COMPUTER SCIENCE
AND SOFTWARE ENGINEERING
AUBURN UNIVERSITY
SAMUEL GINN COLLEGE OF ENGINEERING

Homework 3
COMP 3220

Due: September 14th, Sunday by 11:59PM (midnight) Please submit as a PDF or WORD document using Canvas

(1. 10pts) Question 5.4 (page 236) from the textbook.
Dynamic type binding is closely related to implicit heap-dynamic variables. Explain this relationship.

Implicit heap-dynamic variables only acquire their respective types when assigned values at runtime. Because of this, these variables are always dynamically bound to types.

(2. 20pts) Question 5.6 (page 236) from the textbook.
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 – **sub1**
   ii. sub2 – **sub1**
   iii. sub3 – **main**
b. Repeat part a, but assume dynamic scoping.
   i. sub1 – **sub1**
   ii. sub2 – **sub1**
   iii. sub3 – **sub1**

(3. 20pts) Question 5.9 (page 238) from the textbook.

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 they are declared, that are visible in the bodies of sub1, sub2, and sub3, assuming static scoping is used.

Sub1	**Sub2**	**Sub3**
a – sub1	a – sub2	a – sub3
x – main	w – sub2	b – sub3
y – sub1	x – sub2	w – sub2
z – sub1	y – main	x – sub2
	z – main	y – main
		z – sub3

(4. 20pts) Question 5.10 (page 238) from the textbook.

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.

1. a – 1, b – 2, c – 2, d – 2
2. a – 1, b – 2, c – 3, d – 3, e – 3
3. a – 1, b – 2, c – 2, d – 2
4. a – 1, b – 1, c – 1

(5. 20pts) Question 5.12 (page 240) from the textbook.

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 that 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.

	Variable	Declared
(a)	a, x, w	Sub3
	b, z	Sub2
	y	Sub1
(b)	a, x, w	Sub3
	y, z	Sub1
(c)	a, y, z	Sub1
	x, w	Sub3
	b	Sub2
(d)	a, y, z	Sub1
	x, w	Sub3
(e)	a, b, z	Sub2
	x, w	Sub3
	y	Sub1
(f)	a, y, z	Sub1
	b	Sub2
	x, w	Sub3

(6. 10pts) Some programming languages are typeless. What are the obvious advantages and disadvantages of having no types in a language?

a. Advantages:
    i. It allows for faster programming with much greater flexibility.
b. Disadvantages:
    i. The compiler or interpreter is in control of the data and variables and if variables are wrongfully assigned, the compiler can't catch any of the mistakes; so the programmer must ensure that the expressions and assignments are correct.
    ii. Supporting programs in a typeless language is significantly more difficult that in a strongly typed one because it is more difficult to understand what the original programmer wanted to accomplish with the given code.