Homework 5
Due April 18, 2016

Submissions are due by the beginning of class on the specified due date. Handwritten or typed solutions are acceptable. If you do write your solutions by hand, be sure to write clearly. If the grader cannot read your answer, they cannot give you the points. Late submissions are accepted up to 48 hours past the deadline with a 10% penalty for each day.

1. (20 points) Explain the difference between unstructured and structured control flow. For at least two common structured control flow mechanisms (such as while loops, if-statements, etc...), briefly explain how they replace the use of goto statements in a programming language.

2. (16 points) Explain the difference between enumeration-controlled loops and logically controlled loops. For a language like C in which enumeration-controlled loops are not available, explain (with a sentence or two) and write a small code example showing how they may be emulated.

3. (16 points) What is tail-recursion and why is it desirable over naive recursion? Is it possible to write a fully tail-recursive version of the classic Quicksort algorithm? Hint: if you’re going to look this one up online, think carefully about the answer you submit.

4. (16 points) Consider the following C-style program below. Diagram the changes in the stack that occur when some subroutine bar is called from the current subroutine foo. In particular, show the subroutine frames on the stack before the call of bar, while bar is executing, and after it has returned. Show the positions of the stack pointer (sp) and the frame pointer (fp) at each stage. Also, using the typical subroutine frame layout discussed in class, indicate the stack locations of the arguments to bar as well as the return value.

```c
int bar(int a, int b){
    int ret;
    ret = a+b;
    return ret;
}

void foo(){
    int x;
    x = bar(1,2);
}
```
5. (16 points) Assume we have the following program for some language with C-like syntax.

\begin{verbatim}
int x, y, z;

void foo(a, b, c) {
    a = a*2;
    b = b*2;
    x++;
    c = c+b;
}

x = 1; y = 2; z = 3;
foo(x, y, z);
\end{verbatim}

What are the values of x, y, and z after the call to foo for each of the following cases?

a) The formal parameters a, b, and c are all call-by-value?

b) The formal parameters a, b, and c are all call-by-reference?

c) The formal parameters a, b, and c are all call-by-value/result (i.e. in/out mode in Ada)?

d) The formal parameters a, b, and c are all call-by-name?

6. (16 points) Using the same definition for foo that we used in question 5, assume we make the following call.

\begin{verbatim}
x = 1;
foo(x, x, x+1);
\end{verbatim}

What is the value of x after the call to foo for each of the following cases?

a) The formal parameters a, b, and c are all call-by-value?

b) The formal parameters a, b, and c are all call-by-reference?

c) The formal parameters a, b, and c are all call-by-value/result (i.e. in/out mode in Ada)?

d) The formal parameters a, b, and c are all call-by-name?