## SSL Inlab 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	. 5
3.1.2.1 BinarySearchTree()	. 5
3.1.3 Member Function Documentation	. 6
3.1.3.1 height()	. 6
3.1.3.2 insert()	. 6
3.1.3.3 traverse()	. 6
3.2 BSTNode Class Reference	. 7
3.2.1 Detailed Description	. 7
3.2.2 Constructor & Destructor Documentation	. 7
3.2.2.1 BSTNode()	. 7
3.3 DoublyLinkedList Class Reference	. 8
3.3.1 Detailed Description	. 8
3.3.2 Constructor & Destructor Documentation	. 8
3.3.2.1 DoublyLinkedList()	. 8
3.3.3 Member Function Documentation	. 9
3.3.3.1 insert()	. 9
3.3.3.2 printer()	. 9
3.3.3.3 reverse()	. 9
3.4 DoublyLinkedListNode Class Reference	. 10
3.4.1 Detailed Description	
3.4.2 Constructor & Destructor Documentation	. 10
<b>3.4.2.1 DoublyLinkedListNode()</b> [1/2]	. 10
<b>3.4.2.2</b> DoublyLinkedListNode() [2/2]	. 11
3.5 Heap Class Reference	
3.5.1 Detailed Description	. 12
3.5.2 Constructor & Destructor Documentation	
3.5.2.1 Heap()	
3.5.3 Member Function Documentation	
3.5.3.1 deleteMin()	
3.5.3.2 Heapify()	
3.5.3.3 insert()	
3.5.3.4 left()	
3.5.3.5 min()	
3.5.3.6 parent()	
$\cdot$	

3.5.3.7 right()	14
3.6 SinglyLinkedList Class Reference	14
3.6.1 Detailed Description	15
3.6.2 Constructor & Destructor Documentation	15
3.6.2.1 SinglyLinkedList()	15
3.6.3 Member Function Documentation	15
3.6.3.1 deleteVal()	16
3.6.3.2 find()	16
3.6.3.3 insert()	16
3.6.3.4 printer()	17
3.6.3.5 reverse()	17
3.7 SinglyLinkedListNode Class Reference	17
3.7.1 Detailed Description	18
3.7.2 Constructor & Destructor Documentation	18
<b>3.7.2.1 SinglyLinkedListNode()</b> [1/2]	18
<b>3.7.2.2</b> SinglyLinkedListNode() [2/2]	18
3.8 Trie Class Reference	18
3.8.1 Detailed Description	19
3.8.2 Constructor & Destructor Documentation	19
3.8.2.1 Trie()	19
3.8.3 Member Function Documentation	19
3.8.3.1 checkPrefix()	19
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.cpp File Reference	23
Index	25

# **Chapter 1**

# **Class Index**

## 1.1 Class List

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

2 Class Index

# **Chapter 2**

# File Index

# 2.1 File List

Н	ere	is	а	list	of	all	documented	files	with	brief	descriptions:	
---	-----	----	---	------	----	-----	------------	-------	------	-------	---------------	--

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

File Index

# **Chapter 3**

# **Class Documentation**

## 3.1 BinarySearchTree Class Reference

Class for a Binary Search Tree.

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 (Il 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.

II 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 ( ) [inline]

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

#### **Parameters**

in		val

## 3.1.3 Member Function Documentation

## 3.1.3.1 height()

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

#### **Parameters**

in	BSTNode*	Т
out	//	height

#### Returns

height of the node

#### 3.1.3.2 insert()

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

## 3.1.3.3 traverse()

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

#### **Parameters**

in	BSTNode*	Т
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 file:

• DSA.cpp

## 3.2 BSTNode Class Reference

Node in a Binary Search tree.

Collaboration diagram for BSTNode:

#### **Public Member Functions**

· BSTNode (II 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**

• Il info

Contains the element.

· II 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

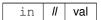
Node in a Binary Search tree.

#### 3.2.2 Constructor & Destructor Documentation

### 3.2.2.1 BSTNode()

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.

#### **Parameters**



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

• DSA.cpp

## 3.3 DoublyLinkedList Class Reference

Class for a Doubly Linked List.

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 (II 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 ( ) [inline]

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

#### **Parameters**

#### 3.3.3 Member Function Documentation

## 3.3.3.1 insert()

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

#### **Parameters**

in	11	data

#### Returns

NULL

## 3.3.3.2 printer()

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 ( ) [inline]
```

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

#### **Parameters**

in <b>NULL</b>
----------------

Returns

**NULL** 

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

• DSA.cpp

## 3.4 DoublyLinkedListNode Class Reference

Node in a Doubly Linked List.

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 (II val)

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

#### **Public Attributes**

• II 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 ( ) [inline]

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

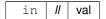
#### **Parameters**

in	NULL	
----	------	--

#### 3.4.2.2 DoublyLinkedListNode() [2/2]

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

#### **Parameters**



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

• DSA.cpp

## 3.5 Heap Class Reference

Class for a binary heap.

#### **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 (II i)

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

• int right (II 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()

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 ( ) [inline]
```

This is to delete the minimum element in a heap.

Returns

NULL

#### 3.5.3.2 Heapify()

```
void Heap::Heapify (
                int root ) [inline]
```

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 ) [inline]
```

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

#### **Parameters**

III   IIII   Vai
------------------

## Returns

NULL

## 3.5.3.4 left()

```
int Heap::left ( \label{eq:interpolation} \mbox{ll $i$ ) [inline]}
```

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 ( ) [inline]
```

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 ( \quad \text{int } i \text{ ) } \quad [\text{inline}]
```

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 ( \label{eq:local_local_local} \texttt{ll} \ \ i \ ) \quad [inline]
```

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 file:

• DSA.cpp

## 3.6 SinglyLinkedList Class Reference

Class for a Singly Linked List.

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 (II data)

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

SinglyLinkedListNode \* find (II data)

This is a member function to find an element.

• bool deleteVal (II 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.

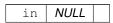
### 3.6.2 Constructor & Destructor Documentation

## 3.6.2.1 SinglyLinkedList()

```
SinglyLinkedList::SinglyLinkedList ( ) [inline]
```

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

#### **Parameters**



## 3.6.3 Member Function Documentation

## 3.6.3.1 deleteVal()

This is a member function to delete an element.

#### **Parameters**

in	11	data
out	bool	

#### Returns

true if successfully deleted else false

## 3.6.3.2 find()

This is a member function to find an element.

#### **Parameters**

in		data
out	11	prev

#### Returns

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

## 3.6.3.3 insert()

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

#### **Parameters**

in	11	data

#### 3.6.3.4 printer()

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 ( ) [inline]
```

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 file:

• DSA.cpp

## 3.7 SinglyLinkedListNode Class Reference

Node in a Singly Linked List.

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 (II val)

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

## **Public Attributes**

• Il 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 ( ) [inline]
```

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

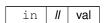
#### **Parameters**

```
in NULL
```

## 3.7.2.2 SinglyLinkedListNode() [2/2]

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

#### **Parameters**



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

• DSA.cpp

## 3.8 Trie Class Reference

Class for a suffix trie.

3.8 Trie Class Reference 19

#### **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.

• Il countPrefix (string s)

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

#### **Public Attributes**

Il 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 ( ) [inline]
```

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()

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

3.8 Trie Class Reference 21

## **Parameters**

in	string	s
----	--------	---

#### Returns

true if found else false

#### 3.8.3.2 countPrefix()

```
ll Trie::countPrefix ( string \ s \ ) \quad [inline]
```

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

#### **Parameters**

in	string	S
out	<i>II</i>	countprefix

#### Returns

number of matches

## 3.8.3.3 find()

This is a member function to find an element.

## **Parameters**

in	<i>Trie</i> *	T
in	11	char c
out	bool	

## Returns

true if found else false

## 3.8.3.4 insert()

```
void Trie::insert ( {\tt string}\ s\ ) \quad [{\tt inline}]
```

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 file:

• DSA.cpp

# **Chapter 4**

# **File Documentation**

## 4.1 DSA.cpp File Reference

This file contains 4 different types of data structures.

#include <bits/stdc++.h>
Include dependency graph for DSA.cpp:

24 File Documentation

# Index

Bina	rySearchTree, 5 BinarySearchTree, 5	left	Heap, 13
	height, 6		
	insert, 6	min	
	traverse, 6		Heap, 13
BST	Node, 7	naro	unt.
	BSTNode, 7	pare	Heap, 14
	LD C	print	•
cnec	kPrefix	P	DoublyLinkedList, 9
00110	Trie, 19 htPrefix		SinglyLinkedList, 16
Cour	Trie, 21		,
	IIIe, 21	reve	rse
delet	teMin		DoublyLinkedList, 9
40.0	Heap, 12		SinglyLinkedList, 17
delet	teVal	right	
	SinglyLinkedList, 15		Heap, 14
Doub	olyLinkedList, 8	Cina	ılyLinkedList, 14
	DoublyLinkedList, 8	Sing	deleteVal, 15
	insert, 9		find, 16
	printer, 9		insert, 16
	reverse, 9		printer, 16
Doub	olyLinkedListNode, 10		reverse, 17
	DoublyLinkedListNode, 10, 11		SinglyLinkedList, 15
DSA	.cpp, 23		
20,1	···	Sing	ılyLinkedListNode, 17
	······································	Sing	llyLinkedListNode, 17 SinglyLinkedListNode, 18
find		Sing	llyLinkedListNode, 17 SinglyLinkedListNode, 18
	SinglyLinkedList, 16	Sing	SinglyLinkedListNode, 18 erse
		trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6
find	SinglyLinkedList, 16 Trie, 21		SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18
	SinglyLinkedList, 16 Trie, 21  p, 11	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19
find	SinglyLinkedList, 16 Trie, 21  p, 11 deleteMin, 12	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19 countPrefix, 21
find	SinglyLinkedList, 16 Trie, 21  p, 11 deleteMin, 12 Heap, 12	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19 countPrefix, 21 find, 21
find	SinglyLinkedList, 16 Trie, 21  o, 11 deleteMin, 12 Heap, 12 Heapify, 12	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19 countPrefix, 21 find, 21 insert, 21
find	SinglyLinkedList, 16 Trie, 21  p, 11 deleteMin, 12 Heap, 12 Heapify, 12 insert, 13	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19 countPrefix, 21 find, 21
find	SinglyLinkedList, 16 Trie, 21  o, 11 deleteMin, 12 Heap, 12 Heapify, 12	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19 countPrefix, 21 find, 21 insert, 21
find	SinglyLinkedList, 16 Trie, 21  D, 11 deleteMin, 12 Heap, 12 Heapify, 12 insert, 13 left, 13	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19 countPrefix, 21 find, 21 insert, 21
find	SinglyLinkedList, 16 Trie, 21  D, 11 deleteMin, 12 Heap, 12 Heapify, 12 insert, 13 left, 13 min, 13	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19 countPrefix, 21 find, 21 insert, 21
find	SinglyLinkedList, 16 Trie, 21  o, 11 deleteMin, 12 Heap, 12 Heapify, 12 insert, 13 left, 13 min, 13 parent, 14 right, 14	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19 countPrefix, 21 find, 21 insert, 21
Hear	SinglyLinkedList, 16 Trie, 21  2, 11  deleteMin, 12 Heap, 12 Heapify, 12 insert, 13 left, 13 min, 13 parent, 14 right, 14 bify Heap, 12	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19 countPrefix, 21 find, 21 insert, 21
find	SinglyLinkedList, 16 Trie, 21  D, 11 deleteMin, 12 Heap, 12 Heapify, 12 insert, 13 left, 13 min, 13 parent, 14 right, 14 bify Heap, 12 ht	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19 countPrefix, 21 find, 21 insert, 21
Hear	SinglyLinkedList, 16 Trie, 21  2, 11  deleteMin, 12 Heap, 12 Heapify, 12 insert, 13 left, 13 min, 13 parent, 14 right, 14 bify Heap, 12	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19 countPrefix, 21 find, 21 insert, 21
Hear	SinglyLinkedList, 16 Trie, 21  D, 11 deleteMin, 12 Heap, 12 Heapify, 12 insert, 13 left, 13 min, 13 parent, 14 right, 14 bify Heap, 12 ht BinarySearchTree, 6	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19 countPrefix, 21 find, 21 insert, 21
Hear Hear heigh	SinglyLinkedList, 16 Trie, 21  D, 11 deleteMin, 12 Heap, 12 Heapify, 12 insert, 13 left, 13 min, 13 parent, 14 right, 14 bify Heap, 12 ht BinarySearchTree, 6	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19 countPrefix, 21 find, 21 insert, 21
Hear Hear heigh	SinglyLinkedList, 16 Trie, 21  D, 11 deleteMin, 12 Heap, 12 Heapify, 12 insert, 13 left, 13 min, 13 parent, 14 right, 14 bify Heap, 12 ht BinarySearchTree, 6	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19 countPrefix, 21 find, 21 insert, 21
Hear Hear heigh	SinglyLinkedList, 16 Trie, 21  D, 11 deleteMin, 12 Heap, 12 Heapify, 12 insert, 13 left, 13 min, 13 parent, 14 right, 14 bify Heap, 12 ht BinarySearchTree, 6	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19 countPrefix, 21 find, 21 insert, 21
Hear Hear heigh	SinglyLinkedList, 16 Trie, 21  20, 11 deleteMin, 12 Heap, 12 Heapify, 12 insert, 13 left, 13 min, 13 parent, 14 right, 14 bify Heap, 12 ht BinarySearchTree, 6  DoublyLinkedList, 9	trave	SinglyLinkedListNode, 18 erse BinarySearchTree, 6 18 checkPrefix, 19 countPrefix, 21 find, 21 insert, 21