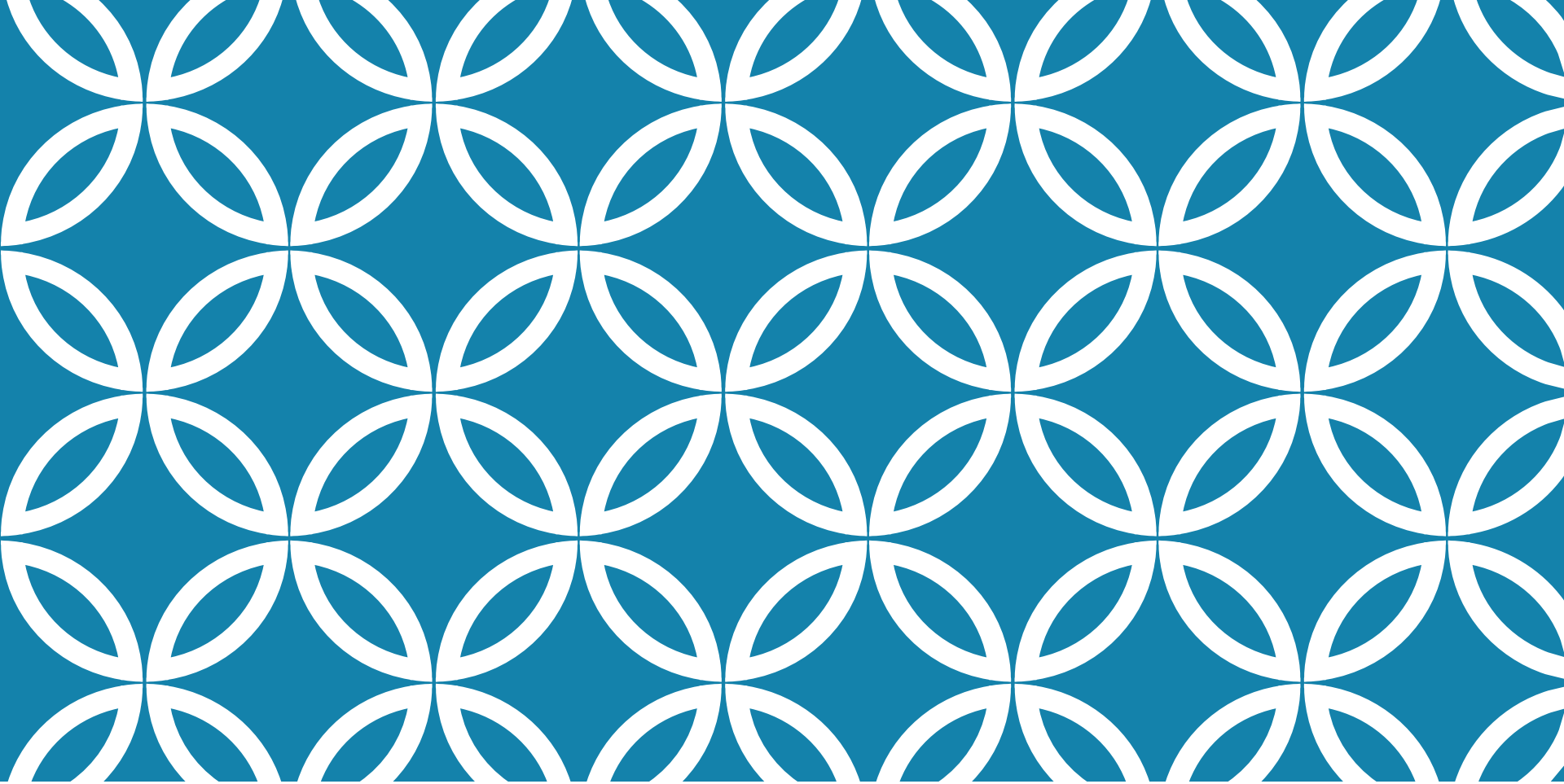


LAB 6
LINKEDSTACK

Data Structures
2021-2022

AGENDA

Class **StackNode**

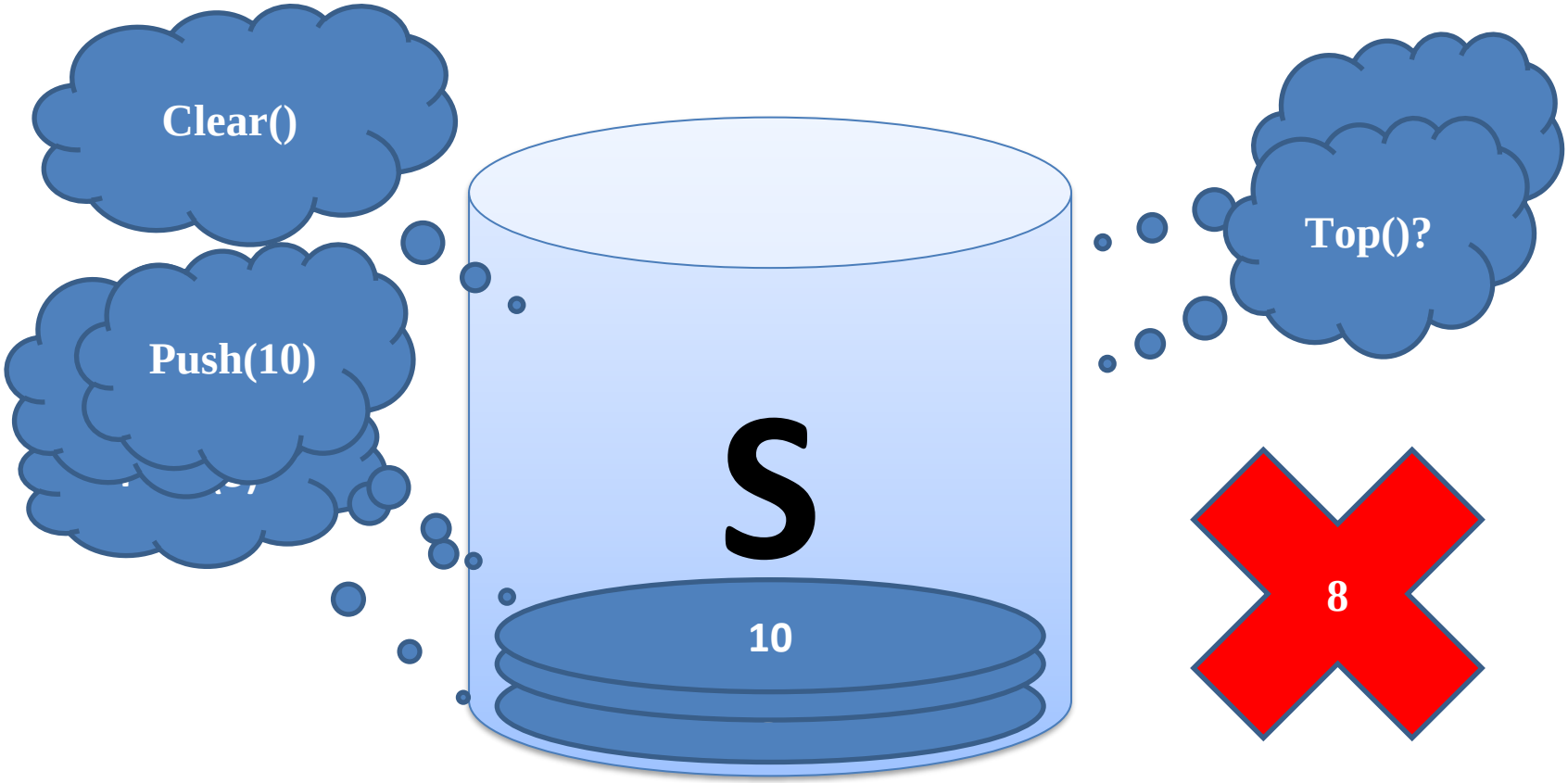
- Constructors

Class **LinkedStack**

- Constructor
- isEmpty()
- Push()
- Pop()
- Clear()
- Destructor

main() Function

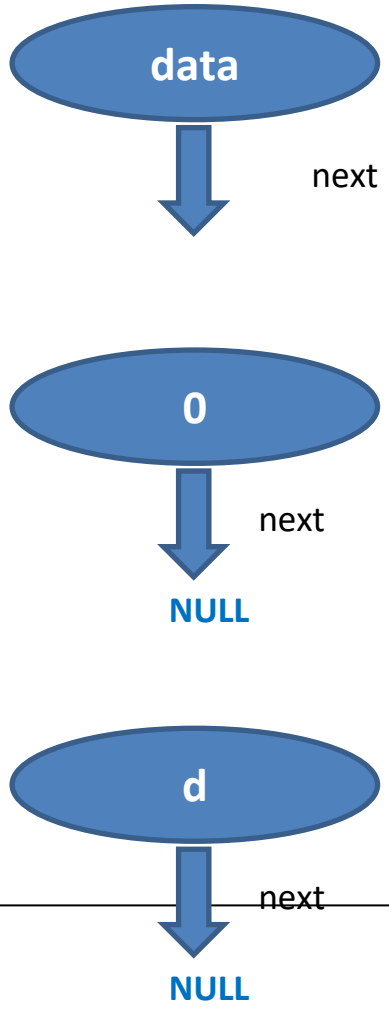
Stack Representation **LIFO**



StackNode class (Using template)

- Class (StackNode):

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

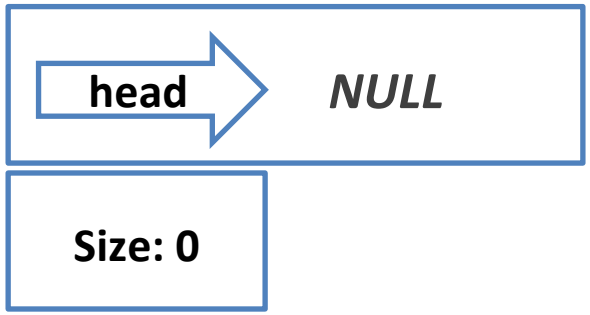


LinkedStack class (Using template)

- Class (LinkedStack), which:

1. **Has member variables of**
   - Head (top)- A pointer to the top of the stack
   - Size – Number of elements in the stack

2. **Has one constructor**
   - Default Constructor
     - Head : Null
     - Size : zero



StackNode (StackNode.h, StackNode.cpp)

```
//StackNode.h
template <class T>
class  StackNode
{
    public:
        T data;
        StackNode<T>*next;
        StackNode();
        StackNode(T);
};
```

```
//StackNode.cpp
#include "StackNode.h"
template <class T>
StackNode<T>::StackNode() {
    data = 0;
    next = NULL;
}

template <class T>
StackNode<T>::StackNode(T d) {
    data = d;
    next = NULL;
}
```

LinkedStack

(LinkedStack.h,LinkedStack.cpp)

```
//LinkedStack.h
template <class T>
class LinkedStack {
   int size;
   StackNode<T> *head;
 public:
     LinkedStack();
};
```

```
//LinkedStack.cpp
#include "LinkedStack.h"
template <class T>
LinkedStack<T>::LinkedStack() {
    size = 0;
     head = NULL; //head = 0;
}
```

LinkedStack: isEmpty() (LinkedStack.h)

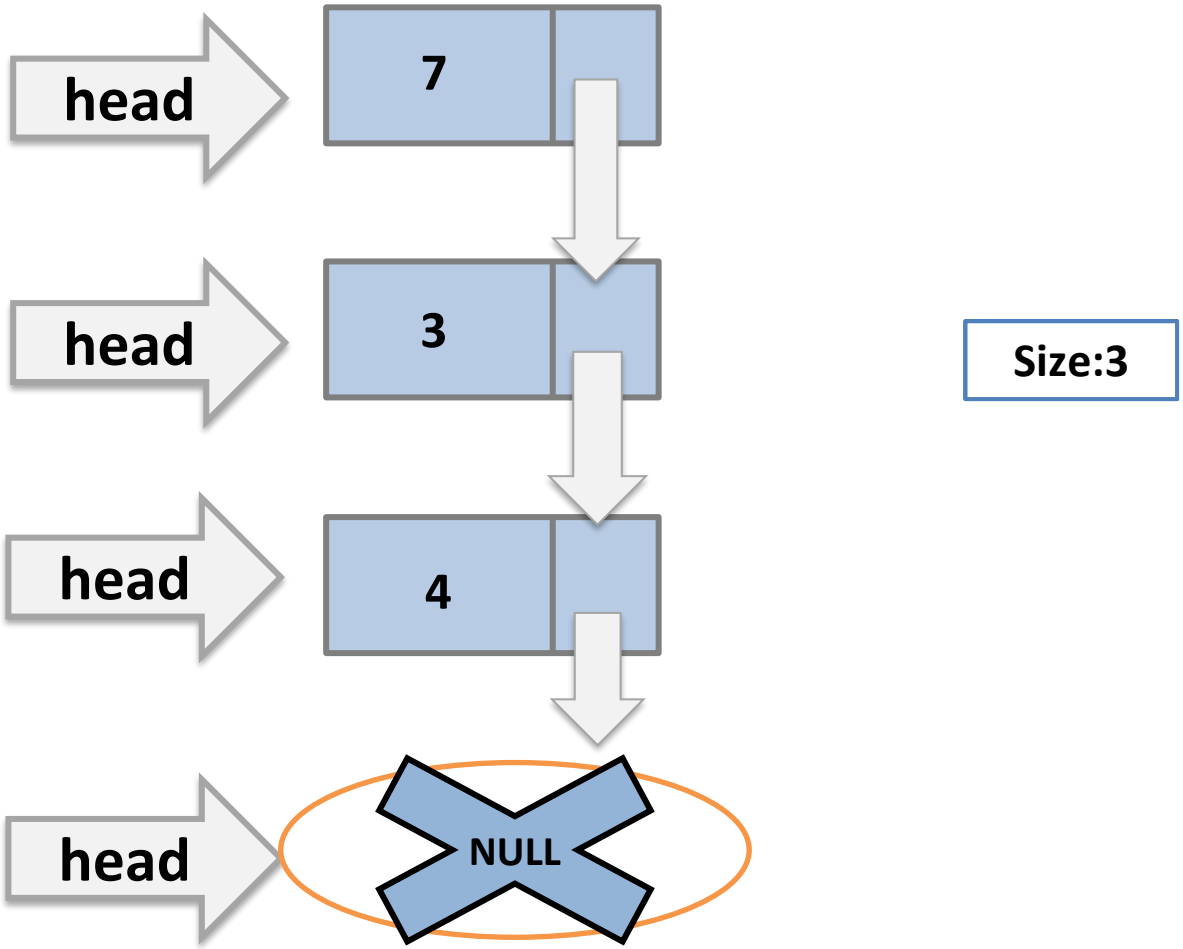
```
//LinkedStack.h
template <class T>
class LinkedStack {
  int size;
  StackNode<T> *head;
 public:
    LinkedStack();
    bool isEmpty();
};
```

LinkedStack: isEmpty()

(LinkedStack.h)

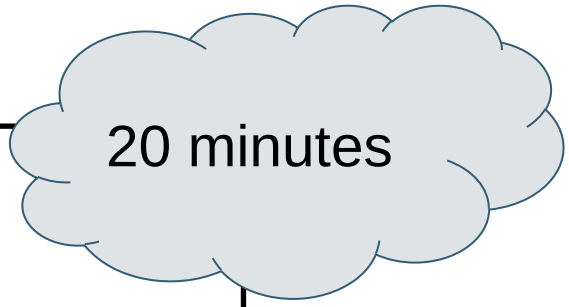
```
//LinkedStack.cpp
#include "LinkedStack.h"
template <class T>
LinkedStack<T>::LinkedStack() {
    size = 0;
    head = NULL; //head = 0;
}

template <class T>
bool LinkedStack<T>::isEmpty() {
    if(size==0)
        return true;
    else
        return false;
    //return(size==0);
}
```

LinkedStack (Push Mechanism)



Task 1: LinkedStack: Push()

(LinkedStack.h)

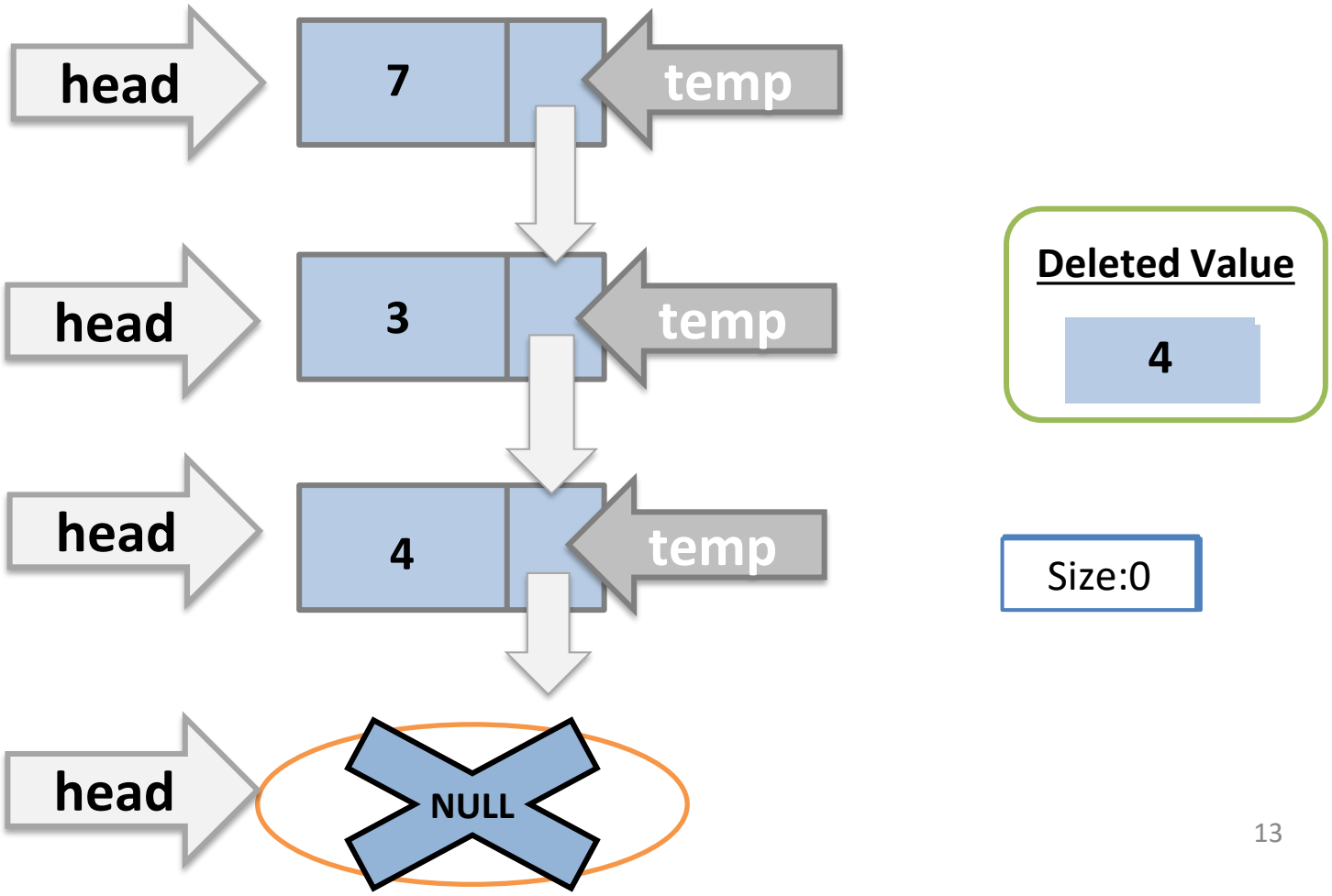



```
//LinkedStack.h
template <class T>
class LinkedStack {
 int size;
 StackNode<T> *head;
 public:
   LinkedStack();
   bool isEmpty();
   void push(T val);
};
```

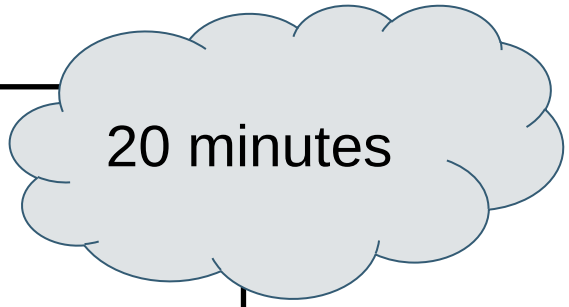
Task 1: LinkedStack Push()

```
//LinkedStack.cpp
template <class T>
void LinkedStack<T>::push(T val) {
  StackNode<T>* node = new StackNode<T>(val);
  node->next = head;
  head = node;
  size ++;
}
```

LinkedStack (Pop Mechanism)



Task 2: LinkedStack: Pop() (LinkedStack.h)




```
//LinkedStack.h
template <class T>
class LinkedStack {
 int size;
 StackNode<T> *head;
 public:
   LinkedStack();
   bool isEmpty();
   void push(T val);
   T pop();
};
```

Task 2: LinkedStack Pop()

```
template <class T>
T LinkedStack<T>::pop() {
        if(isEmpty()) {
          cout<<"You can't pop from an empty stack!!"<<endl;
          return 0;
        }
        StackNode<T> *tmp = head;
        head = head->next;
        T val = tmp->data;
        delete tmp;
        size--;
        return val;
}
```

LinkedStack Clear() & Destructor

```
//LinkedStack.h
template <class T>
class LinkedStack {
 int size;
 StackNode<T> *head;
 public:
   LinkedStack();
   ~LinkedStack();
   bool isEmpty();
   void push(T);
   T pop();
   void clear();
};
```

LinkedStack Clear() & Destructor

```
template <class T>
void LinkedStack<T>::clear()
   {
    StackNode<T> *tmp;
    while(head != NULL) {
       tmp = head;
       head = head->next;
       delete tmp;
    }
      size=0;
    //while(!isEmpty())
    //pop();
}
```

```
template <class T>
LinkedStack<T>::~LinkedStack()
{
     clear();
}
```

TESTING THE STACK

```
//Main.cpp
 #include "LinkedStack.cpp"
 void main() {

        LinkedStack <int> s;
        s.push(5);
        s.push(8);
        s.pop();
        s.push(10);
        if(s.isEmpty()) {
           cout<<"The LinkedStack is now empty!\n";
           s.push(35);
        }
 }
```


thank
you