Julian Garcia
Hw1 CSE 324

Exercise 2-15: Consider the following FORTRAN SUBROUTINE:

```
SUBROUTINE TEST (X, Y, Z)
X = 1
Z = X + Y
RETURN
END
```

and consider the following code fragment:

```
N = 2
CALL TEST (N, N, M)
```

What will be the final value of M if the parameters are passed by reference? What will it be if they are passed by value-result?

Answer:
If the parameters are passed by reference, the final value of M will be 2, if they are passed by value-result, the final value of M will be 3.

The reason the by reference version returns a value of 2 is because of the X = 1 statement. Since X and Y are both pointing to n at the call, when X is set equal to one N is also set equal to one, which also sets Y equal to the value of 1 since its value is dependent on N. Therefore, Z is equal to the addition of 1 + 1 = 2 and so M is equal to 2.

The by value-result version returns a value of three because the statement X = 1 doesn't change the value of N unlike the by reference version. Therefore, Y retains the value of 2 and Z is equal to the addition of 1 + 2 = 3 so M is equal to 3.

Exercise 2-16*: There are several different varieties of pass by value-result: The address of the actual can be computed once, at subprogram entry time, or twice, once on entry and once on exit. Describe the output of this program under each of these two varieties of value-result:

```
DIMENSION A(2)
I = 1
A(1) = 10
A(2) = 11
CALL SUB (I, A(I))
PRINT, A(1), A(2)
END
```

```
SUBROUTINE SUB (K, X)
PRINT, X
K = 2
X = 20
RETURN
END
```

Does the outcome depend on the order in which the results are copied from the formals back into the actuals?

Julian Garcia
Hw1 CSE 324

Answer:

1. If the addresses are computed once, on entry at the caller, then at the caller side we have:
   Call SUB (I, A(I)) and the value of the actual parameters: I is 1 and A(1) = 10
   Then inside the callee SUB we have the value K = 1, and X = 10, SUB prints X so 10 is printed.
   Then K is set to 2 and X is set to 20 and the program returns.
   Now the value of X is given to A(1), and so A(1) is now equal to 20.
   The actual I is also given the value of 2 on the caller side.
   So, when the program Prints A(1) and A(2), it prints 20 and 11, since A(2) was never modified.

   So, ultimately this version of the program prints 10, 20, 11.

2. Now, when the addresses are computed twice, the outcome does depend on the order in which the results are copied from the formals back to the actuals. And there are two different results you can get from the order in which they're copied.

   a. If the addresses are computed twice, once on entry and once on exit, and the value of the formal K is stored is placed back into the actual I before re-evaluating the value of A(I), then the program prints 10 10 20.
      Program walkthrough:
      First, we still have Call SUB(I, A(I)) at the caller side with actual parameters I = 1 and A(1) = 10.
      X is still printed as 10 like in the previous version, and K is set to 2 then X is set to 20.
      Now, on the return the actual I is given the value 2, which turns A(I) into A(2), so A(2) is given the value of 20 while A(1) remains as 10.
      So, Print A(1), A(2) will now print 10 20.

   b. If the addresses are computed twice, once on entry and once on exit, and the value of the formal K is stored is placed back into the actual I after re-evaluating the value of A(I), then the program prints 10 20 11 like in the first method.
      Program Walkthrough:
      First, we still have Call SUB(I, A(I)) at the caller side with actual parameters I = 1 and A(1) = 10.
      X is still printed as 10 like in the previous version, and K is set to 2 then X is set to 20.
      Now, A(I) is evaluated before K is stored back into I, so A(I) is given the value of 20 while A(2) remains 11.

Now the actual I is given the value 2, which turns A(I) into A(2), but A(I) has already been evaluated so there's no other change.
So, with A(1) = 20 and A(2) = 1, Print A(1), A(2) will now print 20 11.