1. **True/False.** Circle the appropriate choice (there are no trick questions).

(a) T F In C, a pointer variable is used for storing an address.

(b) T F In x86 assembly, the eax register can be used to hold integers but not addresses.

(c) T F In C, if \( x \) is an integer variable, the expression \( x << 3 \) computes \( x \times 8 \) but does not change the value of \( x \).

(d) T F If, in C, an array is declared by \( \text{int a[10]}; \), then accessing \( a[11] \) during execution will generate an error message.

(e) T F A callee-saved register should be saved before it is written to in a procedure.

(f) T F A mask is a value used to modify or examine individual bits of a byte or word.

(g) T F The bitwise \( \text{xor} \) operation (\( ^ \) in C) can be used to flip the bits of a word.

(h) T F The bitwise \( \text{and} \) operation (\( \& \) in C) can be used to flip the bits of a word.

(i) T F Compiling C into x86 assembly comprises translating each C statement into a single assembly instruction.

(j) T F In x86 assembly, a label corresponds to an address.

2. **Answer this question on this sheet**

Consider the following x86 code fragment for computing the sum of an array of 10 32-bit integers, pointed to by ecx.

```
#Intel Syntax   #AT&T Syntax
mov eax,0       mov $0,%eax  #
mov edx,0       mov $0,%edx  #
TOP:
cmp edx,10      cmp $10,%edx #
    jl OUT       jl OUT   #
    add eax,[ecx+edx] add (%ecx,%edx,)%eax #
    inc edx      inc %edx   #
    jmp TOP      jmp TOP   #
OUT:
```

(a) There are two bugs in the code. What are they?

(b) Put a comment following each “#” in the above code to describe what that instruction does (or is supposed to do, in the case of a bug).

Please turn this page over.
3. **Put your answer in the blue book.**
   Consider the following C procedure, `foo`.

   ```c
   int foo(int x)
   {
       int i;
       int y = 0;
       for(i=0;i<32;i++) {
           y = y + (x & 0x1);
           x = x >> 1;
       }
       return y;
   }
   ```

   (a) What does `foo` do? That is, what is the relationship between the value that it returns and the value passed into it?

   (b) Write an x86 assembly procedure that computes the same thing that the above C procedure does. Be sure that your assembly procedure could be called from a C procedure.

4. **Put your answer in the blue book.**

   (a) Define a C struct type `CELL` that contains the following fields: an integer `x`, a string `y`, and a `next` field that points to another structure of type `CELL`.

   (b) Write in C a procedure corresponding to the declaration
   ```c
   int list_length(CELL *head);
   ```
   that returns the length of a linked list whose first element is pointed to by `head`.

5. **Put your answer in the blue book.**

   A simple C procedure for computing the integer square root of a number (i.e. the greatest integer less than or equal to the square root of the number) is as follows:

   ```c
   int square_rt(int x, int low, int high)
   {
       if (low >= high - 1)
           return low;
       int mid = (low + high) >> 1;
       if ((mid * mid) > x)
           return square_rt(x,low,mid);
       else
           return square_rt(x,mid,high);
   }
   ```

   where the initial call to `square_rt` would be `square_rt(n,1,n)`. Translate `square_rt` into x86 assembly, so that it could be called from C.