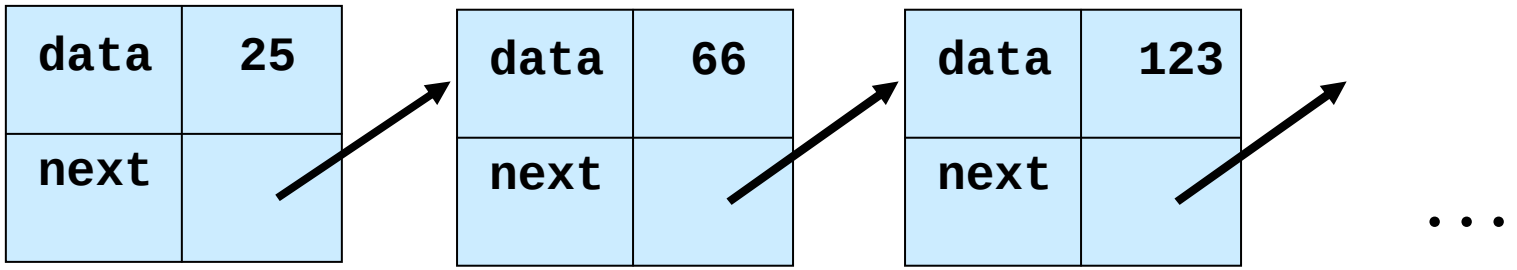
Linked Structures

•Self-referential structs can be used to create linked data structures:

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

• **`next`** holds the address of a **`node struct`**

• through the next pointer we can link **`node`** structs together:

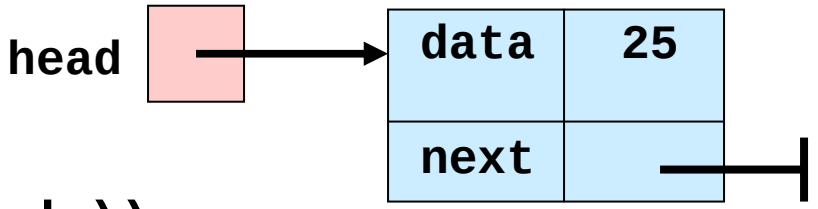


Linked List

- Ordered Collection of data
- Need a single variable which is pointer to 1st node on list
- nodes are linked together in-order by following **`next`** pointers

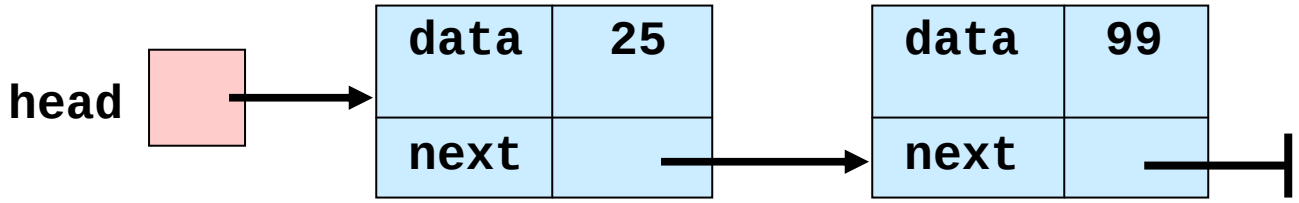
```
node *head;
head = malloc(sizeof(node));
head->data = 25;
head->next = NULL;
```

List of length 1:




```
head->next = malloc(sizeof(node));
head->next->data = 99;
head->next->next = NULL;
```

List of length 2:

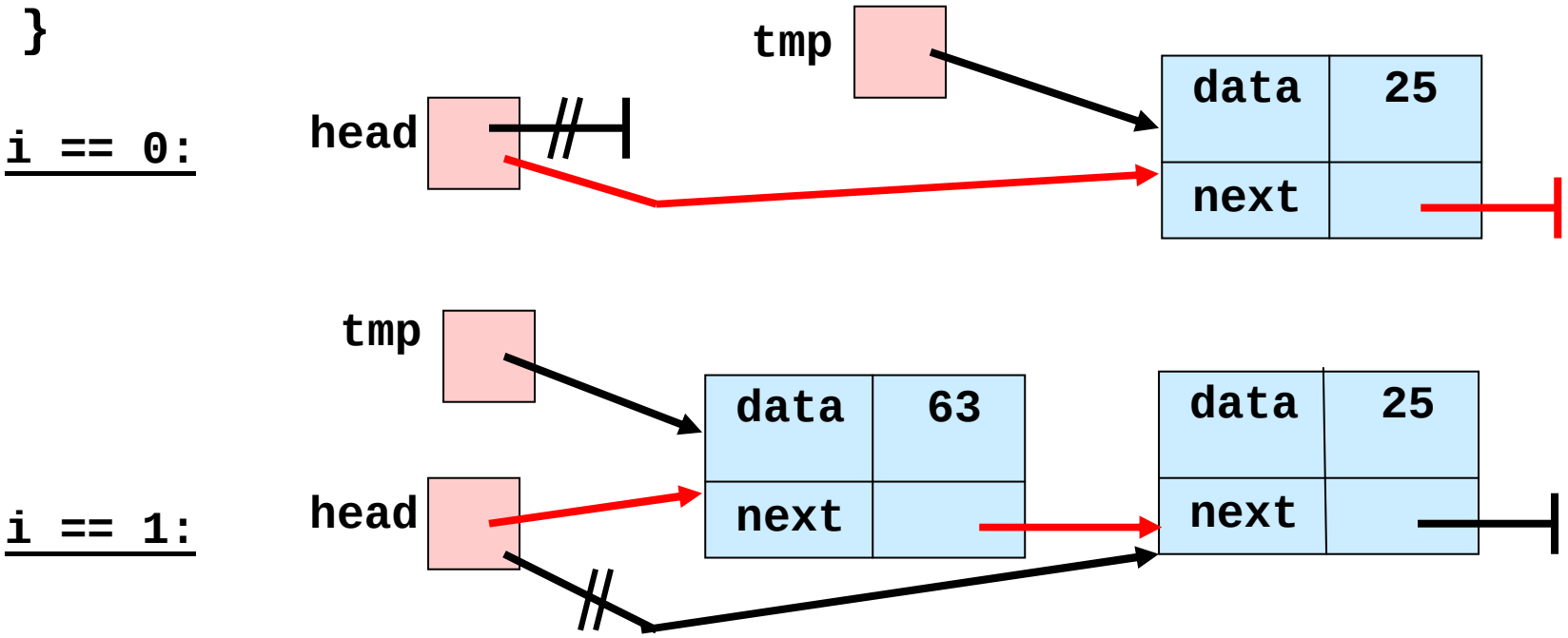


Operations on a List

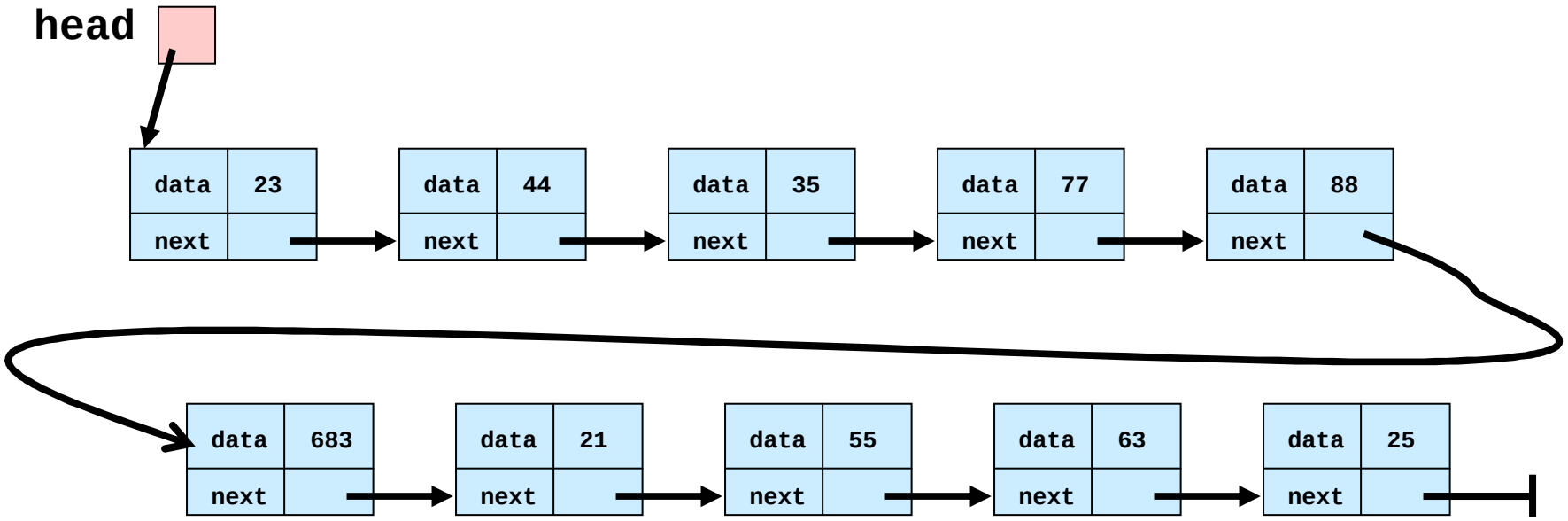
- All start at node pointed to by head pointer, and traverse next pointers to access other nodes in the list
- Accessing the ith node is O(n):
  - first access head node, follow its pointer to access the 2nd node, follow its pointer to access the 3rd node, and so on

Insert at Head of List

```
head = NULL;
for(i=0; i < 10; i++) {
        tmp = malloc(sizeof(node));
        if(tmp == NULL) { Error("malloc failed"); }
        tmp->data = GetNextDataValue();
        tmp->next = head;
        head = tmp;
}
```

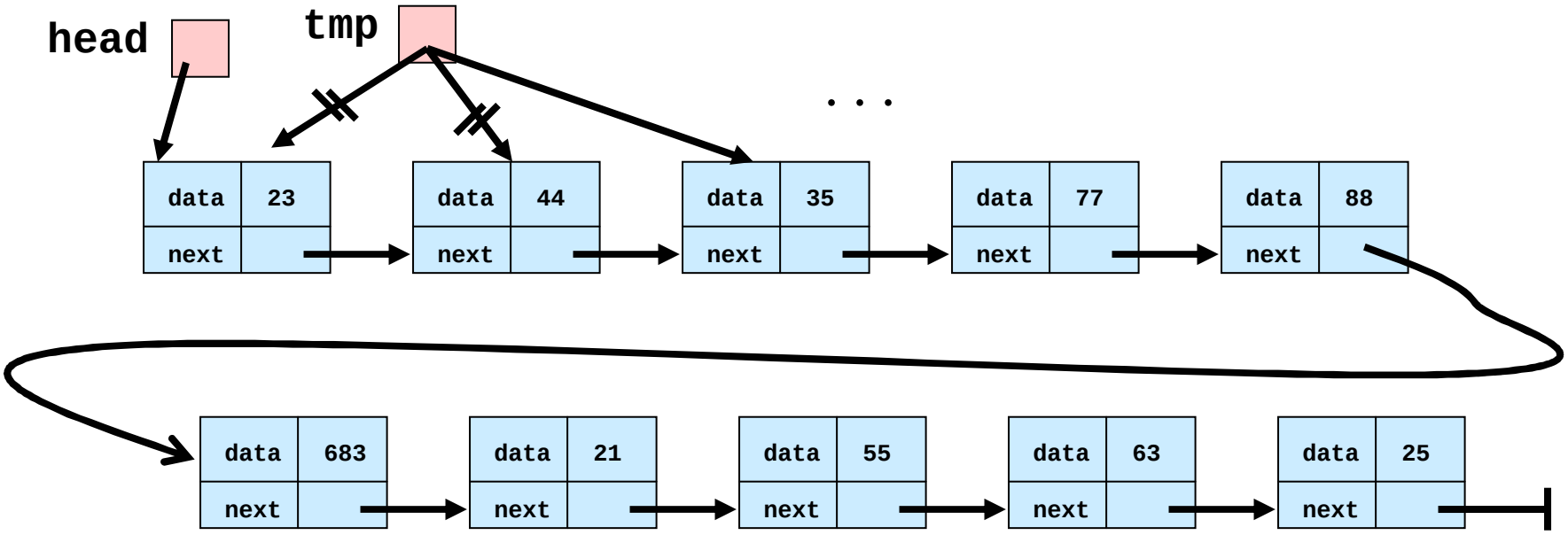


Resulting List of 10 nodes:



Traverse the List

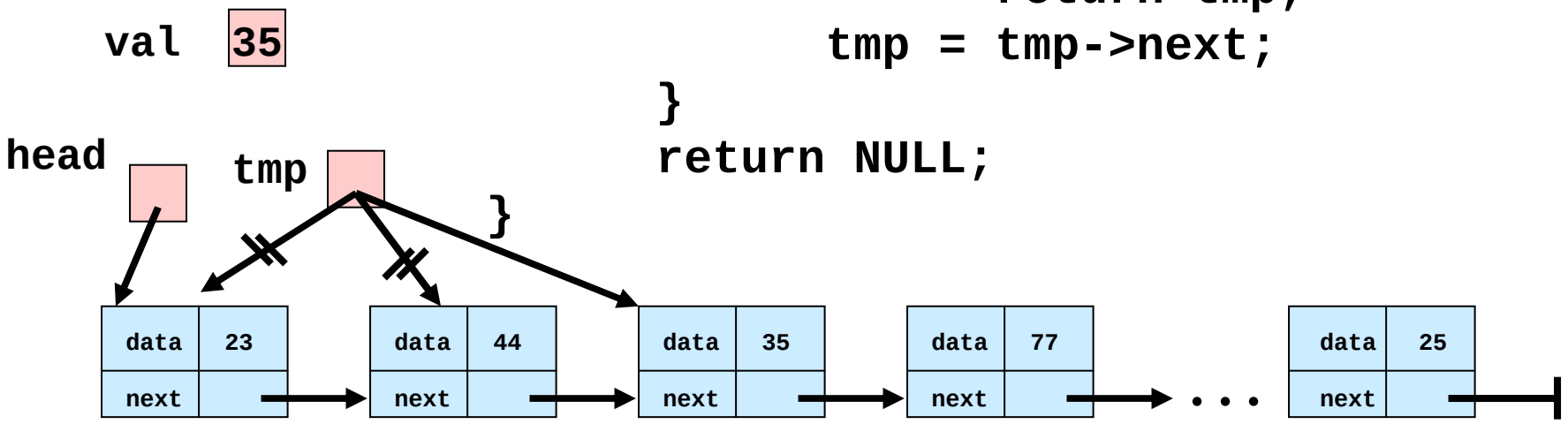
```
tmp = head;    // start at the 1st node
while(tmp != NULL) {
      printf("%d ", tmp->data);
      tmp = tmp->next; // make tmp point to next node
}
// output:  23 44 35 77 88 683 21 55 63 25
```



Find Element In List

- Start at head node, compare search value to data field
- traverse next pointers until matching data field is found, or until no more list

```
node *FindInList(node *head, int val) {
      node *tmp;
      tmp = head;
      while(tmp != NULL) {
            if(tmp->data == val)
                  return tmp;
            tmp = tmp->next;
      }
      return NULL;
}
```



Insert in the middle

```
node *new_node, *tmp, *head;

new_node = malloc(sizeof(node));
new_node->data = 20;
tmp = head->next;

// insert new_node after tmp
new_node->next = tmp->next;
tmp->next = new_node;
```