Gentry Atkinson
CS5318: Spring 2019
Assignment 4
Due: 30 April 2019

1. Supposed that the following function declarations (in C++ syntax) are available in a program:
    - int pow(int, int);
    - double pow(int, double);
    - double pow(double, double);

    and suppose that the following code calls the pow function:

    int x;
    double y;
    x = pow(2, 3);
    C: 1
    Java: 1
    Ada: 1
    y = pow(2, 3);
    C: 1
    Java: 1
    Ada: illegal
    x = pow(2, 3.2);
    C: 2
    Java: 2
    Ada: illegal
    y = pow(2, 3.2);
    C: 2
    Java: 2
    Ada: 2
    x = pow(2.1, 3);
    C: 3
    Java: 3
    Ada: illegal
    y = pow(2.1, 3);
    C: 3
    Java: 3
    Ada: illegal
    x = pow(2.1, 3.2);
    C: 3
    Java: 3

Ada: illegal

y = pow(2.1, 3.2);

C: 3

Java: 3

Ada: 3

Given the languages (a) C++, (b) Java, and (c) Ada, write down the number of the pow function called in each of the eight calls, or write "illegal" if a call cannot be resolved in the language, or if a data type conversion cannot be made.

2. Assume x is an int variable and y is an int * variable.
    1. Which of the following C expressions are l-values? Which are not? Why?

(1) x + 2	**not an lvalue**
(2) &x	**not an lvalue**
(3) *&x	**is an lvalue**
(4) &x + 2	**not an lvalue**
(5) *(&x + 2)	**is an lvalue**
(6) &*y	**not an lvalue**

    2. Is it possible for a C expression to be an l-value but not an r-value? Explain.

        Any expression that includes a type declaration such as "int x" can only be an lvalue, never an rvalue. For instance:

        int x = y is legal but

        y = int x is illegal.

        Therefore it is possible for a C expression to be an l-value but not an r-value.

3. Given the C declarations:

```
struct {int i; double j;} x, y;
struct {int i; double j;} z;
```

    1. The assignment x = z generates a compilation error, but the assignment x = y does not. Why? **X and Y are of the same type so the assignment is valid. C uses name equivalence for structs which does not exist between the two anonymous structs so the compiler sees x and z as different types. Without explicit rules for casting the assignment x = z illegal.**
    2. Give two different ways to fix the code so that x = z works.

        1) struct {int i; double j;} x, y, z;

```
x=z;

2) struct {int i; double j;} x, y;

struct {int i; double j;} z;
x.i = z.i;

x.j = z.j;
```

4. Give the output of the following program (written in C syntax) using the four parameter passing methods: Pass by Value, Pass by Reference, Pass by Value-Result, and Pass by Name.

```
int i;
int a[2];

void p(int x, int y)
{  x++;
   i++;
   y++;
}

main()
{ a[0] = 1;
  a[1] = 1;
  i = 0;
  p(a[i], a[i]);
  printf("%d %d\n", a[0], a[1]);
  return 0;
}
```

Pass by Value: 1 1
Pass by Reference: 3 1
Pass by Value-Result: 2 2
Pass by Name: 1 2

5. The following Ada program contains a function parameter.

```
with Text_IO; use Text_IO;
with Ada.Integer_Text_IO; use Ada.Integer_Text_IO;

Procedure params is
  procedure q is
    type intFunc is access function (n: integer) return integer;
    m: integer := 0;

    function f (n: integer) return integer is
    begin
     return m + n;
    end f;
    procedure p (g: intFunc) is
     m: integer := 3;
    begin
     put(g(2)); new_line;
    end p;

  begin
    p(f'access);
  end q;

begin
  q;
end params;
```

1. Draw the stack of activation records after the call to g in p.

m	**Activation of q**
m	**Activation of p**
g	
n	**Activation of f**
m	
m	**Activation of p**
g	

2. What does the program print and why?

   2 because f is called from inside of q so it uses q's local value of m which is 0. f returns m + n = 0 + 2 = 2. p prints the two and the a newline.