SSL Outlab 6

Generated by Doxygen 1.8.17

1 Class Index 1

1.1 Class List 1

2 File Index 3

2.1 File List 3

3 Class Documentation 5

3.1 BinarySearchTree Class Reference 5

3.1.1 Detailed Description 5

3.1.2 Constructor & Destructor Documentation 6

3.1.2.1 BinarySearchTree() 6

3.1.3 Member Function Documentation 6

3.1.3.1 height() 6

3.1.3.2 insert() 6

3.1.3.3 traverse() 7

3.2 BSTNode Class Reference 7

3.2.1 Detailed Description 8

3.2.2 Constructor & Destructor Documentation 8

3.2.2.1 BSTNode() 8

3.3 DoublyLinkedList Class Reference 8

3.3.1 Detailed Description 9

3.3.2 Constructor & Destructor Documentation 9

3.3.2.1 DoublyLinkedList() 9

3.3.3 Member Function Documentation 9

3.3.3.1 insert() 9

3.3.3.2 printer() 10

3.3.3.3 reverse() 10

3.4 DoublyLinkedListNode Class Reference 11

3.4.1 Detailed Description 11

3.4.2 Constructor & Destructor Documentation 11

3.4.2.1 DoublyLinkedListNode() `[1/2]` 11

3.4.2.2 DoublyLinkedListNode() `[2/2]` 12

3.5 Heap Class Reference 12

3.5.1 Detailed Description 13

3.5.2 Constructor & Destructor Documentation 13

3.5.2.1 Heap() 13

3.5.3 Member Function Documentation 13

3.5.3.1 deleteMin() 13

3.5.3.2 Heapify() 13

3.5.3.3 insert() 14

3.5.3.4 left() 14

3.5.3.5 min() 14

3.5.3.6 parent() 15

3.5.3.7 right() 15
3.6 SinglyLinkedList Class Reference 15
3.6.1 Detailed Description 16
3.6.2 Constructor & Destructor Documentation 16
3.6.2.1 SinglyLinkedList() 16
3.6.3 Member Function Documentation 16
3.6.3.1 deleteVal() 17
3.6.3.2 find() 17
3.6.3.3 insert() 17
3.6.3.4 printer() 18
3.6.3.5 reverse() 18
3.7 SinglyLinkedListNode Class Reference 18
3.7.1 Detailed Description 19
3.7.2 Constructor & Destructor Documentation 19
3.7.2.1 SinglyLinkedListNode() [1/2] 19
3.7.2.2 SinglyLinkedListNode() [2/2] 19
3.8 Trie Class Reference 20
3.8.1 Detailed Description 20
3.8.2 Constructor & Destructor Documentation 20
3.8.2.1 Trie() 20
3.8.3 Member Function Documentation 21
3.8.3.1 checkPrefix() 21
3.8.3.2 countPrefix() 21
3.8.3.3 find() 21
3.8.3.4 insert() 22

4 File Documentation **23**
4.1 DSA.h File Reference 23

Index **25**

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

BinarySearchTree
Class for a Binary Search Tree 5
BSTNode
Node in a Binary Search tree 7
DoublyLinkedList
Class for a Doubly Linked List 8
DoublyLinkedListNode
Node in a Doubly Linked List 11
Heap
Class for a binary heap 12
SinglyLinkedList
Class for a Singly Linked List 15
SinglyLinkedListNode
Node in a singly linked list 18
Trie
Class for a suffix trie 20

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

DSA.h
This file contains 4 different types of data structures 23

Chapter 3

Class Documentation

3.1 BinarySearchTree Class Reference

Class for a Binary Search Tree.

```
#include <DSA.h>
```

Collaboration diagram for BinarySearchTree:

Public Types

- enum **order** { **PRE**, **IN**, **POST** }

Public Member Functions

- BinarySearchTree ()

  This is a constructor method to create a Binary search tree. Sets root to NULL.
- void insert (ll val)

  This is a member function to insert a new element. Inserts a new node with data as the element in it.
- void traverse (BSTNode ∗T, order tt)

  This is a printer function to print the tree in the traversal order given.
- ll height (BSTNode ∗T)

  This is a member function to get the height of the node.

Public Attributes

- BSTNode ∗ root

  Pointer to the root of the tree.

3.1.1 Detailed Description

Class for a Binary Search Tree.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 BinarySearchTree()

```
BinarySearchTree::BinarySearchTree ( )
```

This is a constructor method to create a Binary search tree. Sets root to NULL.

Class for a Binary Search Tree.

Parameters

in	*//*	val

3.1.3 Member Function Documentation

3.1.3.1 height()

```
ll BinarySearchTree::height (
             BSTNode * T )
```

This is a member function to get the height of the node.

Parameters

in	*BSTNode∗*	T
out	*//*	height

Returns

height of the node

3.1.3.2 insert()

```
void BinarySearchTree::insert (
             ll val )
```

This is a member function to insert a new element. Inserts a new node with data as the element in it.

Parameters

in	*ll*	data

Returns

NULL

3.1.3.3 traverse()

```
void BinarySearchTree::traverse (
            BSTNode * T,
            order tt )
```

This is a printer function to print the tree in the traversal order given.

Parameters

in	*BSTNode**	T
in	*order*	TT
out	*prints*	the binary search tree in the order given

Returns

NULL

The documentation for this class was generated from the following files:

- DSA.h
- DSA.cpp

3.2 BSTNode Class Reference

Node in a Binary Search tree.

```
#include <DSA.h>
```

Collaboration diagram for BSTNode:

Public Member Functions

- BSTNode (ll val)

  This is a constructor method to create a DoublyLinkedListNode. Sets info to val. Sets level to 0. Sets left to NULL. Sets right to NULL.

Public Attributes

- ll info

    Contains the element.
- ll level

    Contains level of the node in the tree.
- BSTNode ∗ left

    Pointer to the left node.
- BSTNode ∗ right

    Pointer to the right node.

3.2.1 Detailed Description

Node in a Binary Search tree.

3.2.2 Constructor & Destructor Documentation

3.2.2.1 BSTNode()

```
BSTNode::BSTNode (
              ll val )
```

This is a constructor method to create a DoublyLinkedListNode. Sets info to val. Sets level to 0. Sets left to NULL. Sets right to NULL.

Node in a Binary Search tree.

Parameters

in	ll	val

The documentation for this class was generated from the following files:

- DSA.h
- DSA.cpp

3.3 DoublyLinkedList Class Reference

Class for a Doubly Linked List.

```
#include <DSA.h>
```

Collaboration diagram for DoublyLinkedList:

Public Member Functions

- DoublyLinkedList ()

  This is a constructor method to create a SinglyLinkedList. Sets head to NULL. Sets tail to NULL.
- void insert (ll data)

  This is a member function to insert a new element. Inserts a new node with data as the element in it.
- void printer (string sep=", ")

  This is a printer function to print the values in the list.
- void reverse ()

  This is a member function to reverse the order of the list.

Public Attributes

- DoublyLinkedListNode ∗ head

  Pointer to head of the list.
- DoublyLinkedListNode ∗ tail

  Pointer to tail of the list.

3.3.1 Detailed Description

Class for a Doubly Linked List.

3.3.2 Constructor & Destructor Documentation

3.3.2.1 DoublyLinkedList()

```
DoublyLinkedList::DoublyLinkedList ( )
```

This is a constructor method to create a SinglyLinkedList. Sets head to NULL. Sets tail to NULL.

Parameters

in	*NULL*	

3.3.3 Member Function Documentation

3.3.3.1 insert()

```
void DoublyLinkedList::insert (
            ll data )
```

This is a member function to insert a new element. Inserts a new node with data as the element in it.

Parameters

in	*//*	data

Returns

NULL

3.3.3.2 printer()

```
void DoublyLinkedList::printer (
            string sep = ", " )
```

This is a printer function to print the values in the list.

Parameters

in	*NULL*	
out	*Prints*	the list

Returns

NULL

3.3.3.3 reverse()

```
void DoublyLinkedList::reverse ( )
```

This is a member function to reverse the order of the list.

Parameters

in	*NULL*	

Returns

NULL

The documentation for this class was generated from the following files:

- DSA.h
- DSA.cpp

3.4 DoublyLinkedListNode Class Reference

Node in a Doubly Linked List.

```
#include <DSA.h>
```

Collaboration diagram for DoublyLinkedListNode:

Public Member Functions

- DoublyLinkedListNode ()

  This is a constructor method to create a DoublyLinkedListNode. Sets data to -1. Sets next to NULL. Sets prev to NULL.
- DoublyLinkedListNode (ll val)

  This is a constructor method to create a DoublyLinkedListNode. Sets data to val. Sets next to NULL. Sets prev to NULL.

Public Attributes

- ll data

  Data in the node.
- DoublyLinkedListNode ∗ next

  Pointer to next node.
- DoublyLinkedListNode ∗ prev

  Pointer to the previous node.

3.4.1 Detailed Description

Node in a Doubly Linked List.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 DoublyLinkedListNode() [1/2]

```
DoublyLinkedListNode::DoublyLinkedListNode ( )
```

This is a constructor method to create a DoublyLinkedListNode. Sets data to -1. Sets next to NULL. Sets prev to NULL.

Parameters

in	*NULL*	

3.4.2.2 DoublyLinkedListNode() [2/2]

```
DoublyLinkedListNode::DoublyLinkedListNode (
            ll val )
```

This is a constructor method to create a DoublyLinkedListNode. Sets data to val. Sets next to NULL. Sets prev to NULL.

Parameters

in	*ll*	val

The documentation for this class was generated from the following files:

- DSA.h
- DSA.cpp

3.5 Heap Class Reference

Class for a binary heap.

```
#include <DSA.h>
```

Public Member Functions

- Heap (int cap)

  This is a constructor method to create a Heap. Initializes a new array with cap as the number of elements Sets n to 0.
- int parent (int i)

  This is a member function to find the parent of a node.
- int left (ll i)

  This is a member function to find the left child of a node.
- int right (ll i)

  This is a member function to find the right child of a node.
- void insert (int val)

  This is a member function to insert a new element. Inserts a new node with data as the element in it.
- int min ()

  This is a member function to find the minimum element in a heap.
- void Heapify (int root)

  This is to make it into a heap when both the left and right subheaps satisfy the heap property but not the whole heap.
- void deleteMin ()

  This is to delete the minimum element in a heap.

Public Attributes

- int cap

  Maximum number of elements in the heap.

3.5.1 Detailed Description

Class for a binary heap.

3.5.2 Constructor & Destructor Documentation

3.5.2.1 Heap()

```
Heap::Heap (
             int cap )
```

This is a constructor method to create a Heap. Initializes a new array with cap as the number of elements Sets n to 0.

Parameters

in	*int*	cap

3.5.3 Member Function Documentation

3.5.3.1 deleteMin()

```
void Heap::deleteMin ( )
```

This is to delete the minimum element in a heap.

Returns

NULL

3.5.3.2 Heapify()

```
void Heap::Heapify (
             int root )
```

This is to make it into a heap when both the left and right subheaps satisfy the heap property but not the whole heap.

Parameters

`in`	*int*	root

Returns

NULL

3.5.3.3 insert()

```
void Heap::insert (
             int val )
```

This is a member function to insert a new element. Inserts a new node with data as the element in it.

Parameters

`in`	*int*	val

Returns

NULL

3.5.3.4 left()

```
int Heap::left (
             ll i )
```

This is a member function to find the left child of a node.

Parameters

`in`	*int*	i
`out`	*int*	2∗i+1

Returns

The index of the left child of element

3.5.3.5 min()

```
int Heap::min ( )
```

This is a member function to find the minimum element in a heap.

Returns

The element with the minimum value

3.5.3.6 parent()

```
int Heap::parent (
             int i )
```

This is a member function to find the parent of a node.

Parameters

in	*int*	i
out	*int*	(i-1)/2

Returns

The index of the parent of element

3.5.3.7 right()

```
int Heap::right (
             ll i )
```

This is a member function to find the right child of a node.

Parameters

in	*int*	i
out	*int*	2∗(i+1)

Returns

The index of the left child of element

The documentation for this class was generated from the following files:

- DSA.h
- DSA.cpp

3.6 SinglyLinkedList Class Reference

Class for a Singly Linked List.

`#include <DSA.h>`

Collaboration diagram for SinglyLinkedList:

Public Member Functions

- SinglyLinkedList ()

  This is a constructor method to create a SinglyLinkedList. Sets head to NULL. Sets tail to NULL.
- void insert (ll data)

  This is a member function to insert a new element. Inserts a new node with data as the element in it.
- SinglyLinkedListNode ∗ find (ll data)

  This is a member function to find an element.
- bool deleteVal (ll data)

  This is a member function to delete an element.
- void printer (string sep=", ")

  This is a printer function to print the values in the list.
- void reverse ()

  This is a member function to reverse the order of the list.

Public Attributes

- SinglyLinkedListNode ∗ head

  Pointer to the head of the list.
- SinglyLinkedListNode ∗ tail

  Pointer to the tail of the list.

3.6.1 Detailed Description

Class for a Singly Linked List.

3.6.2 Constructor & Destructor Documentation

3.6.2.1 SinglyLinkedList()

```
SinglyLinkedList::SinglyLinkedList ( )
```

This is a constructor method to create a SinglyLinkedList. Sets head to NULL. Sets tail to NULL.

Parameters

in	*NULL*	

3.6.3 Member Function Documentation

3.6.3.1 deleteVal()

```
bool SinglyLinkedList::deleteVal (
             ll data )
```

This is a member function to delete an element.

Parameters

in	*ll*	data
out	*bool*	

Returns

true if successfully deleted else false

3.6.3.2 find()

```
SinglyLinkedListNode * SinglyLinkedList::find (
             ll data )
```

This is a member function to find an element.

Parameters

in	*ll*	data
out	*ll*	prev

Returns

NULL if not found else returns pointer to the node containing the element

3.6.3.3 insert()

```
void SinglyLinkedList::insert (
             ll data )
```

This is a member function to insert a new element. Inserts a new node with data as the element in it.

Parameters

in	*ll*	data

3.6.3.4 printer()

```
void SinglyLinkedList::printer (
            string sep = ", " )
```

This is a printer function to print the values in the list.

Parameters

in	*NULL*	
out	*Prints*	the list

Returns

NULL

3.6.3.5 reverse()

```
void SinglyLinkedList::reverse ( )
```

This is a member function to reverse the order of the list.

Parameters

in	*NULL*	

Returns

NULL

The documentation for this class was generated from the following files:

- DSA.h
- DSA.cpp

3.7 SinglyLinkedListNode Class Reference

Node in a singly linked list.

```
#include <DSA.h>
```

Collaboration diagram for SinglyLinkedListNode:

Public Member Functions

- SinglyLinkedListNode ()

  This is a constructor method to create a SinglyLinkedListNode. Sets data to -1. Sets next to NULL.
- SinglyLinkedListNode (ll val)

  This is a constructor method to create a SinglyLinkedListNode. Sets data to val. Sets next to NULL.

Public Attributes

- ll data

  Data stored in the node.
- SinglyLinkedListNode ∗ next

  Pointer to the next node.

3.7.1 Detailed Description

Node in a singly linked list.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 SinglyLinkedListNode() `[1/2]`

```
SinglyLinkedListNode::SinglyLinkedListNode ( )
```

This is a constructor method to create a SinglyLinkedListNode. Sets data to -1. Sets next to NULL.

Parameters

in	*NULL*	

3.7.2.2 SinglyLinkedListNode() `[2/2]`

```
SinglyLinkedListNode::SinglyLinkedListNode (
            ll val )
```

This is a constructor method to create a SinglyLinkedListNode. Sets data to val. Sets next to NULL.

Parameters

in	*ll*	val

The documentation for this class was generated from the following files:

- DSA.h
- DSA.cpp

3.8 Trie Class Reference

Class for a suffix trie.

```
#include <DSA.h>
```

Public Member Functions

- Trie ()
- bool find (Trie ∗T, char c)

  This is a member function to find an element.
- void insert (string s)

  This is a member function to insert a new element. Inserts a new node with data as the element in it.
- bool checkPrefix (string s)

  This is a member function to check if a prefix is present in the trie.
- ll countPrefix (string s)

  This is a member function to get the number of count of matches of a prefix in the trie.

Public Attributes

- ll count

  Keeps count of nodes below it.
- map< char, Trie ∗ > nodes

  Node in a suffix trie.

3.8.1 Detailed Description

Class for a suffix trie.

3.8.2 Constructor & Destructor Documentation

3.8.2.1 Trie()

```
Trie::Trie ( )
```

This is a constructor method to create a Suffix Trie. Sets count to 0. Sets nodes to empty map.

Parameters

in	*NULL*	

3.8.3 Member Function Documentation

3.8.3.1 checkPrefix()

```
bool Trie::checkPrefix (
            string s )
```

This is a member function to check if a prefix is present in the trie.

Parameters

in	*string*	s

Returns

true if found else false

3.8.3.2 countPrefix()

```
ll Trie::countPrefix (
            string s )
```

This is a member function to get the number of count of matches of a prefix in the trie.

Parameters

in	*string*	s
out	*ll*	countprefix

Returns

number of matches

3.8.3.3 find()

```
bool Trie::find (
            Trie * T,
            char c )
```

This is a member function to find an element.

Parameters

in	*Trie∗*	T
in	*ll*	char c
out	*bool*	

Returns

true if found else false

3.8.3.4 insert()

```
void Trie::insert (
            string s )
```

This is a member function to insert a new element. Inserts a new node with data as the element in it.

Parameters

in	*string*	s

Returns

NULL

The documentation for this class was generated from the following files:

- DSA.h
- DSA.cpp

Chapter 4

File Documentation

4.1 DSA.h File Reference

This file contains 4 different types of data structures.

```
#include <bits/stdc++.h>
```

Include dependency graph for DSA.h:

Index

BinarySearchTree, 5
- BinarySearchTree, 6
- height, 6
- insert, 6
- traverse, 7

BSTNode, 7
- BSTNode, 8

checkPrefix
- Trie, 21

countPrefix
- Trie, 21

deleteMin
- Heap, 13

deleteVal
- SinglyLinkedList, 16

DoublyLinkedList, 8
- DoublyLinkedList, 9
- insert, 9
- printer, 10
- reverse, 10

DoublyLinkedListNode, 11
- DoublyLinkedListNode, 11

DSA.h, 23

find
- SinglyLinkedList, 17
- Trie, 21

Heap, 12
- deleteMin, 13
- Heap, 13
- Heapify, 13
- insert, 14
- left, 14
- min, 14
- parent, 15
- right, 15

Heapify
- Heap, 13

height
- BinarySearchTree, 6

insert
- BinarySearchTree, 6
- DoublyLinkedList, 9
- Heap, 14
- SinglyLinkedList, 17
- Trie, 22

left
- Heap, 14

min
- Heap, 14

parent
- Heap, 15

printer
- DoublyLinkedList, 10
- SinglyLinkedList, 17

reverse
- DoublyLinkedList, 10
- SinglyLinkedList, 18

right
- Heap, 15

SinglyLinkedList, 15
- deleteVal, 16
- find, 17
- insert, 17
- printer, 17
- reverse, 18
- SinglyLinkedList, 16

SinglyLinkedListNode, 18
- SinglyLinkedListNode, 19

traverse
- BinarySearchTree, 7

Trie, 20
- checkPrefix, 21
- countPrefix, 21
- find, 21
- insert, 22
- Trie, 20