Massachusetts Institute of Technology
Department of Electrical Engineering and Computer Science

6.087: Practical Programming in C

IAP 2010

**Problem Set 5**
Linked lists, trees

**Out:** January 19, 2010. **Due:** January 20, 2010.

### Problem 5.1

In this problem, we continue our study of linked lists. Let the nodes in the list have the following structure

```
struct node
{
  int data;
  struct node* next;
};
```

Use the template in Lec06 to add elements to the list.

(a) Write the function `void display(struct node* head)` that displays all the elements of the list.

(b) Write the function `struct node* addback(struct node* head,int data)` that adds an element to the end of the list. The function should return the new head node to the list.

(c) Write the function `struct node* find(struct node* head,int data)` that returns a pointer to the element in the list having the given data. The function should return NULL if the item does not exist.

(d) Write the function `struct node* delnode(struct node* head,struct node* pelement)` that deletes the element pointed to by `pelement` (obtained using find). The function should return the updated head node. Make sure you consider the case when `pelement` points to the head node.

(e) Write the function `void freelist (struct node* head)` that deletes all the element of the list. Make sure you do not use any pointer after it is freed.

(f) Write test code to illustrate the working of each of the above functions.

All the code and sample outputs should be submitted.

**Problem 5.2**

In this problem, we continue our study of binary trees. Let the nodes in the tree have the following structure

```
struct tnode
{
  int data;
  struct tnode* left;
  struct tnode* right;
};
```

Use the template in Lec06 to add elements to the list.

(a) Write the function `struct tnode* talloc(int data)` that allocates a new node with the given data.

(b) Complete the function `addnode()` by filling in the missing section. Insert elements $3, 1, 0, 2, 8, 6, 5, 9$ in the same order.

(c) Write function `void preorder(struct tnode* root)` to display the elements using pre-order traversal.

(d) Write function `void inorder(struct tnode* root)` to display the elements using in-order traversal. Note that the elements are sorted.

(e) Write function `int deltree(struct tnode* root)` to delete all the elements of the tree. The function must return the number of nodes deleted. Make sure not to use any pointer after it has been freed. (Hint: use post-order traversal).

(f) Write test code to illustrate the working of each of the above functions.

All the code and sample outputs should be submitted.

6.087 Practical Programming in C

January (IAP) 2010