CSci 4651 Fall 2003
Problem Set 6: Storage management.
Due Wednesday, November 5th

IMPORTANT, please read before you start working on the problems: This problem set uses Java syntax for convenience. However, this is NOT Java code. Read each problem carefully to understand which model you are working in (pass-by-value vs. pass-by-reference, static vs. dynamic scope rules, etc.). Assume that all code given below is syntactically correct and does not cause compilation error. Assume that `print(‘‘x = ’’ + x)` is a printing command which works like `System.out.println(‘‘x = ’’ + x)` in Java for all types of variables used in this problem set.

If you are confused about the syntax, please **ask** in class, during my office hours, or by e-mail.

Problem 1. Consider the following code (the lines are numbered for easy reference):

```
1.   int x = 3;
2.   {
3.       int y = 5 + x;
4.       {
5.           int x = 2 + y;
6.       }
7.       x = y;
8.   }
```

Question 1. Draw the program stack right after line 5 gets executed.
Question 2. What is the final value of x in this program fragment?
Question 3. Consider two variable declarations: the one on line 1 and the one on line 3. Which lines of the code constitute the scope of each of these declarations?
Question 4. Which lines constitute the lifetime of each of the declarations in Question 3?

Problem 2. Consider the call `f(2, -1)` to the following function:

```
int f  (int x, int y) {
    int z = 0;
    int w = x * x;
    if (y < 0) {
        int z = -w;
        print("z = " + z);
    }
    return z;
}
```

Question 1. Draw the program stack at the point right after the line
`int z = -w;`
is executed.

Question 2. What will be printed by the `print` statement? What will be the value returned by the function? Explain your answers using the stack diagram from Question 1.

Problem 3. Recall that Fibonacci numbers are defined the following way: $F_0 = 0, F_1 = 1, F_n = F_{n-1} + F_{n-2}$ for $n \geq 2$.

Question 1. Consider the following recursive function which computes the n-th Fibonacci number:

```
int fib (int n) {
    if (n == 0) return 0;
    if (n == 1) return 1;
    return fib(n-1) + fib(n - 2);
}
```

Suppose a program computes `fib(3)`. The program will start the computation by pushing the activation record for `fib(3)` on the stack. Then `fib(2)` will be called from `fib(3)`, and its activation record will be pushed on the stack:

```
push fib(3)
push fib(2)
...
```

Continue the sequence until the return of `fib(3)` (so that the last stack operation is `pop fib(3)`).

How many activation records are generated for this computation? What is the maximum number of activation records for `fib` residing on the stack simultaneously during this computation?

Question 2. Fill in the missing code for a tail-recursive version of `fib` below, assuming that `fibTail(3, 0, 1)` computes F_3.

```
int fibTail (int n, int f1, int f2) {
    if (n == 0) return f1;
    return fibTail(....);
}
```

Note: you may test your function using Java (the above code is syntactically correct, provided the missing parameters are specified in a syntactically-correct manner).

Question 3. Assuming that **no optimization** has been performed, write down the sequence of push/pop operations on the program stack for the tail-recursive function in Question 2 in computation of F_3. How many activation records were generated in this computation? How many of them (maximum) resided on the stack simultaneously?

Question 4 (Extra credit). If the optimization has been performed to turn the tail-recursive function in Question 3 into a while loop, what is the loop? Use the Java syntax for your answer.

Problem 4. Consider the following function:

```
int f(int n, int m) {
    n = n + 1;
    m = m + 1;
    return n + m;
}
```

Question 1. Suppose that the function is called in the following fragment of code:

```
    int x = 2;
    int y = 3;
    print(f(x,y));
    print(x);
    print(y);
```

What would be the printed by the code fragment in each of the following cases:

1. Both function parameters are passed by value.
2. Both function parameters are passed by reference.
3. The first function parameter is passed by value, and the second one is passed by reference.

Question 2. Now consider the following fragment of code:

```
    int x = 2;
    print(f(x,x));
    print(x);
```

What would be the printed by the code fragment in each of the following cases:

1. Both function parameters are passed by value.
2. Both function parameters are passed by reference.
3. The first function parameter is passed by value, and the second one is passed by reference.
4. The first function parameter is passed by reference, and the second one is passed by value.

Problem 5. Consider the following program, where `main` is the first function in program execution. Parameters are passed by value.

```
int x = 2;
int y = 3;

void f(int n) {
    x = x + n;
    y = y - n;
}

void main () {
    int x = 1;
    f(1);
    {
        int y = 5;
        f(x);
    }
}
```

Question 1. Assuming dynamic scope rules, draw the program stack at two points in execution: right before the call `f(1)` returns and right before the call `f(x)` returns. Show values of all variables in the stack pictures.

Note that for the second function call the stack will have both kinds of blocks: in-line blocks and those associated with a function.

Question 2. What are the final values of **global** `x` and `y` in the case of dynamic scope rules? Use the stack pictures to explain your answer.

Questions 3 and 4. The same as 1 and 2, but for static scope rules.

Problem 6. Consider the following program where `main` is the first function in program execution. Assume the static scope rules. `int -> int` is the type of functions from an integer to another integer.

```
int x = 0;

int f(int a) {
    if (a == 1) return x;
    else {
        x = x  + a;
        return f(a - 1);
    }
}

void g (int -> int h) {
    int x = 5;
    print(h(2));
}

void main () {
    g(f);
}
```

Question 1. Note that `f` is recursive. Draw the program stack and function closures when all activation records for `f` needed for this program are pushed on the stack.

Question 2. What is going to be printed in the program? Use the stack picture to explain your answer.

Problem 7. Assume the static scope rules and notations as in the previous problem. Consider the following code fragment, where `f` returns a function g:

```
1.   int-> int f () {
2.       int x = 0;
3.       return  ( void g() { x = x + 1; } );
4.   }
5.   void main () {
6.       int -> int h = f();
7.       h();
8.       int -> int j = f();
9.       j();
10. }
```

Draw the program stack and all the function closures at the following points of the program execution:

1. Right before line 7 is executed.

2. Right after line 7 is executed.
3. Right after line 8 is executed.
4. Right after line 9 is executed.

Show the values of all variables.