prof. Dr. Mohamed Kholief
TA	Mahmoud ElMorshedy

Sheet – 01 – Ch.5[1]

Review Questions:

7) Define binding and binding time.

- **Binding:** is an association between an entity and an attribute, such as between a variable and its type, or between an operation and a symbol.
- **Binding Time:** is the time at which binding takes place.

8) After language design and implementation, what are the four times binding can take place in a program?

- Consider the following example in C:

```
static float pi = 3.14159
float area = pi * pow(r, 2)
```

- **Language design time:** the asterisk symbol (*) is bound to the multiplication operation.
- **Language implementation time (compiler design time):** the range of possible values and precision is bound to the float datatype.
- **Compile time:** the float datatype is bound to the variable “area”
- **Load time:** the static variable pi is bound to a memory cell.
- **Link time:** the call to the math library function “pow()” is bound to the subprogram code.
- **Runtime:** the value of the variable “area” is bound to a memory cell.

[1] Refer to the textbook (Concepts of Programming Languages – Robert W. Sebesta – 10th Edition)

9) Define static binding and dynamic binding.

- A binding is **static** if it first occurs **before run time** and remains **unchanged** throughout program execution.
- A binding is **dynamic** if it first occurs **during execution** or **can change** during execution of the program

12) Define static, stack-dynamic, explicit heap-dynamic, and implicit heap-dynamic variables. What are the advantages and disadvantages?

- Lifetime of a variable is the time during which the variable is bound to a memory cell (between allocation and deallocation of the variable). Can be:

	Definition	Advantages	Disadvantages
Static	bound to a memory cell before execution and remains bound to the same memory cell during execution.	- Efficiency (direct addressing) - History-sensitive subprogram support.	- Lack of flexibility (no recursion)
Stack-dynamic	Storage bindings are created for variables when their declaration statements are exexuted.	- allows recursion - conserves storage	- Overhead of allocation and deallocation - Subprograms cannot be history sensitive - Inefficient references (indirect addressing)
Explicit heap-dynamic	Allocated and deallocated by explicit directives, specified by the programmer, which take effect during execution (e.g. malloc, free in C or new, delete in C++).	- provides for dynamic storage management	- inefficient and unreliable
Implicit heap-dynamic	Allocation and deallocation caused by assignment statements (e.g. strings and arrays in JS).	- flexibility (generic code)	- Inefficient, because all attributes are dynamic - Loss of error detection

16) What is the referencing environment of a statement?

- The referencing environment of a statement is the collection of all names that are visible in the statement

Problem Set:

6) Consider the following JavaScript skeletal program:

```
// The main program
var x;
function sub1(){
     var x;
     function sub2(){
          …
     }
}
function sub3(){
     …
}
```

Assume that the execution of this program is in the following unit order:
main calls **sub1**
sub1 calls **sub2**
sub2 calls **sub3**

a. Assuming static scoping, in the following, which declaration of x is the correct one for a reference to x?
   i. sub1
   ii. sub2
   iii. sub3
b. Repeat part a, but assume dynamic scoping.

Answer:

a. Static scoping:
   i. sub1: sub1
   ii. sub2: sub1
   iii. sub3: main

b. Dynamic Scoping:
   i. sub1: sub1
   ii. sub2: sub1
   iii. sub3: sub1

9) Consider the following Python program:

```
x = 1
y = 3
z = 5
def sub1():
     a = 7
     y = 9
     z = 11
     …
def sub2():
     global x
     a = 13
     x = 15
     w = 17
     …
     def sub3():
          nonlocal a
          a = 19
          b = 21
          z = 23
          …
…
```

List all the variables, along with the program units where the are declared, that are visible in the bodies of **sub1**, **sub2**, and **sub3**, assuming static scoping is used.
Answer:

	main	sub1	sub2	sub3
sub1	x = 1	a = 7, y = 9, z = 11	–	–
sub2	x = 15, y = 3, z = 5	–	a = 13 then 19 if sub3 is called, w = 17	–
sub3	x = 15, y = 3	–	a = 19, w = 17	b = 21, z = 23

10) Consider the following C program:

```
void fun(void){
     int a, b, c;  /* definition 1 */
     …
     while(…){
          int b, c, d;  /* definition 2 */
          …    ← 1
          while(…){
               int c, d, e;  /* definition 3 */
               …    ← 2
          }
          …    ← 3
     }
     …    ← 4
}
```

For each of the four marked points in this function, list each visible variable, along with the number of the definition statement that defines it.

Answer:

	definition 1	definition 2	definition 3
1	a	b, c, d	_
2	a	b	c, d, e
3	a	b, c, d	_
4	a, b, c	_	_

12) Consider the following program, written in JavaScript-like syntax:

```
// main program
var x, y, z;
function sub1(){
     var a, y, z;
     …
}
function sub2(){
     var a, b, z;
     …
}
function sub3(){
     var a, x, w;
     …
}
```

Given the following calling sequences and assuming dynamic scoping is used, what variables are visible during execution of the last subprogram activated? Include with each visible variable the name of the unit where it is declared.

a. **main** calls **sub1**; **sub1** calls **sub2**; **sub2** calls **sub3**.
b. **main** calls **sub1**; **sub1** calls **sub3**.
c. **main** calls **sub2**; **sub2** calls **sub3**; **sub3** calls **sub1**.
d. **main** calls **sub3**; **sub3** calls **sub1**.
e. **main** calls **sub1**; **sub1** calls **sub3**; **sub3** calls **sub2**.
f. **main** calls **sub3**; **sub3** calls **sub2**; **sub2** calls **sub1**.

Answer:

	sub1	sub2	sub3
a) main -> sub1 -> sub2 -> sub3	y	b, z	a, x, w
b) main -> sub1 -> sub3	y, z	_	a, x, w
c) main -> sub2 -> sub3 -> sub1	a, y, z	b	x, w
d) main -> sub3 -> sub1	a, y, z	_	x, w
e) main -> sub1 -> sub3 -> sub2	y	a, b, z	x, w
f) main -> sub3 -> sub2 -> sub1	a, y, z	b	x, w

Assignment:

- Problems 7, 8, 11 from the problem set questions in the textbook.