CMPSC 461: Programming Language Concepts
Assignment 4 Solution

Problem 1 [4pt] Give an example in a programming language that you're familiar with in which a variable is alive but not in scope.

Solution:
In C, we can allocate a heap space for a local variable in a function without deallocating that heap space. After the function exists, the variable is no longer in scope, but it is alive.

Problem 2 [8pt] Consider the following class instances in a C++ program:

```
1  static myClass A;
2  int main()
3  {
4    myClass* B = new myClass();
5    foo();
6    delete B;
7    return 0;
8  }
9
10 void foo()
11 {
12   myClass* C = new myClass();
13   myClass D;
14 }
```

a) (4pt) What is the storage allocation (static/stack/heap) for the objects associated with A, B, C and D?

Solution:
A is static allocated, B and C are heap allocated, D is stack allocated.

b) (4pt) Consider one execution of the program above. The execution trace, a sequence of program statements executed at run time, of this program is
`4 5 12 13 6 7`
For each object associated with A, B, C, and D, write down its lifetime (use a subset of execution trace, e.g., `12 13` to represent the lifetime).

Solution:
Lifetime of A: `4 5 12 13 6 7`
Lifetime of B: `4 5 12 13 6` or `4 5 12 13`
Lifetime of C: `12 13 6 7`
Lifetime of D: `13`

Problem 3 [12pt] Consider the following pseudo-code. Assume the language has one global scope, and one scope for each braced code block (including function and branch):

```
int x = 3;

int f1(int x) {
    if (x > 4)
        f1(x-1);
    else {
        int x = 1;
        f2();
    }
}

int f2() {
    print x;
}
f1(5);
```

a) (4pt) Draw the symbol tables in separation for all 4 scopes in this program. Assume each table contains two columns: name and kind.

Solution:
Global scope (Table A):

Name	Kind
x	id
f1	fun
f2	fun

Function f1 (Table B):

Name	Kind
x	para

Nested scope in f1 (Table C):

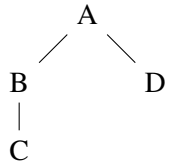
Name	Kind
x	id

Function f2 (Table D):

Name	Kind

b) (4pt) If the language uses static scoping rules, what's the expected output from the print statement? Justify your answer by showing the tree of the symbol tables at the print statement.

Solution:
3. The hierarchy of tables is as follows:



c) (4pt) If the language uses dynamic scoping rules, what's the expected output from the print statement? Justify your answer as you did in Problem 3 b).

Solution:
1. The hierarchy of tables is as follows:

A
|
B
|
B
|
C
|
D

Problem 4 [14pt] Consider the following Scheme program.

```
(define A
  (let* ((x 2)
         (C (lambda (P)
              (let ((x 4))
                 (P))))
         (D (display x))
         (B (let ((x 3))
                 (C D))))
    (B)))
```

a) (7pt) What would the program print (the "display" function) if Scheme used dynamic scoping and shallow binding? Justify your answer by showing the tree of symbol tables when execution reaches the display expression.

Solution:
Function A (Table A):

Name	Kind
x	id
C	fun
D	fun
B	fun

Function C (Table C):

Name	Kind
P	para
x	id

Function D (Table D):

Name	Kind

Function B (Table B):

Name	Kind
x	id

The hierarchy when the print statement is executed:
A
|
B
|
C
|
D
So the program will print out the "x" in function C, which is 4.

b) (7pt) What would the program print if Scheme used dynamic scoping and deep binding? Justify your answer by showing the tree of symbol tables when execution reaches the display expression.

The hierarchy when the reference of D is created:
A
|
B
|
D
So the program will print out the "x" in function B, which is 3.

Problem 5 [12pt] Consider the following pseudo-code, assuming nested subroutines and static scope:

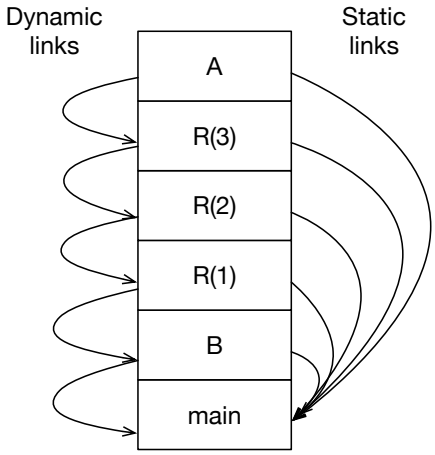
```
1  main() {
2    int g;
3    int x;
4    function B(int a) {
5      x = a + 5;
6      R(1);
7    }
8    function A(int n) {
9      g := n;
10   }
11   function R(int m) {
12     print x;
13     x = x / 2;
14     if (x > 1)
15       R(m + 1);
16     else
17       A(m);
18   }
19   // body of main
20   B(3);
21   print g;
```

1. (3pt) What does the program print?

   Solution:
   8 4 2 3

2. (6pt) Draw a picture of the run-time stack when A has just been called. For each frame, show the static and dynamic links. You do not have to show the storage for any other information.

   Solution:



3. (3pt) Refer to the run-time stack, briefly explain how function A finds variable g.

   Solution: It uses the static links to locate the frame of Main. Within that frame, it finds g which is at a statically known offset.