CSE/EE 373 Autumn 1994
Data Structures and Algorithms
Assignment # 3 Common Mistake Solutions

1. Write both recursive and non-recursive pseudo-code procedures to reverse a singly-linked list in O(n) time. You need not preserve the original list – that is, each procedure should transform the list into its reverse.

   Answer:

```
struct list_typ
{

    element_typ key;
    struct list_typ *next;

}
```

   a) non-recursive

```
Reverse(L)
struct list_ptr *L;
{
struct list_ptr *p;
struct list_ptr *tmp;
struct list_ptr *result;

result = nil;
while(L != NULL)
{

    p = L;
    tmp = (struct list_typ *)malloc(sizeof(struct list_typ));
    tmp -> key = p -> key;
    tmp -> next = result;
    result = tmp;
    p = p -> next;

}

return(result);

}
```

b) recursive

```
Reverse(L, R)
struct list_ptr *L;
struct list_ptr *R;
{
struct list_ptr *p;
struct list_ptr *tmp;

if (L != NULL)
{
    tmp = (struct list_typ *)malloc(sizeof(struct list_typ));
    tmp -> key = L -> key;
    tmp -> next = R;
    R = tmp;
    Reverse(L -> next, R);

}
else
    return(R);

}
```

2. Problem 4.5 in the text

   Answer:

   A complete binary tree has the maximum number of nodes:

Depth	Number of nodes
0	1
1	1+2
2	1+2+4
3	1+2+4+8
4	1+2+4+8+16
...	
h	$2^0 + 2^1 + 2^2 + \ldots + 2^h$

   Number of nodes in a complete binary tree with height h:

$$\sum_{i=0}^{h} 2^i = 2^{h+1} - 1$$

3. Problem 3.24 in the text

   Answer:

   Yes, there is enough stack space. The call tree is a binary tree, each node is a call to the function fib. There're about 2^{50} such calls, so the running time is long. But the depth of the tree is about 50, function calls are like depth-first search. Therefore as long as there is space for about 50 function calls on the stack, it should be fine.