COMS W4115
Programming Languages and Translators
Homework Assignment 3

Prof. Stephen A. Edwards
Columbia University

Due Monday, December 8th, 2015
at 4:10 PM

Submit solution on paper (no email). Please write your name clearly on the paper.

Do this assignment alone. You may consult the instructor and the TAs, but not other students.

1. (20 pts.) For the following C array,

   ```
   int a[3][2];
   ```

   assume you are working with a 32-bit little-endian processor with the usual alignment rules (e.g., a Pentium).

   (a) Show how its elements are laid out in memory.

   (b) Write an expression for the (byte) address of `a[i][j]` in terms of *a*, *i*, and *j*.

   (c) Verify parts a) and b) by writing a small C program that allows you to **test your hypothesis**. Examine the assembly language output with the C compiler's -S flag (e.g., `gcc -O -S array.c`). Such a program should be simple and contain and access such an array, but not be so simple that the compiler can optimize most of it away. **Turn in an annotated assembly listing** that explains how it verifies your hypothesis. Make sure the assembly listing is no more than about 40 lines, either by simplifying your program or trimming the output.

2. (20 pts.) For a 32-bit little-endian processor with the usual alignment rules, show the **memory layout** and **size in bytes** of the following three C variables.

   ```
   union {
     struct {
       int a;   /* 32-bit */
       short b; /* 16-bit */
     } s;
     int c;    /* 32-bit */
   } u1;
   ```

   ```
   struct {
     char a;
     short b;
     int c;
     short d;
   } s1;
   ```

   ```
   struct {
     char a;
     short b;
     char c;
     short d;
     short e;
   } s2;
   ```

3. (20 pts.) Draw the layout of the stack just before *bar* is called in *foo*. Indicate storage for function arguments, local variables, return addresses, and stored frame pointers. Indicate where the stack and frame pointers point.

   ```
   void bar(int x, int y);

   void foo(int a, int b)
   {
     int c, d, e;
     bar(4, 2);
   }
   ```

4. (20 pts.) Draw the layouts of a Circle and a Rectangle object as well as the virtual tables for their classes. Indicate how the runtime decides to call the appropriate *area* function for *s1* in *main*.

   ```
   public class Shape {
      public double area() { ... }
   }

   class Circle extends Shape {
      private double diameter;
      public double area() { ... }
   }

   class Rectangle extends Shape {
      private double height, width;
      public double area() { ... }
   }

   public class Main {
     public static void main() {
       Shape s1 = new Rectangle(35, 42);
       System.out.println( s1.area() );
     }
   }
   ```

5. (20 pts.) Show the steps in deriving the normal form, if any, of each of the following Lambda terms. Assume + is a function that takes a numeric argument and returns a function that adds that argument to its argument.

   (a) $(\lambda x\,.\, +\, x\; 5)\; 3$

   (b) $(\lambda x\,.\, x)\; (\lambda x\,.\, x)\; (\lambda x\,.\, x)$

   (c) $(\lambda z\,.\, z\; z)\; (\lambda z\,.\, z\; z)$

   (d) $(\lambda x\,.\, \lambda y\,.\, +y\; 3)\; ((\lambda z\,.\, z\; z)\; (\lambda z\,.\, z\; z))\; ((\lambda x\,.\, +\, x\; 3)\; 36)$