LINKED LIST EXERCISES

1. What is wrong with the following declaration?

```
struct element { double value; struct element link; } ;
```

2. Suppose that a linked list is made up of nodes of type

```
typedef struct node* link;
struct node {
   int key;
   link next;
};
```

that you are given a pointer "list" of type link, which points to the first node of the list; and that the last node has NULL as its link.

(a) Write a code fragment to delete the second node of the list. Assume that there are at least two nodes on the list.

(b) Write a code fragment to add a new node with key 17 just after the second node of the list. Assume that there are at least two nodes on the list.

(c) Write an iterative function count() that takes a link as input, and prints out the number of elements in the linked list.

(d) Write an iterative function max() that takes a link as input, and returns the value of the maximum key in the linked list. Assume all keys are positive, and return -1 if the list is empty.

3. Repeat parts (c) and (d) above, but use a recursive function.

4. Repeat 2 (c), but assume the linked list is circular, i.e., the last node points to the first node.

5. What is printed by the following code fragment on your system, and what does it mean?

```
int x, y[10];
char z;
printf("%d %d %d\n", sizeof(x), sizeof(y), sizeof(z));
```

6. What is the difference between the following given the following declaration

```
typedef struct node* link;
struct node { int key; link next; };
```

```
A. link x = malloc(sizeof(*x));
B. link x = malloc(sizeof(*link));
C. link x = malloc(sizeof(struct node));
```

7. What does the following program print out?

```
#include <stdio.h>
#include <stdlib.h>
typedef struct node* link;
struct node {
   int key;
   link next;
};

int main(void) {
   link x, y, t;
   x = malloc(sizeof *x);
   y = malloc(sizeof *y);
   x->next = y; x->key = 1;
   y->next = x; y->key = 1;
   for (t = x; t->key < 100; t = t->next)
      t->key = x->key + y->key;
   printf("%d\n", t->key);
   return 0;
}
```