Section Handout

Problem 1: The Incredibles (given on a midterm a few years ago)

Consider the following **`struct`** definition:

```
   typedef struct {
      int violet;
      char *dash[2];
      char superboy[4];
   } superhero;

   static superhero **pixar(superhero *syndrome);
   static superhero *theincredibles(short *frozone, superhero elastigirl)
   {
line 1    frozone += elastigirl.superboy[*frozone];
line 2    ((superhero *)((superhero *) elastigirl.dash[0])->dash)->violet = 400;
line 3    return *pixar(&elastigirl) + 10;
   }
```

Generate code for the entire **`theincredibles`** function. Be clear about what assembly code corresponds to what line.

Problem 2: New Zoo Revue (given on a midterm several years ago)

You are to generate code for the following nonsense code. Don't be concerned about optimizing your instructions or conserving registers. We don't ask that you draw the activation records, but it can only help you get the correct answer if you do. Be clear about what assembly code corresponds to each line of C.

```
struct human {
   int doug;
   short emmyjo[2];
};

struct other {
   char *freddie;
   struct human charlie;
   struct human *henrietta;
};

static struct human **AskingQuestions(struct human *heroes);
static struct human **BeingCalm(short *conformity, struct other *gooddeeds)
{
   conformity[*conformity] = 0;
   gooddeeds += ((struct human *) (gooddeeds->henrietta[0].emmyjo))->doug;
   return AskingQuestions((struct human *) &gooddeeds);
}
```

- Generate code for the entire **`BeingCalm`** function.

(over)

- The assembly code generated by the compilation of **`AskingQuestions`** is given below. Provide a C-code implementation of **`AskingQuestions`**, given the constraint that all local variables must be of type **`struct human`** and/or **`struct human *`**, and the only typecast allowed is **`(struct other *)`**.

```
SP = SP - 12;
R1 =.2 M[SP + 8];
M[SP + 4] = R1;
R2 = M[SP + 16];
BNE R2, 0, PC + 16;
R3 = M[SP];
R4 = R3 + 8;
M[SP] = R4;
RV = SP + 12;
SP = SP + 12;
RET;
```