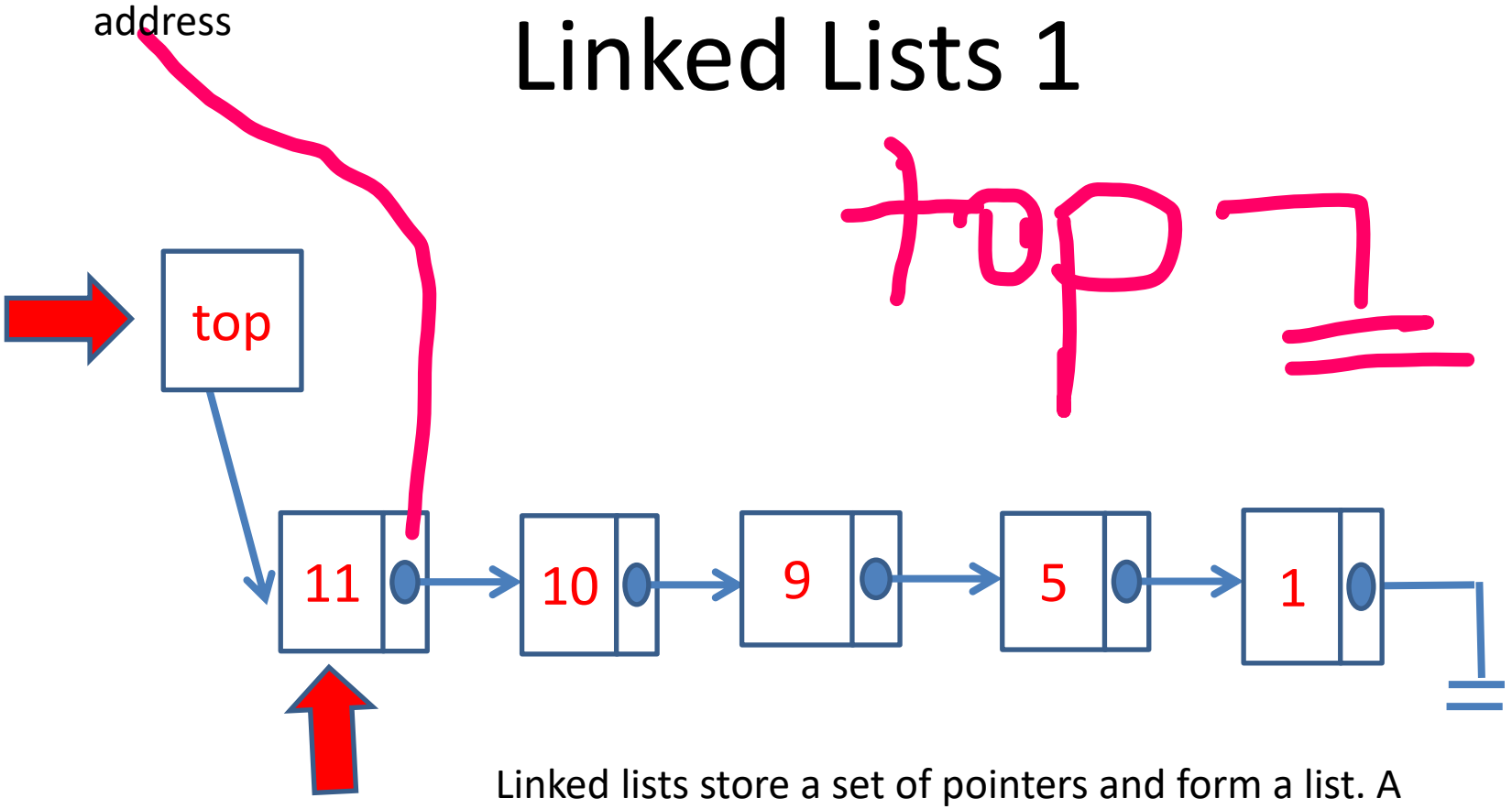
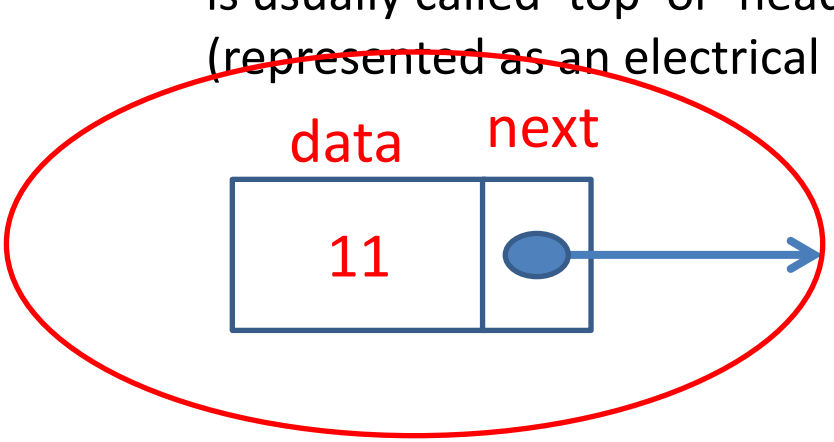
COMP1603
Linked Lists 1

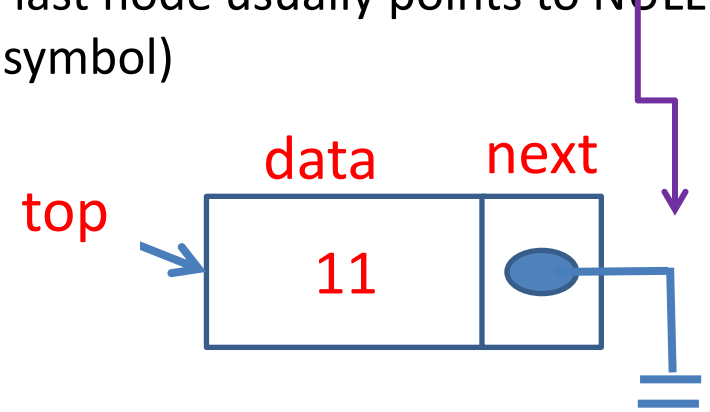


Linked lists store a set of pointers and form a list. A pointer contains a **memory address**. A linked list is a set of structs or Nodes that are connected. The "next" field contains the address of the next Node or NULL if the list come to an end.

Linked Lists Intro

- A linked list is a linear data structure which consists of a set of nodes that are linked together using memory addresses (pointers). The list has a head which is usually called top. A linked list may also be empty.
- In order to connect to another node, a linked list cell contains a ‘next’ field (diagram).
- The beginning of the list also has to be referenced and this pointer is usually called ‘top’ or ‘head’. The last node usually points to NULL (represented as an electrical earth symbol)





Note on Linked Lists

- Linked lists are dynamic structures compared to arrays, which are static. They can help save storage.
- One can allocate or deallocate storage as necessary with linked lists but array sizes are fixed.
- Linked lists facilitate dynamic programming.
- Examples of Linked Lists
  - -a list of student information
  - -a list of information on books at a library

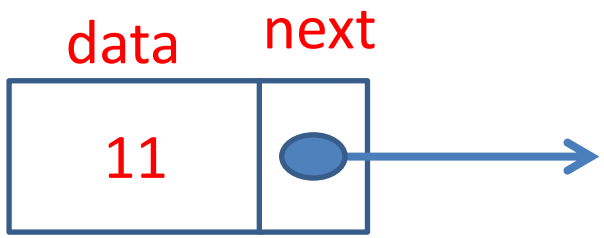
Some Linked List Applications

- Linked Lists can be used to implement Stacks , Queues.
- Linked Lists can also be used to implement Graphs
- Undo functionality in Photoshop or Word. Linked list of states.
- A polynomial can be represented in an array or in a linked list by simply storing the coefficient and exponent of each term.
- Note that for any polynomial operation , such as addition or multiplication of polynomials , linked list representation is more easier to deal with.
- Overall, linked lists are useful for dynamic memory allocation and deallocation.

Adapted-Quora.com

Sample Declarations

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

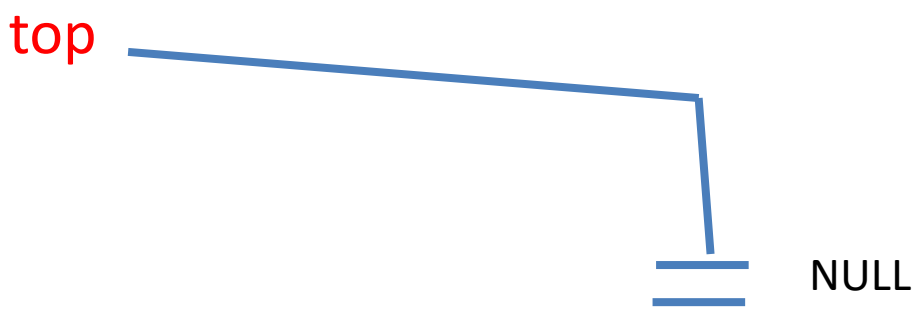


Means that this node can point to another node with the same structure. This other node would have the following fields:

int data;

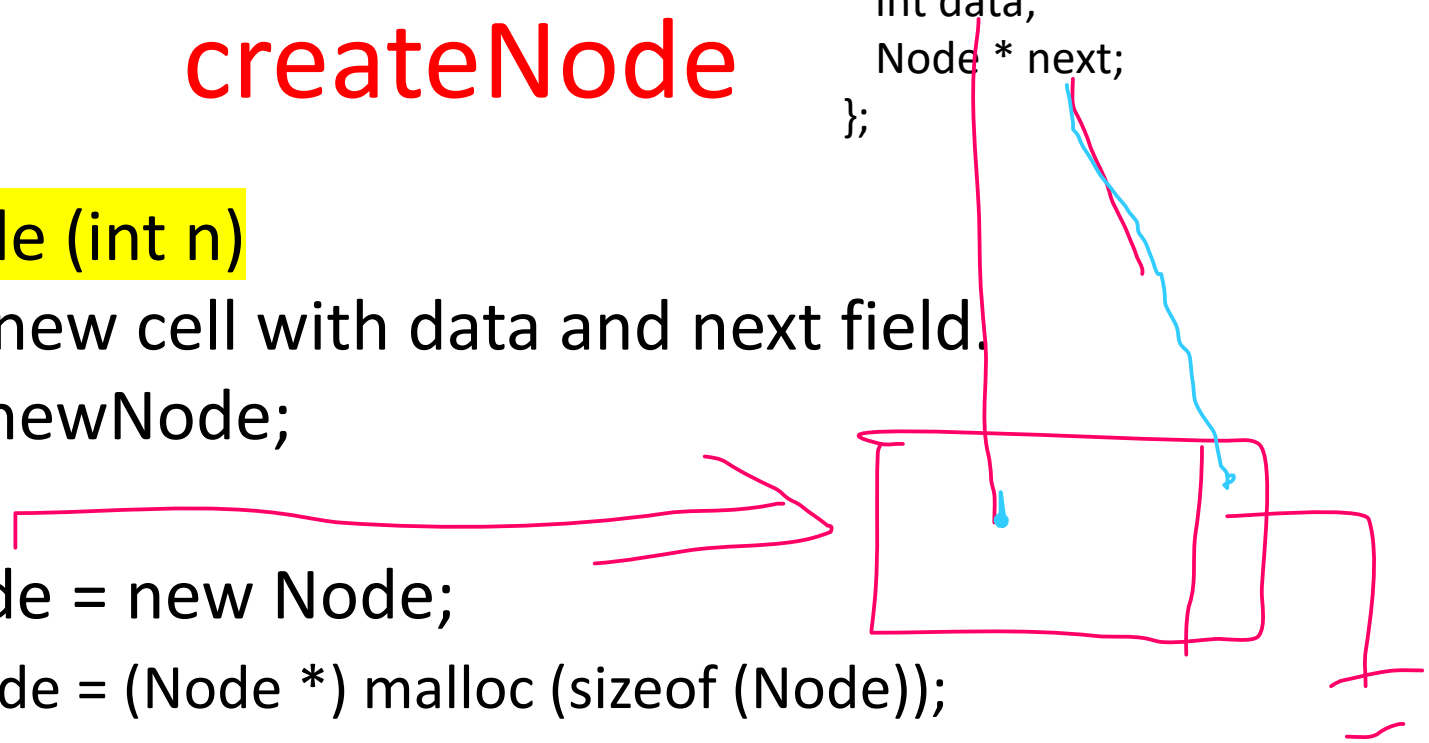
Node * next;

Draw an Empty List



createNode

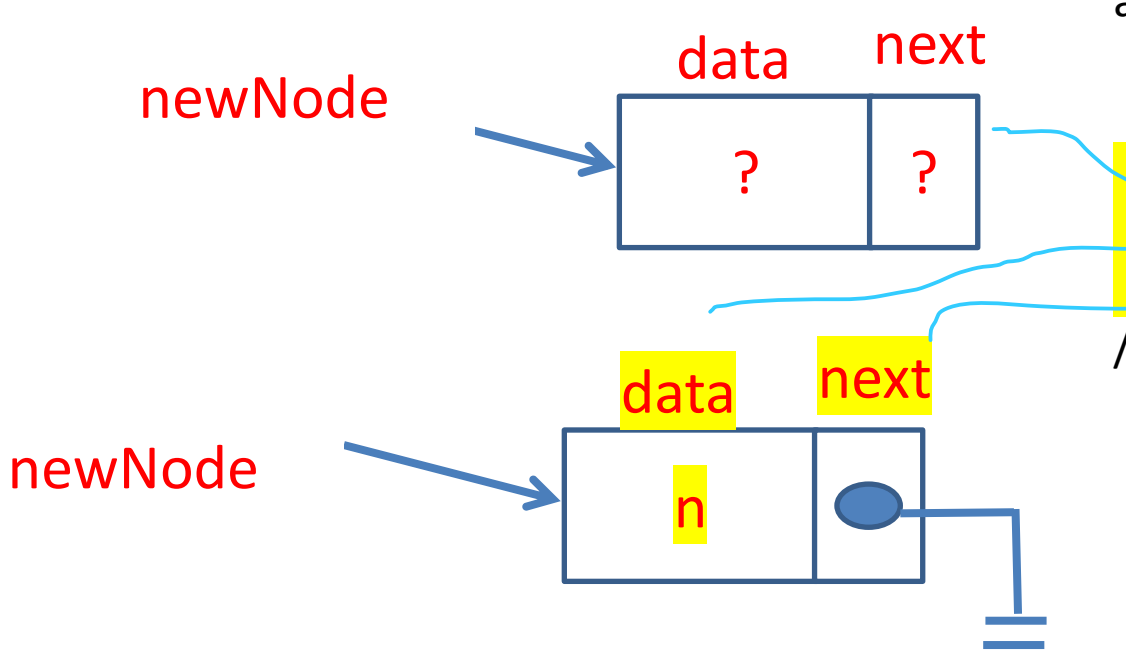
```
struct Node {
   int data;
   Node * next;
};
```



```
createNode (int n)
//Make a new cell with data and next field.
   Node * newNode;

    newNode = new Node;
    //newNode = (Node *) malloc (sizeof (Node));
   newNode->data = n;
   newNode->next = NULL; //next is undefined
                         //initially
//returns a pointer to the new node so it can be
accessed
}
```

createNode

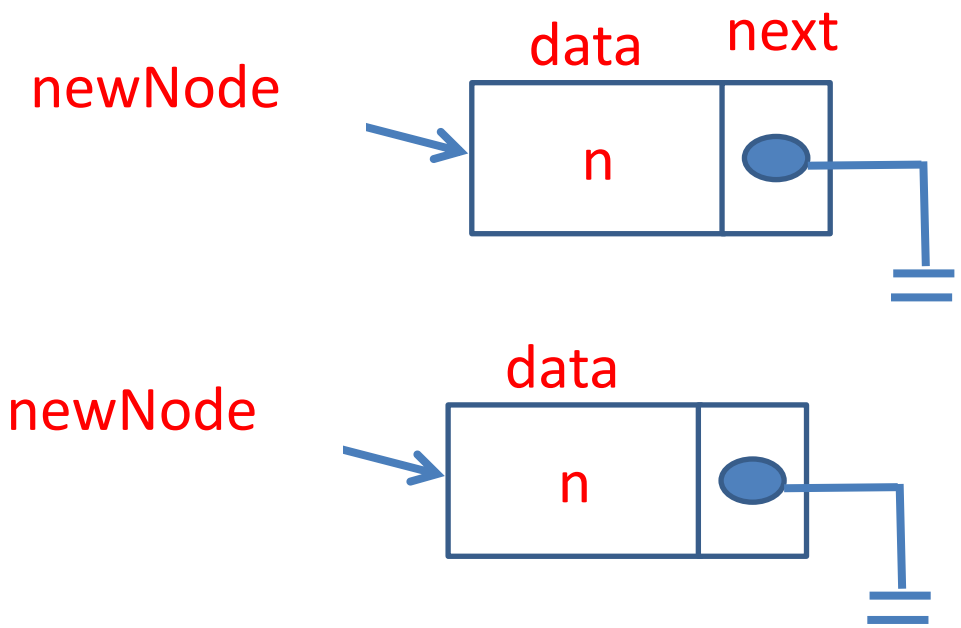
Each time createNode is called the following happens

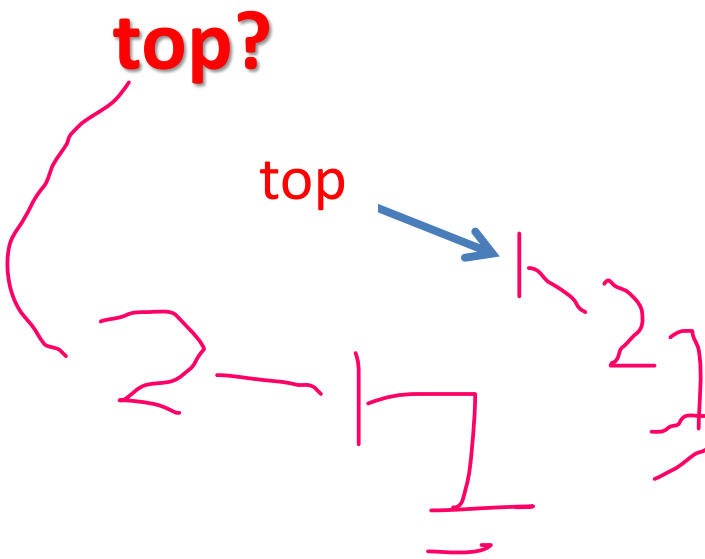



```
Node * createNode (int n) {
//Make a new cell with data
and next field.
  Node * newNode;

  newNode = new Node;
  newNode->data = n;
  newNode->next = NULL;
//next is undefined
  initially
```

createNode in a Program

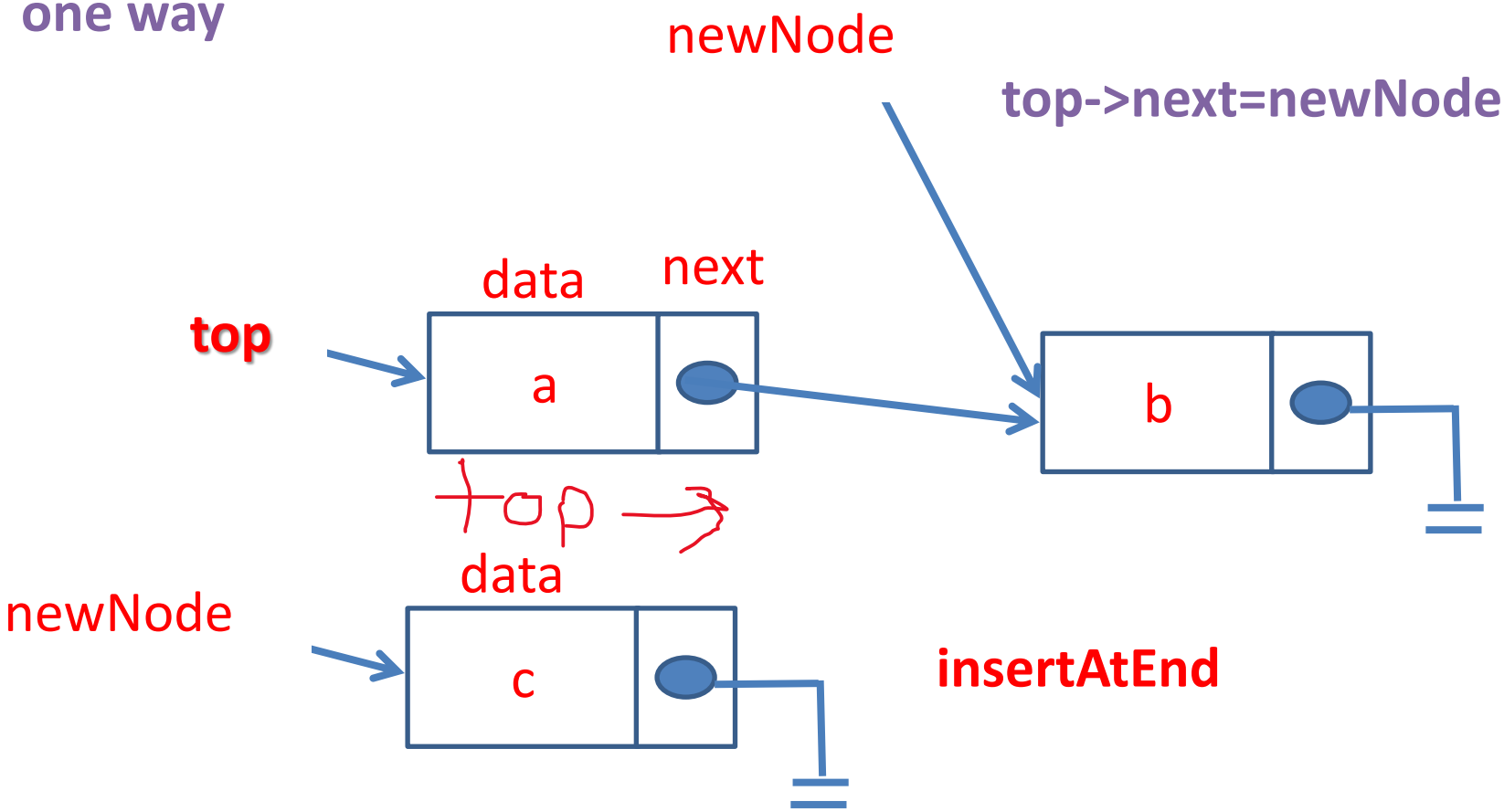
createNode creates a Node everytime and returns a pointer to that new Node.
To do useful programming, each Node needs to be connected together to build a linked list.
Sometimes a list can be built in order, reverse order etc.



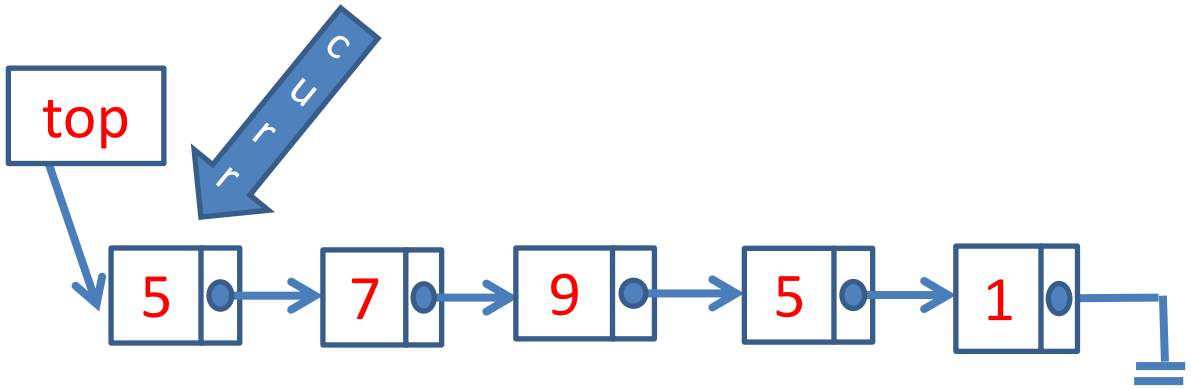


createNode in a Program

Connect nodes together – one way

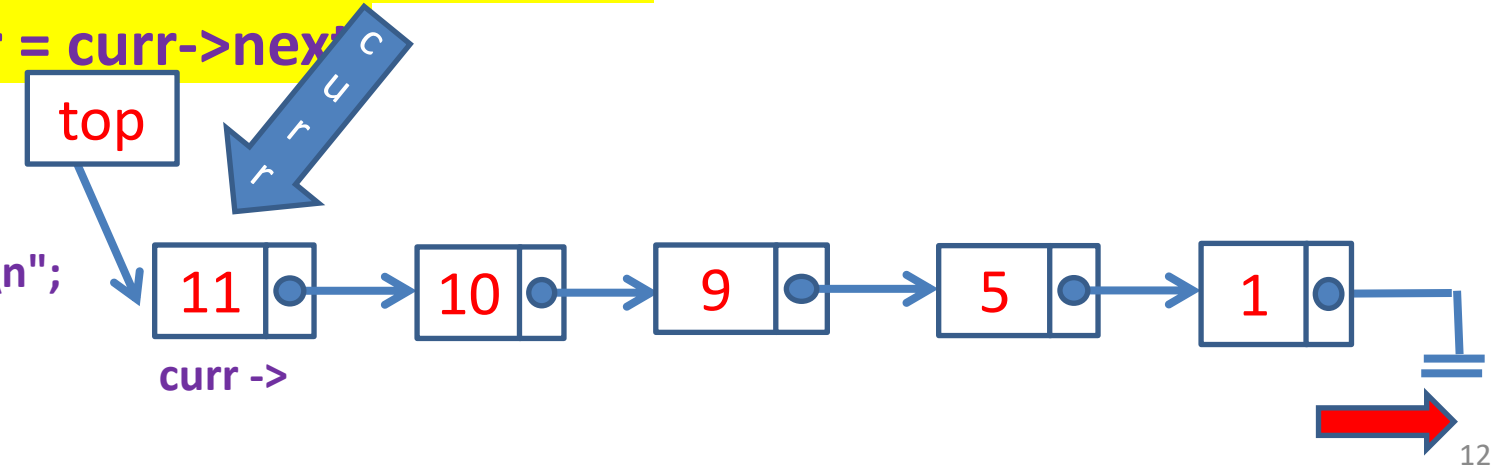


Meaning of curr=curr->next



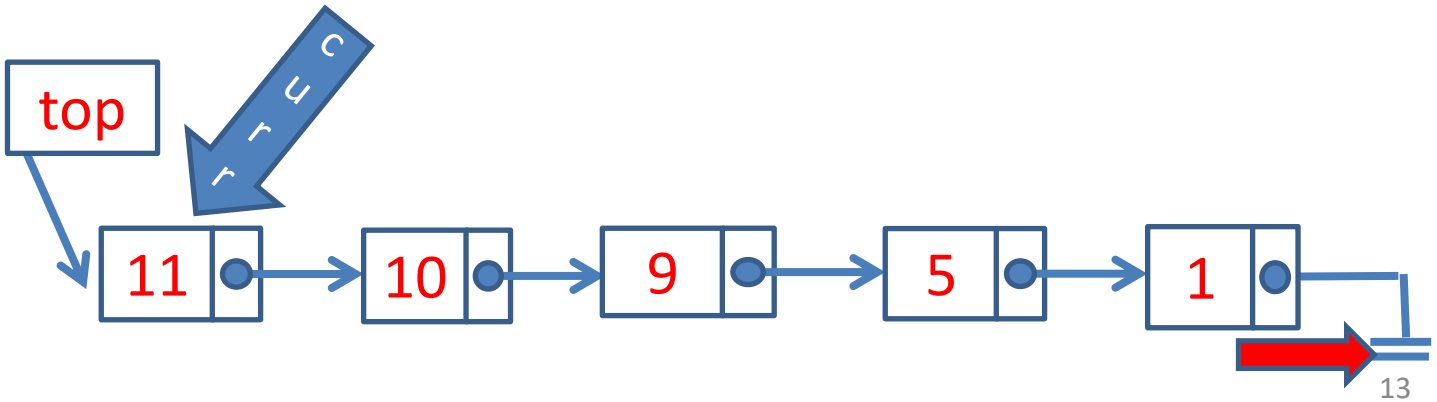
printList

```
void printList (Node * top) {
  Node * curr;
  if (top == NULL) {
    cout << "List empty." <<endl;
    return;
  }
  curr = top;
  while (curr != NULL) {
    cout << curr -> data << "\t";
    curr = curr->next;
  }

cout << "\n";
}
```



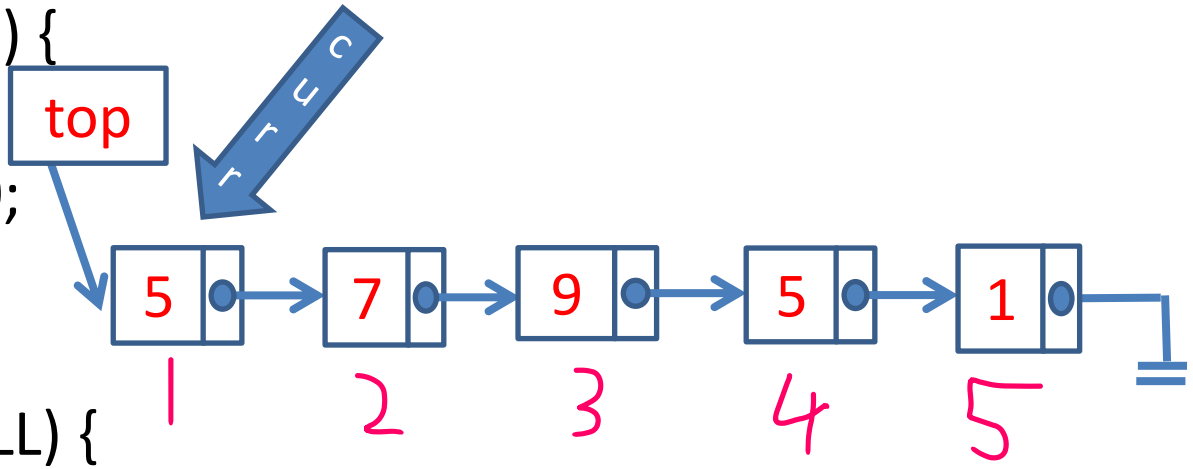
Contains-is key in the list?

```
bool contains (Node * top, int key) { //key is 9, 20

  Node * curr;

  curr = top;
  while (curr != NULL) {
    if (curr->data == key)
      return true;
    else
      curr = curr->next;
  }


 return false;
}
```



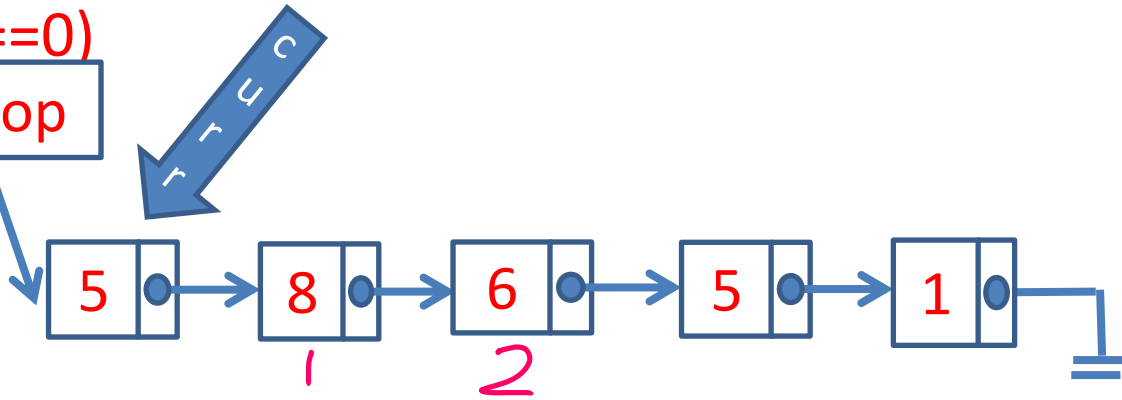
Size of linked list

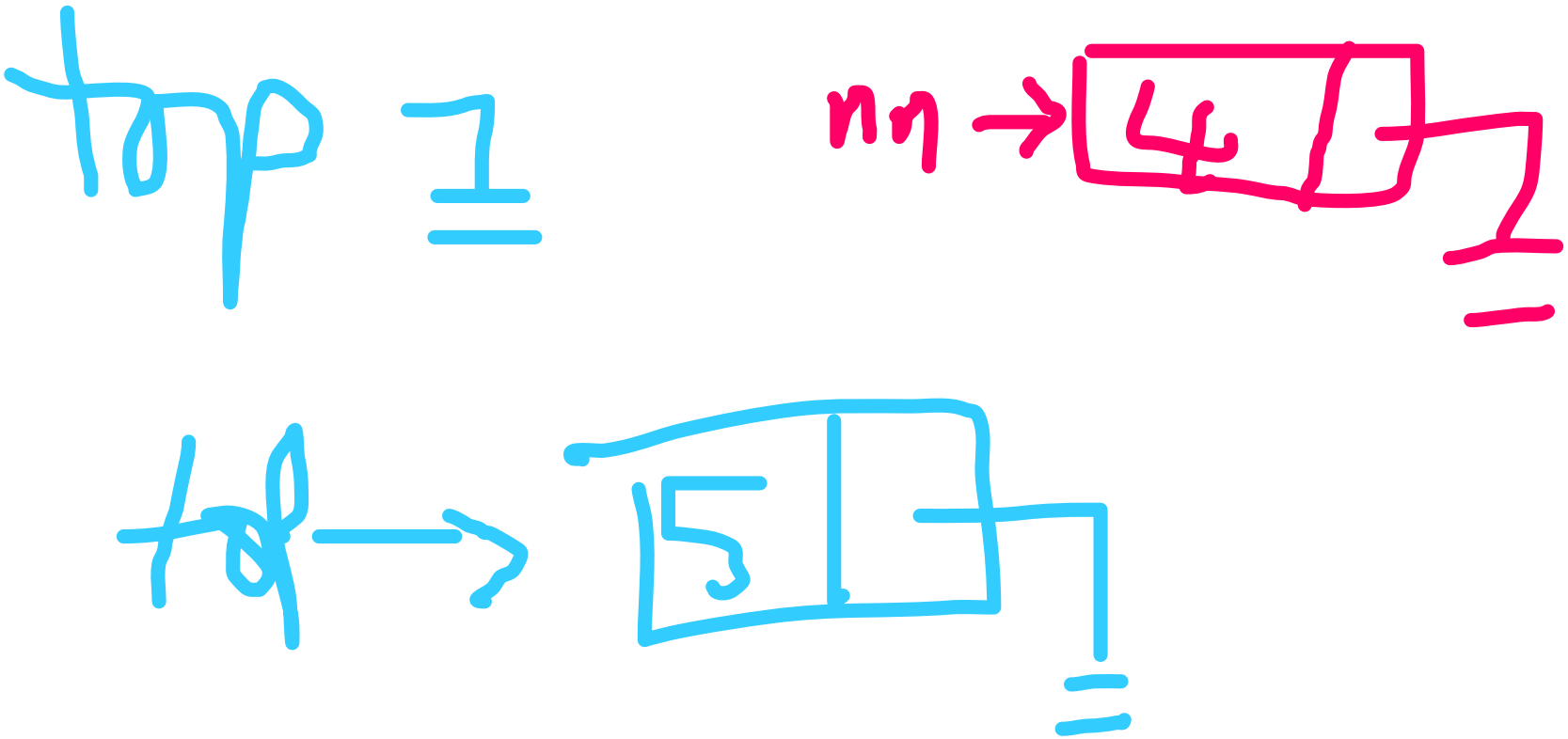
```
int size (Node * top) {
  Node * curr;
  int numNodes = 0;

  curr = top;
  while (curr != NULL) {
    numNodes = numNodes + 1;
    curr = curr->next;
  }
  return numNodes;
}
```



Problem-Count the number of times an even integer occurs in a list

```
int countEven (Node *top) {
  int count=0;
  Node *curr=top;
  while (curr!=NULL) {
        if (curr->data%2==0)
              count++;
        curr=curr->next;
  }
  return count;
//   cout<<"Count even="<<countEven(top) ;
}
```



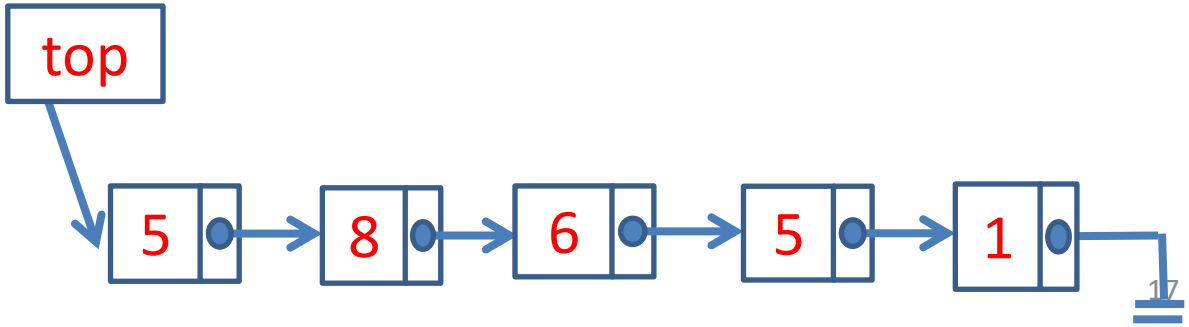

top
nn
4
top
5

Review
and Exercises for Whiteboard-Also draw and show the approach

1.Add one to all elements of a linked list, top

Useful code:

```
 curr = top;
   while (curr != NULL) {
      numNodes = numNodes + 1;
      curr = curr->next;
   }
```



- 2.Find the sum of the elements in a linked list top.

Useful code:

```
curr = top;
  while (curr != NULL) {
     numNodes = numNodes + 1;
     sum=
     curr = curr->next;
  }
```

3. Assume there are 10 elements in a linked list top. Print the 5^{th} element.

SOLN

Do curr=top then

curr=curr->next 4 times

Print curr->data

What is a linked list

- A linked list is a linear data structure which consists of a set of nodes that are linked together using memory addresses (pointers). The list has a head which is usually called top. A linked list may also be empty.

What are two uses of linked lists?

- Store data for students in a class (data?)
- Store customers who bought items (data?)

What is the difference between a static and dynamic structure?

Write code/pseudocode to find the number of times “5” occurs in a linked list called **head**.

- Useful code:
- curr = top;
- while (curr != NULL) {
- numNodes = numNodes + 1;
- curr = curr->next;
- }

Draw the following

- An empty list top
- Add the first node with “1”
- Add the second node with “2”
- Add the third node with “3”
- Explain how to delete “2”
- Draw new list