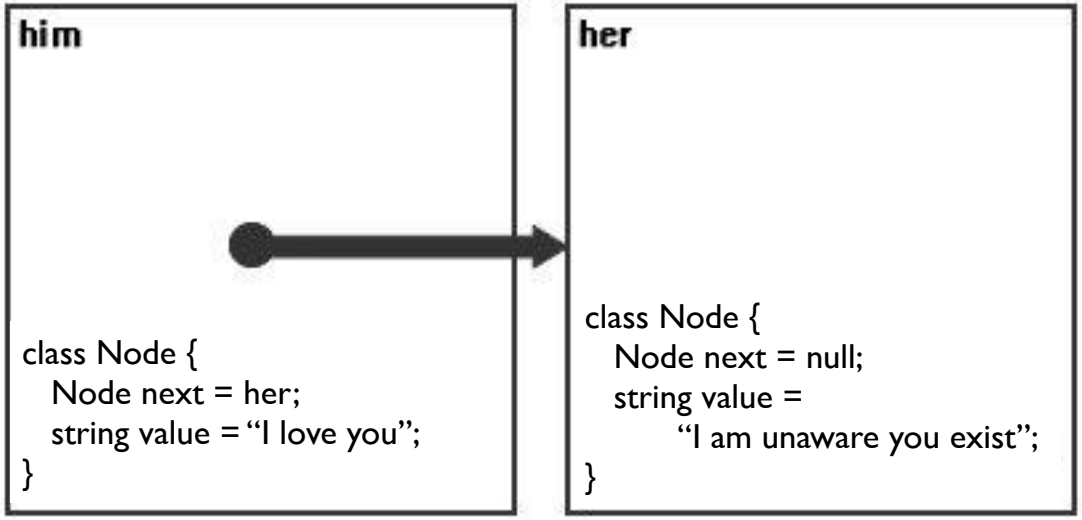
# Brief Overview of Linked Lists

The tragedy of Linked Lists




*A programmatical observation by Alishah Novin*

# Why Dynamic Data Structures

## Static Data Structures

e.g. Arrays

Ok when you know in advance how much memory you need

```
//structure that represents a film
typedef struct {
   char title[50];
} Film ;

//Global arrays
Film films[100]; // stores actual film titles
```

## Variable memory requirements

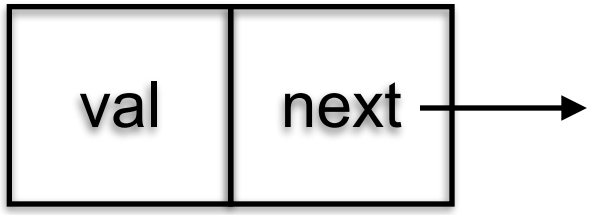
e.g. GPS Analysis assignment, data is made up of 'points'

```
<trkpt lat="53.293602000" lon="-6.179220000"><ele>21.8</ele><time>2013-03-02T10:28:04Z</time></trkpt>
<trkpt lat="53.293597000" lon="-6.179221000"><ele>21.7</ele><time>2013-03-02T10:28:04Z</time></trkpt>
<trkpt lat="53.293683000" lon="-6.179256000"><ele>21.6</ele><time>2013-03-02T10:28:33Z</time></trkpt>
```

There may be hundreds or even thousands

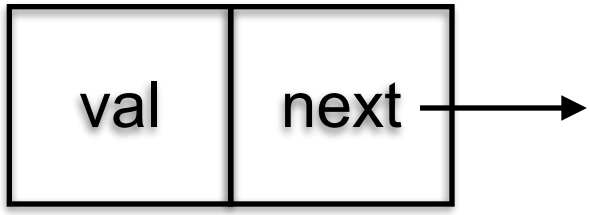
# Simple List Node

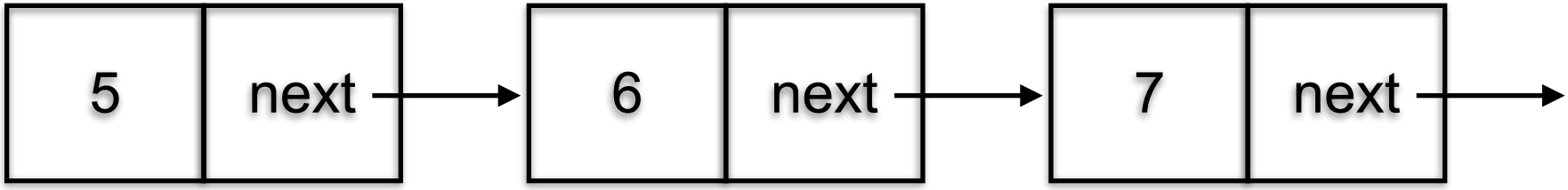
```
struct node
{
    int val;
    struct node *next;
};
```



# Simple List Node

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



# create_list

```
struct node* create_list(int val)
{
    printf("\n creating list with headnode as [%d]\n",val);
    struct node *ptr = (struct node*)malloc(sizeof(struct node));
    if(NULL == ptr)
    {
        printf("\n Node creation failed \n");
        return NULL;
    }
    ptr->val = val;
    ptr->next = NULL;

    head = curr = ptr;
    return ptr;
}
```

allocate memory

-> notation because ptr is a pointer

# Add to the list

```
struct node* add_to_list(int val)
{
    //First check to see if there is anything in the list.
    if(NULL == head)
    {
        return (create_list(val)); // list is created if it
doesn't exist already.
    }
    printf("\n Adding node to the list containing [%d]\n",val);
    // Create a node for the new data.
    struct node *ptr = (struct node*)malloc(sizeof(struct node));
    if(NULL == ptr)
    {
        printf("\n Node creation failed \n");
        return NULL;
    }
    ptr->val = val; // Add the data
    ptr->next = NULL;
    curr->next = ptr; // Link in the new node
    curr = ptr;             // Update the pointer

    return ptr;
}
```

# Printing the list contents

```
void print_list(void)
{
    struct node *ptr = head;

    while(ptr != NULL)
    {
        printf("\n [%d] \n",ptr->val);
        ptr = ptr->next;
    }

    return;
}
```

head is a global variable

# Printing the list contents

```
void print_list(void)
{
    struct node *ptr = head;

    while(ptr != NULL)
    {
        printf("\n [%d] \n",ptr->val);
        ptr = ptr->next;
    }

    return;
}
```

head is a global variable

Useful code template for working through an existing list.

# Summary

Why do we need linked lists?

Simple singly linked list

Data can only be added at the end

Simple print function for working through the list.