Programming Language Concepts Homework 5

Due Wednesday Oct 30; Joseph Sepich (jps6444)

1 Problem 1

1.1 Part a

```
int g(int x, int prev) {
  if (x == 1) {
    return prev;
  }
  return g(x - 1, prev * x);
}
```

1.2 Part b

```
int g(int x) {
  int product = x;
  for (int i = x - 1; i > 0; i = i-1) {
    product = product * i;
  }
  return product;
}
```

1.3 Part c

The "optimized" version is more efficient with space efficiency and time efficiency, because of how function calls affect the state of the stack. When using the recursive version, a new activation record is created and stored on the stack for each function call. For the given function this would store x activation records! This also requires the time it takes to create and destory these activation records, because when the activation records are created the parameters ands control must be passed and on destruction the return value and control must be passed. The optimized version realizes that the extra fucntion calls are useless, because you are doing the same thing over and over, so instead the values in the current activation records are updated iteratively and the "optimized" version therefore only uses one activation record.

2 Problem 2

On many systems when the variable/name i is declared and its address is assigned teh data stored at that address is a zero making integer variables initialized as 0. Since i is being initialzied to zero, it prints 0 through 9 every time it is run. On other systems that don't do this it may not be assigned this default value and error out.

3 Problem 3

3.1 Part 1: Call by Value

Run through line by line:

1. p = 2 (declaration)
2. q = 3 (declaration)
3. print f1(p, q)
4. b = 2 * c = 6
5. c = 5 + b = 11
6. prints b + c = **17** on return
7. print p, q prints **2, 3** since copies used in function f1
8. f1(p, q) doesn't print or update p or q
9. print f2(q)
10. returns f1(q, q)
11. b = 2 * c = 6
12. c = 5 + b = 11
13. prints b + c = **17** on return
14. print p, q prints **2, 3** since copies used in functions

Output is: **17, 2, 3, 17, 2, 3**

3.2 Part 2: Call by Reference

Run through line by line:

1. p = 2 (declaration)
2. q = 3 (declaration)
3. print f1(p, q)
4. p = 2 * q = 6
5. q = 5 + p = 11
6. prints p + q = **17** on return
7. print p, 1 prints **6, 11** since references used in function f1
8. f1(p, q)
9. p = 2 * q = 22
10. q = 5 + p = 27
11. f1 returns 22 + 27 = 49 but does not print
12. print f2(q)
13. returns f1(q, q)
14. q = 2 * q = 54
15. q = 5 + q = 59
16. prints q + q = **118** on return
17. print p, q prints **22, 59** since references used in functions

Output is: **17, 6, 11, 118, 22, 59**

3.3 Part 3: Call by Value-Return

Run through line by line:

1. p = 2 (declaration)
2. q = 3 (declaration)
3. print f1(p, q)
4. b = 2 * c = 6
5. c = 5 + b = 11
6. prints b + c = **17** on return
7. p = 6 on return
8. q = 11 on return
9. print p, q prints **6, 11** since values are updated on return of f1
10. f1(p, q)
11. b = 2 * c = 22
12. c = 5 + b = 27
13. p = 22, q = 27 on return
14. print f2(q)
15. returns f1(q, q)
16. b = 2 * c = 54
17. c = 5 + b = 59
18. prints b + c = **113** on return
19. q assigned to 59 on return
20. print p, q prints **22, 59** since values updated on return

Output is: **17, 6, 11, 113, 22, 59**

4 Problem 4

4.1 Part 1: A calling D

It is not possible for A to call D. The depth of A is < the depth of D, so D is not in the scope of A unless it is the direct parent of D, which we know the distance between the depth of A and D is greater than 1, so A does not define D (not its parent), so it cannot call D, since it is not in scope.

4.2 Part 2: E calling B

It is possible for E to call B. The depth of E is equal to the depth of B. If E called B then the static link would be found by getting the diff of depth(E) - depth(B) and hopping up E's static links to the symbol table of the funciton at that level. Then the static link of B would be assigned to the given function. In this case since they are at the same level, you would go up one hop to A, and B would have a static link to A, just like E.