2.5 Exercises

Assume that the following types and variables have been defined in C and are available for use:

```
typedef char AirportCode[4]

typedef   struct NodeTag {
                AirportCode       Airport;
                struct NodeTag    *Link;
          } NodeType;


NodeType *L, *M, *N;
```

1. Write a function, InsertNewFirstNode(A,&L), which inserts a node having the airport code A as the new first node of list, L, where &L is the address of variable L.
2. Write a function, DeleteFirst(&L), which deletes the first node of a linked list L.
3. Given a non-null pointer N to a node of a list L, and a pointer M to a new node to be inserted, write a C function to insert the node that is M's referent *before* the node that is N's referent on list L. [*Hint:* Adjust pointers to insert M after N and then swap the airport codes in N and M.]
4. Write a function, Copy(L), which makes a copy of a linked list, L, and returns a pointer to the first node of the copy.
5. Write a function, Reverse(&L), which reverses the order of the nodes on list L. For example, if L == (ZRH, GLA, YYZ) beforehand, then executing Reverse(&L) changes L to be the list L == (YYZ, GLA, ZRH). [*Hint:* Write two subroutines to remove the first node N of a list L1, and to insert a node N as the new first node on a list L2. Then, starting with an empty list L2 = NULL, successively remove nodes from L1 and insert them on L2 until L1 is empty.] (*Note:* The airport codes: ZRH, GLA, and YYZ stand for Zürich, Switzerland; Glasgow, Scotland; and Toronto, Ontario, respectively.)
6. What is wrong with the following search program for finding the node on list L containing the airport code A and returning a pointer to it?

```
    NodeType *FindNode(char *A, NodeType *L)
    {

        while ( (strcmp(L->Airport,A) != 0) && (L != NULL) ) {
5           L = L->Link;
        }

        return L;
    }
```

How could you fix the bug in the program above by changing only one line?