Joseph Camacho-Terrazas
9/29/20
Chapter 5 Problem Set

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

Implicit heap-dynamic variables are bound to a type at runtime. Dynamic type binding is the act of binding a type to a variable at runtime. Therefore, the relationship between the two is that implicit heap-dynamic variables use dynamic type binding.

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

Joseph Camacho-Terrazas
9/29/20
Chapter 5 Problem Set
9:

Variable	**Declared**
sub1:	
a=7	sub1
y=9	sub1
z=11	sub1
x=1	main
sub2:	
a=13	sub2
x=15	sub2
w=17	sub2
y=3	main
z=5	main
sub3:	
a=19	sub3
b=21	sub3
z=23	sub3
w=17	sub2
x=15	sub2
y=3	main

Joseph Camacho-Terrazas
9/29/20
Chapter 5 Problem Set

11. Consider the following skeletal C program:

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

Names, Bindings, and Scopes

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`.

Joseph Camacho-Terrazas
9/29/20
Chapter 5 Problem Set

Variables Visible	**Defined**
a:	
a	main
b	fun1
c	fun2
d	fun3
e	fun3
f	fun3
b:	
a	main
b	fun1
c	fun1
d	fun3
e	fun3
f	fun3
c:	
a	main
b	fun1
c	fun1
d	fun1
e	fun3
f	fun3
d:	
a	main
b	fun1
c	fun1
d	fun1
e	fun3
f	fun3
e:	
a	main
b	fun1
f	fun3
c	fun2
d	fun2
e	fun2
f:	
a	main
b	fun1
c	fun2
d	fun2
e	fun2
f	fun3

Joseph Camacho-Terrazas
9/29/20
Chapter 5 Problem Set