# Programming Languages HW #5

Charlie Coleman

1. **C:** Print statements:

   7. x - line 5 def, y - line 3 def
   11. x - line 5 def, y - line 8 def
   13. x - line 2 def, y - line 3 def
   15. x - line 2 def, y - line 3 def

   Output:

```
6, 4
6, 7
2, 4
2, 4
```

   **C#:** Print statements:

   7. x - line 5 def, y - line 8 def
   11. x - line 5 def, y - line 8 def
   13. x - line 2 def, y - line 3 def
   15. x - line 2 def, y - line 3 def

   Output:

```
6, 7
6, 7
2, 4
2, 4
```

   **M-3:** Print statements:

   7. x - line 5 def, y - line 8 def
   11. x - line 5 def, y - line 8 def
   13. x - line 2 def, y - line 3 def
   15. x - line 2 def, y - line 3 def

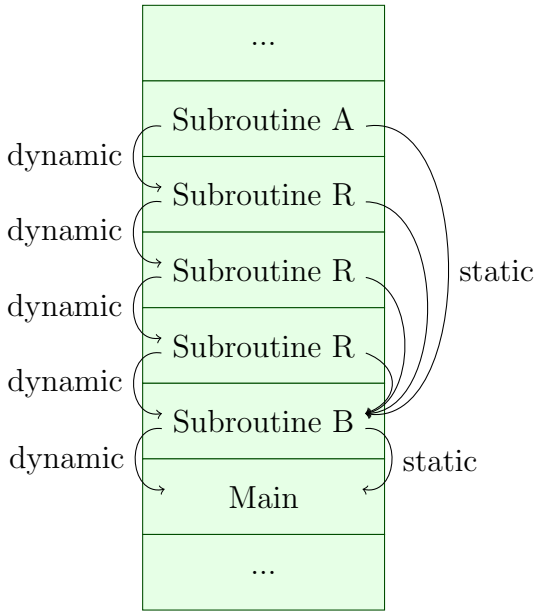
   Output:

```
9, 7
9, 7
2, 4
2, 4
```

2. (a) Output:

```
5
3
1
1
```

(b)




(c) A will follow the static links up the call chain to find the value of g

3. (a) Output:

```
0, 0
0, 0
0, 0
0, 0
0, 0
```

(b) Output:

```
11, 0
11, 7
1, 7
1, 18
1, 7
```

4. (a) 1
   (b) -3
   (c) -2

5. false, error. It does not make sense for a language to check both boolean expressions when either being false means the return is false. C checks the left-most expression on the and, if it is true it checks the other. The division by zero returns an error here.

6. Structural - all. Strict name - none. Loose name - A & B

7. C/Fortran: can result in simpler case statements when a default/others case is not a requirements. This can cause hard to diagnose errors if a case is supposed to get caught but isn't beign caught.

Pascal/Modula: What I would call the worst of both worlds. Only helpful when you know inputs won't be outside of the covered ranges. Does not warn you when a value is not being covered, but will throw an error when those values come up. Does help some with debugging values that should be covered but aren't.

Ada: requires more code in some cases as you basically must include an others case. Doesn't allow compilation of code missing cases, which leads to less dynamic bugs.

My preference: Ada. This is prevents errors and leads to more bug resistant code. Is used in some languages important to Computer Eng, like VHDL.

8. `buzz` may have overwritten some or all of the functions of `fizz`. If one of these functions is changed to return a different type, it could cause issues with the calling function, as it would expect the type that is defined in `fizz` but recieve the return type from `buzz`.