LAB 5
LINKED LISTS

Data Structures
2020-2021

AGENDA

1. Node Class
   i. Declaration & Constructors
2. LinkedList Class
   i. Declaration & Constructors
   ii. Task1: Append Function
   iii. InsertAt Function
   iv. DeleteAt Function
   v. Task 2: MoveNode Function

1 - NODE CLASS DECLARATION & CONSTRUCTORS

Class (Node)

1. **Has member variables of**
   - The value of the node
   - A pointer to a node
2. **Has two constructors**
   - Default Constructor
   - Constructor With parameters

CLASS 'NODE': IMPLEMENTATION

LinkedList.h

```
class Node {
public:
        int data;
        Node * next;
        Node();
        Node(int value);
};
```

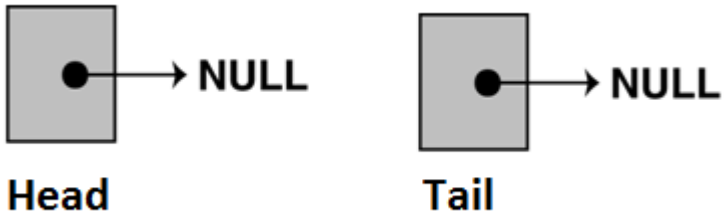
LinkedList.cpp

```
#include "LinkedList.h"


Node::Node() {
  data = 0;
  next = NULL; //next = 0;
}
Node::Node(int value) {
  data = value;
  next = NULL; //next = 0;
}
```

2 - LINKEDLIST CLASS DECLARATION & CONSTRUCTORS

Class (LinkedList)

1. **Has member variables of**
   - A pointer to a node (Head)
   - A pointer to a node (Tail)
   - The number of elements
2. **Has one constructor**
   - Default Constructor



CLASS 'LINKEDLIST': IMPLEMENTATION

LinkedList.h

```
class LinkedList {
        Node * head;
        Node * tail;
        int size;
public:
        LinkedList();
};
```

LinkedList.cpp

```
…
…
LinkedList::LinkedList() {
        head = tail = NULL;
        size = 0;
}
```

3 - APPEND FUNCTION

Write function "Append" that adds new node, for a given value, to the LinkedList.

```
void Append(int val);
```

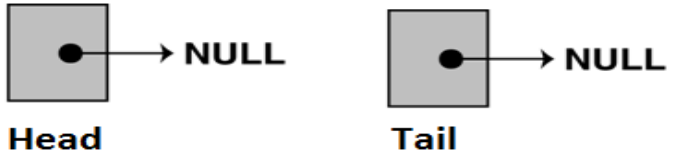
We have 2 cases while adding a node:-

- The LinkedList is empty.
- The LinkedList is non-empty.

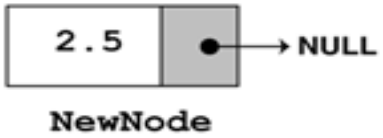
3 - APPEND FUNCTION

CASE 1: EMPTY LINKEDLIST

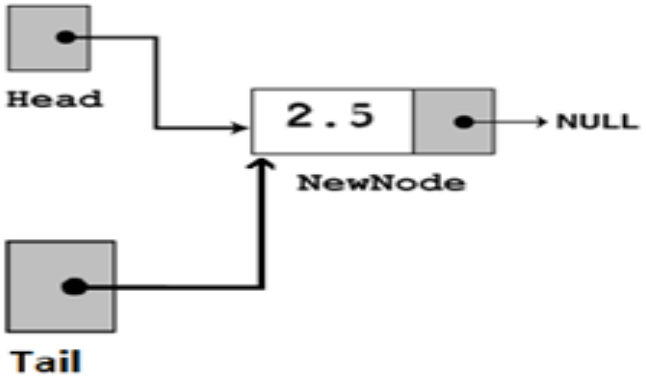
Initially, the head and tail node pointers are NULL.



<u>Step 1:-</u> Create new node for the given value.



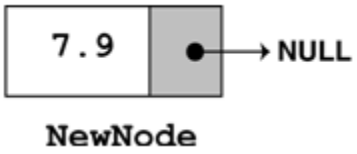
<u>Step 2</u>:- Add it to the LinkedList



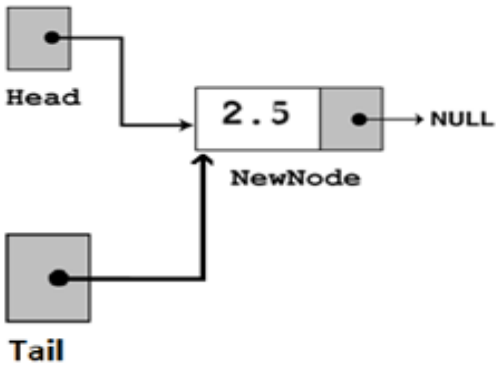
3 - APPEND FUNCTION

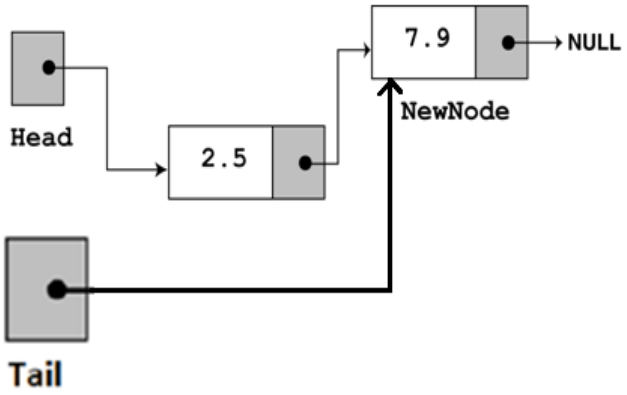
CASE 2: NON-EMPTY LINKEDLIST

Step 1:- Create new node for the given value.



Step 2:- Add it to the LinkedList





3 - APPEND FUNCTION

PSEUDO-CODE

Create a new node for the given value

If there are no nodes in the list
    Make the new node the first node.
Else
    Add the new node to the end of the list.
End If.

Increase size

3 - APPEND FUNCTION: IMPLEMENTATION

LinkedList.h

```
class LinkedList{

        Node * head;

        Node * tail;

        int size;

public:


        // Methods & Const.'s

        LinkedList();

        void Append(float val);

};
```

LinkedList.cpp

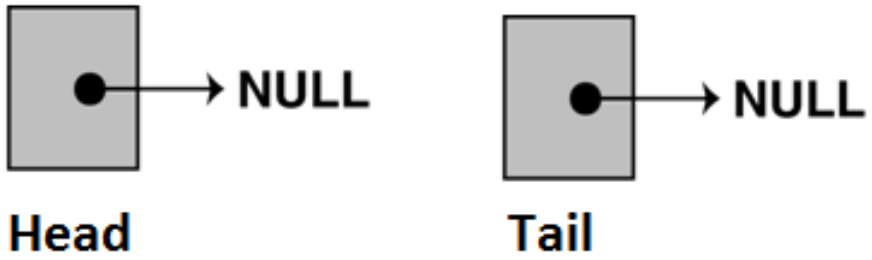
```
…
…

void LinkedList::Append(float value) {

        Node * temp =
                new Node(value);

        if(head == NULL) {
        // if (size == 0)
                head = tail = temp;
        } else {
                tail->next = temp;
                tail = tail->next;
        }

        size++;

}
```

MAIN & TEST

Main.cp

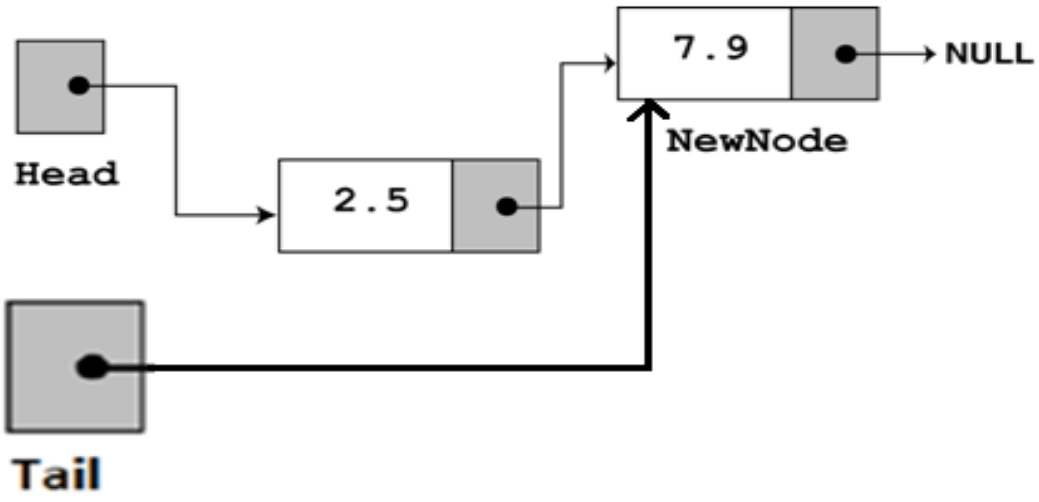
p

```
LinkedList myList;
```



MAIN (TESTING “APPEND” FUNCTION)

```
LinkedList myList;
myList.Append(2.5);
myList.Append(7.9);
```

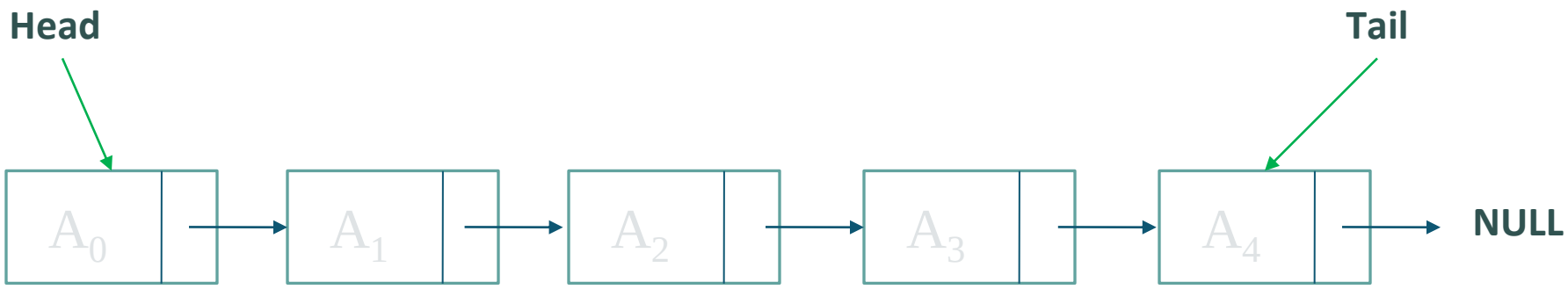


4 – INSERTAT FUNCTION

add an element at specific position.

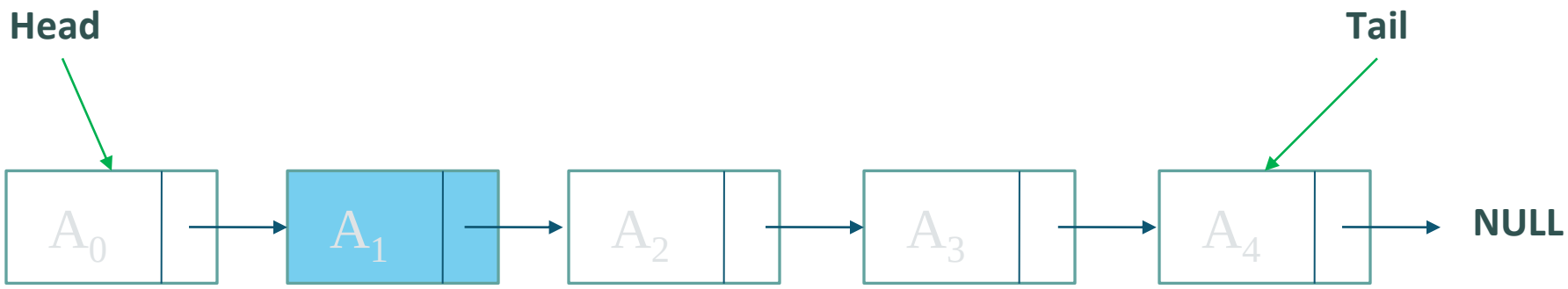
ADDING AN ELEMENT

Insert at (2)



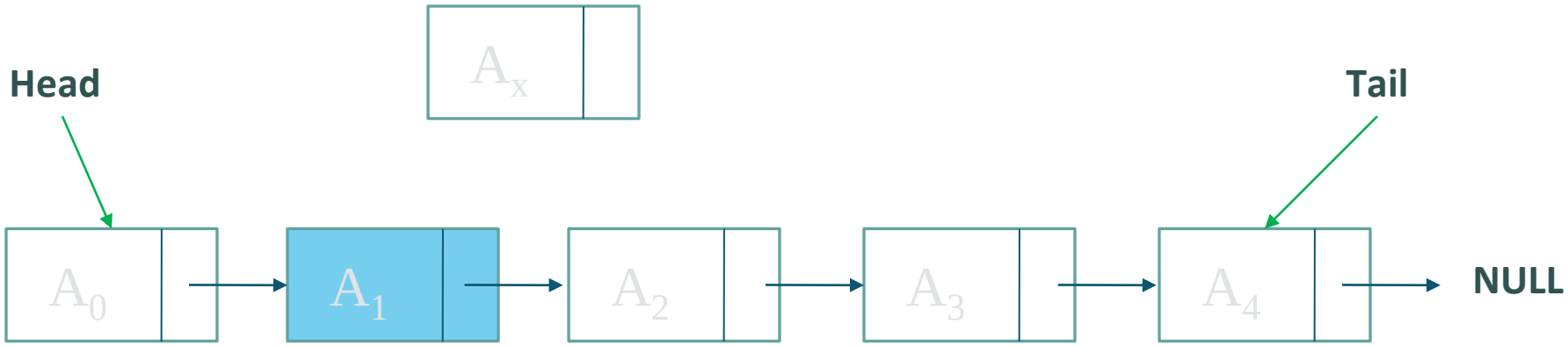
ADDING AN ELEMENT

Insert at (2)



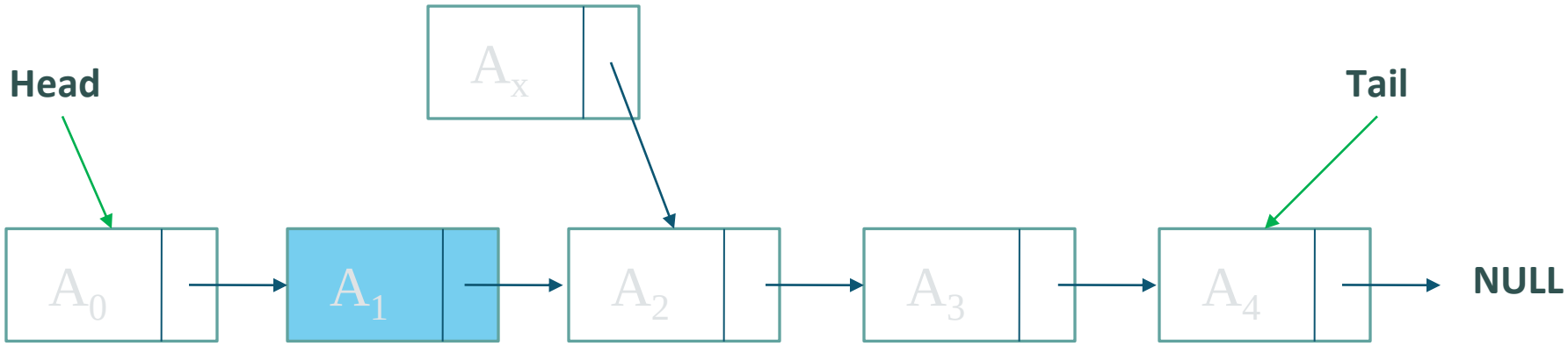
ADDING AN ELEMENT

Insert at (2)



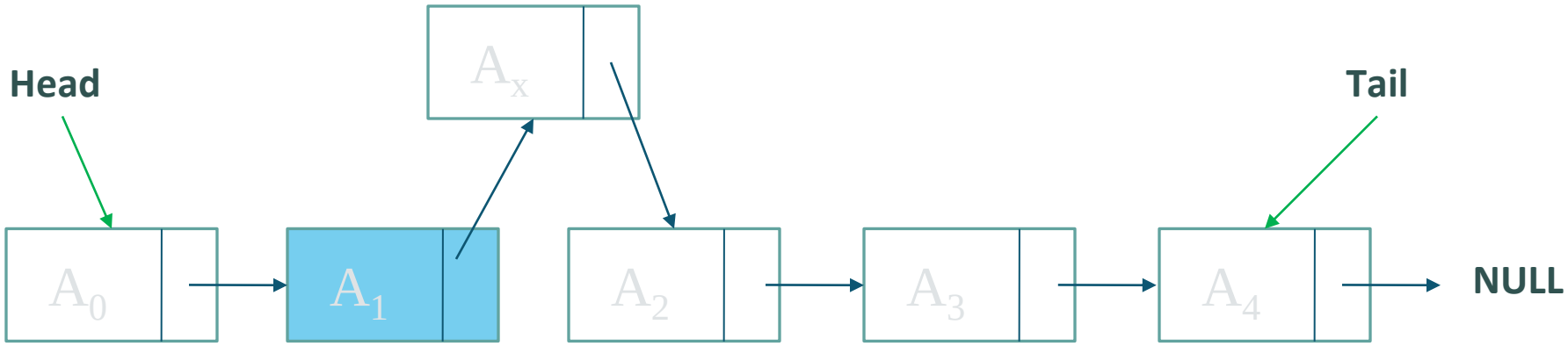
ADDING AN ELEMENT

Insert at (2)



ADDING AN ELEMENT

Insert at (2)



4 – INSERTAT FUNCTION: IMPLEMENTATION

LinkedList.h

```
class LinkedList{
        Node * head;
        Node * tail;
        int size;
public:
        // Methods & Const.'s
        LinkedList();
        void Append(float val);
        void InsertAt(int val, int pos);
};
```

4 – INSERTAT FUNCTION: IMPLEMENTATION

LinkedList.cpp

```
void LinkedList::InsertAt(int val, int pos)
{
assert(pos >= 0 && pos<size);
Node * tmp = head;
Node * newNode = new Node (val);
if (pos == 0)
{
   newNode->next = head;
   head = newNode;
}
```

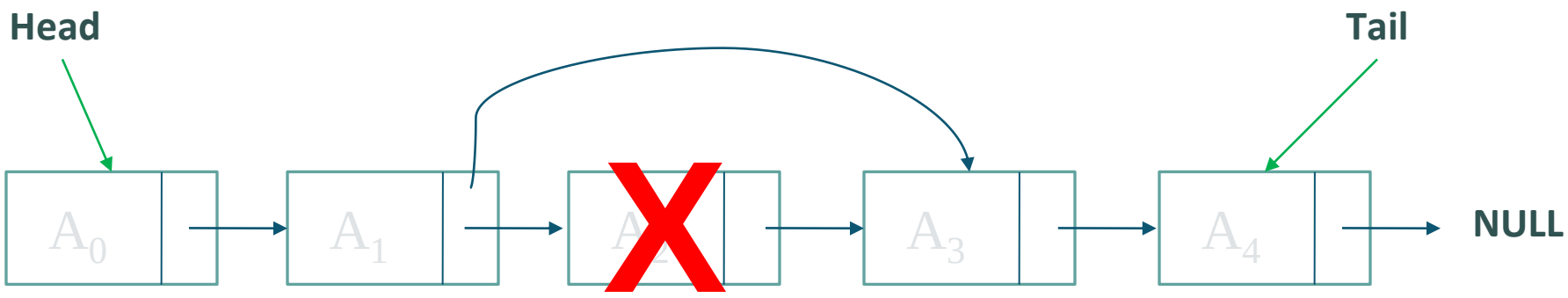
```
else
{
for (int i = 0; i<pos - 1; i++)
  tmp = tmp->next;
  newNode->next = tmp->next;
  tmp->next = newNode;
}
size++;
}
```

5 – DELETEAT FUNCTION

remove an element at specific position.

DELETING AN ELEMENT

Delete A_2



5 – DELETEAT FUNCTION: IMPLEMENTATION

LinkedList.h

```
class LinkedList{
        Node * head;
        Node * tail;
        int size;
public:
        // Methods & Const.'s
        LinkedList();
        void Append(float val);
void InsertAt(int val, int pos);
void DeleteAt(int pos);
};
```

5 – DELETEAT FUNCTION: IMPLEMENTATION

LinkedList.cpp

```
void LinkedList::DeleteAt(int pos)
{
assert(pos >= 0 && pos<size);
Node * tmp = head;
if (pos == 0)
{
head = head->next;
delete tmp;
}
```

```
else
{
for (int i = 0; i<pos - 1; i++)
tmp = tmp->next;
Node * del = tmp->next;
tmp->next = del->next;
delete del;
if (pos == size - 1)
tail = tmp;
}
size--;}
```

6 – MOVENODE FUNCTION

Write the C++ code for the function MoveNode(value, pos) which given a linked list, moves the node with value "value" to the position given by the variable "pos".

Note: write only the code of the function assuming that you already have the definition of the Linked List class with all its functions as implemented in the lecture

6 – MOVENODE FUNCTION: IMPLEMENTATION

LinkedList.h

```
class LinkedList{
        Node * head;
        Node * tail;
        int size;
public:
        LinkedList();
        void Append(float val);
        void InsertAt(int val, int pos);
        void DeleteAt(int pos);
        void MoveNode(int val, int pos);
```

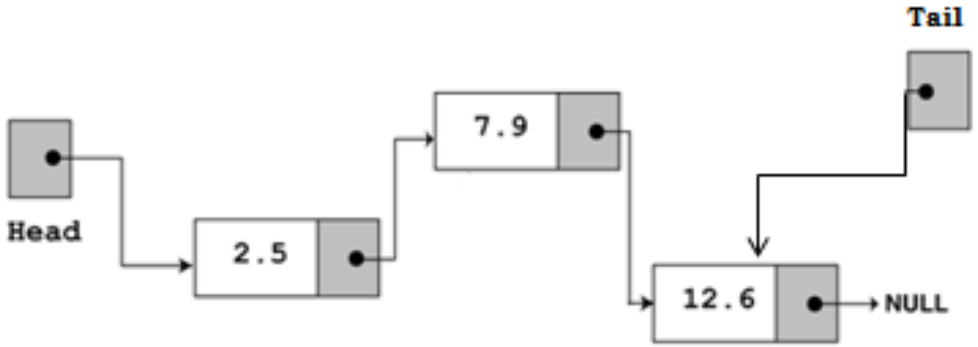
6 – MOVENODE FUNCTION: IMPLEMENTATION

LinkedList.cpp

```
void LinkedList ::MoveNode(int val, int pos)
{
  Node * tmp=head;
        int c=0;
        while(tmp!=NULL&& tmp->value!=val)
        {
                tmp=tmp->next;
                c++;
        }
        DeleteAt(c);
        InsertAt(pos, val);
}
```

7- DISPLAY FUNCTION

The list now contains these items in this form :



After Calling “display” fn, the output :

- 2.5
- 7.9
- 12.6

7- DISPLAY FUNCTION (PSEUDO-CODE)

Assign List head to node pointer

While node pointer is not **NULL**

- Display the value member of the node pointed to by node pointer.
- Assign node pointer to its own next member.

End While

7- DISPLAY FUNCTION: IMPLEMENTATION

LinkedList.h

```
class LinkedList{
public:
        Node * head;
        Node * tail;
        int size;
        LinkedList();
        void Append(float val);
        void InsertAt(int val, int pos);
        void DeleteAt(int pos);

        void Display();
};
```

LinkedList.cpp

```
…
…
void LinkedList::Display() {
        Node * temp = head;
        while (temp!=NULL) {
                cout<<temp->data;
                cout<<endl;
                temp = temp->next;
        }
}
```


thank
you