Without

Nodes

```
struct Node {
      void *data;
      struct Node *left, *right;
      };

/* do the entire function in assembler */
/* from the label to the return */
long count(struct Node *ptr)
{
      long rval = 0;

      if(ptr)
      {
          rval = 1;    /* count the node itself */
          /* add the child counts */
          rval += count(ptr->left);
          rval += count(ptr->right);
      }
      return rval;
}
```

P7

```
long fx( long p1, long p2, long p3,
      long p4, long p5, long p6,
      long p7)
{
      long rval;

      /* render only the following line of C in assembler */
      rval = p1 + p7;

      return rval;
}
```

Ptr

```
/* be able to get into and out of memory with a pointer */

void inc(int *ip)
{

      /* think this one over carefully, it hides a bunch of stuff */
      *ip += 1;

}
```

Structs

```
/* given a struct, get any elelemnt or the address of any element
** THis means you have to know how structs are laid out, alignment,
** padding etc. */

struct Record {
    char name[15];
    short scores[2][6];
    int win, loss, tie;
};

void fx( struct Record *ptr)
{

      int i, *ip;
      short *sp;

      /* be able to do any of the next 4 lines */

      i = ptr->win;
      ip = &ptr->tie;

      i = ptr->scores[0][3];
      sp = &ptr->scores[1][1];

}
```

While

```
void wtest( char *bytePointer, long count)
{
      /* render the entire while loop in assembler */
      while(count)
      {
            count--;
            bytePointer[count] = 0;
      }

      /* but only the while loop above here */

}
```