2) (10 pts) DSN (Linked Lists)

An alternate method of storing a string is to store each letter of the string in a single node of a linked list, with the first node of the list storing the first letter of the string. Using this method of storage, no null character is needed since the next field of the node storing the last letter of the string would simply be a null pointer. Write a function that takes in a pointer to a linked list storing a string and returns a pointer to a traditional C string storing the same contents. Make sure to dynamically allocate your string in the function and null terminate it before returning the pointer to the string. Assume that a function, length, exists already (that you can call in your solution), that takes in a pointer to a node and returns the length of the list it points to. The prototype for this function is provided below after the struct definition.

```
typedef struct node {
    char letter;
    struct node* next;
} node;

int length(node* head);

char* toCString(node * head) {

    // Find length of the list.
    int len = length(head); // 1 pt

    // 2 pts - allocate space.
    char* res = malloc((len+1)*sizeof(char));

    int i=0; // 1 pt - init loop variable

    // Go through list - 4 pts, 1 pt per line.
    while (head != NULL) {
        res[i] = head->letter;
        head = head->next;
        i++;
    }

    // 1 pt - null terminate string.
    res[len] = '\0';

    // 1 pt - return pointer to string.
    return res;
}
```

Note: other solutions are possible, map points accordingly.