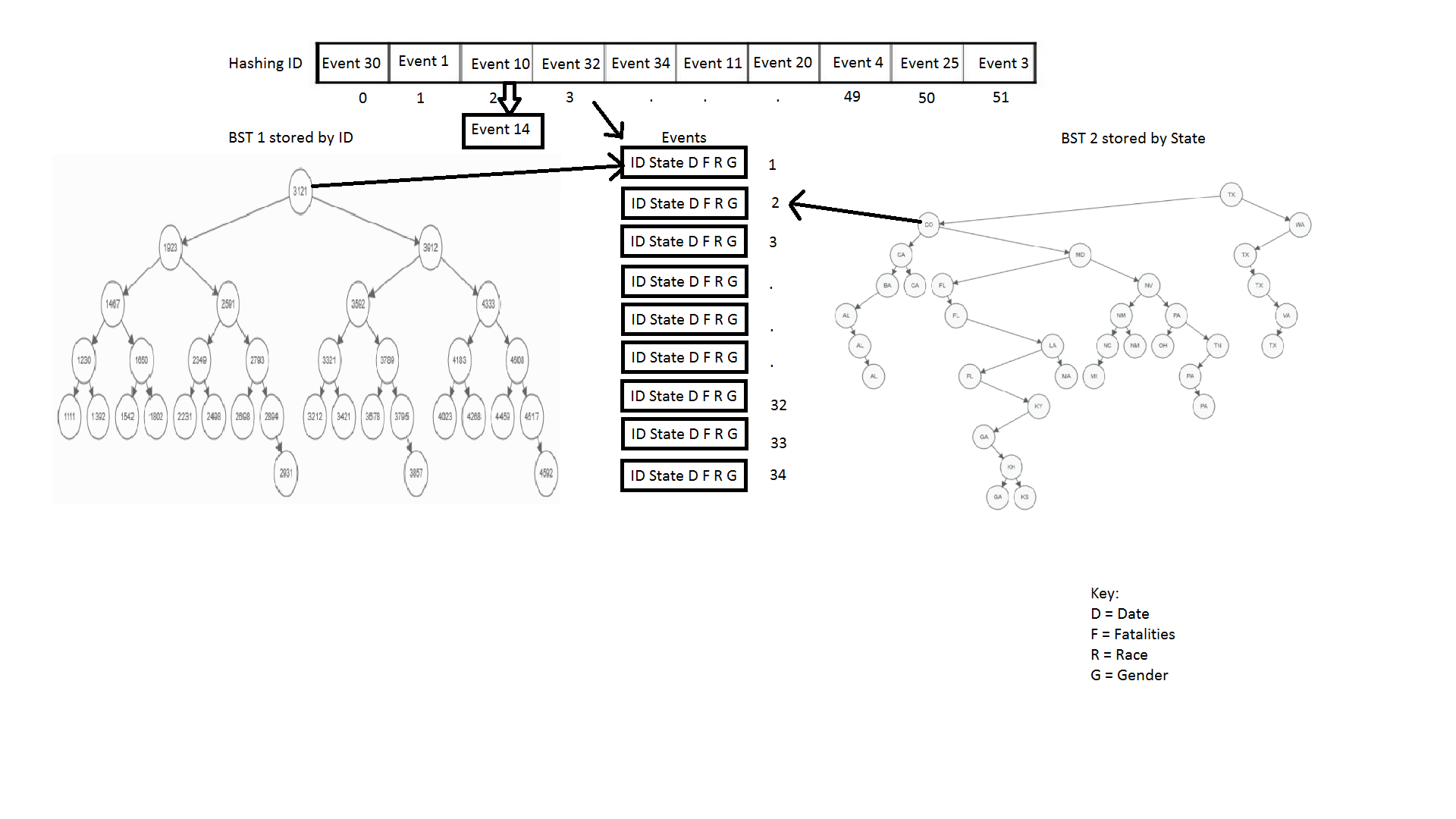
Team #7 Weekly Report No. 3

Project Title: Gun Violence in America

**Data Structure Diagram**



**UML Diagrams**

|  |
| --- |
| Incidents |
| * ID : string * state : string * date : string * fatalities : int * race : string * gender : string |
| * Incidents() : * Incidents( sID : string, sState : string, sDate : string , sFatalities : int, sRace : string, sGender : string) : * setID(sID : string) : void * setState(sState : string) : void * setDate(sDate : string) : void * setFatalities(sFatalities : int) : void * setRace(sRace : string) : void * setGender(sGender : string) : void * getID() const : string * getState() const : string * getDate() const : int * getFatalities() const : int * getRace() const : string * getGender() const : string |

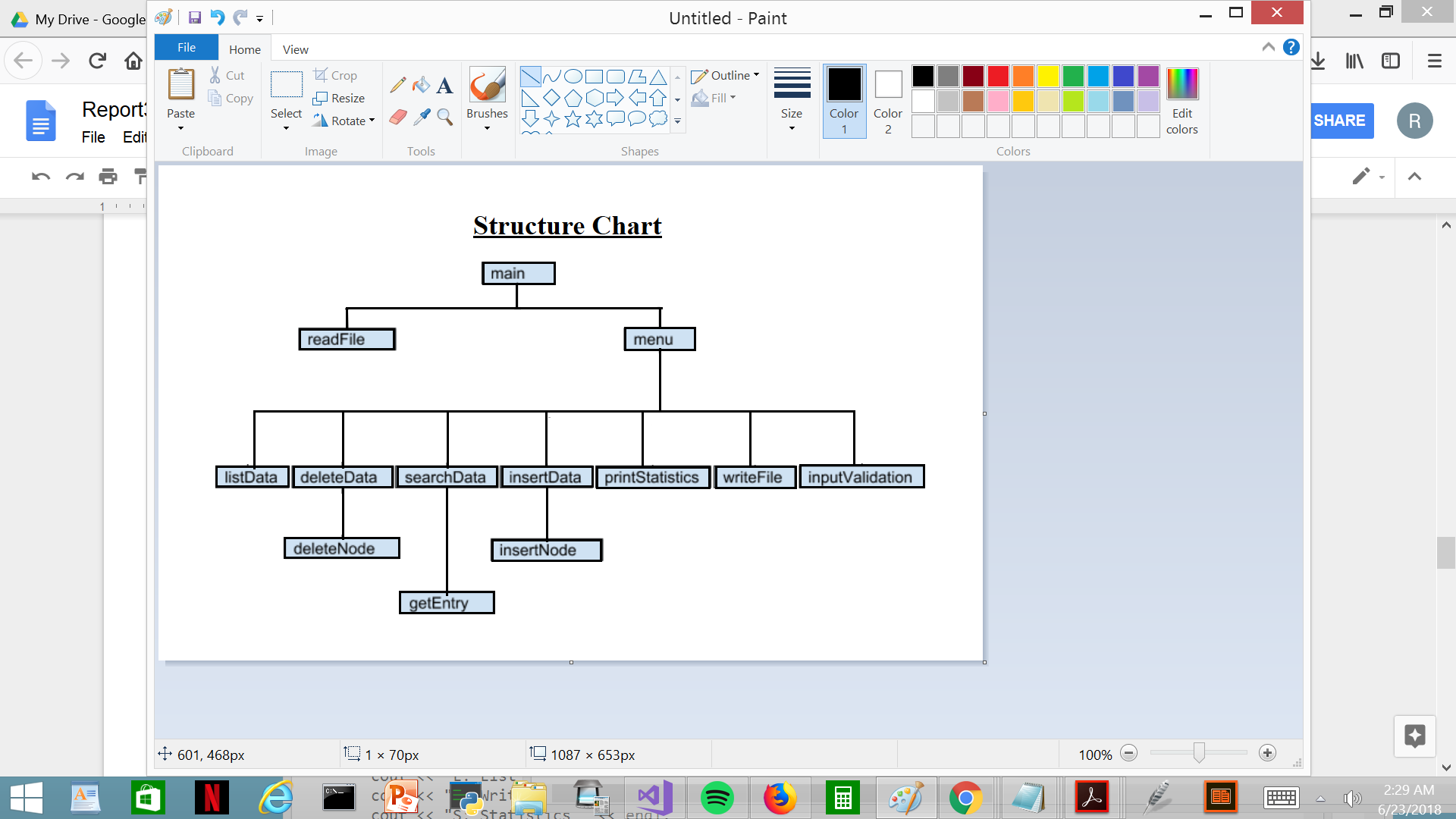
|  |
| --- |
| Binary Node |
| * item : ItemType * leftPtr : BinaryNode<ItemType>\* * rightPtr : BinaryNode<ItemType>\* |
| * BinaryNode( anItem : ItemType) : * BinaryNode( anItem : ItemType, left : BinaryNode<ItemType>\* , right : BinaryNode<ItemType>\*) : * setItem( anItem : const ItemType) : void * setLeftPtr( left : BinaryNode<ItemType>\*) : void * setRightPtr( right : BinaryNode<ItemType>\*) : void * getItem() const : ItemType * getLeftPtr() const : BinaryNode<ItemType>\* * getRightPtr() const : BinaryNode<ItemType>\* * isLeaf() const : bool |

|  |
| --- |
| Binary Tree |
| * rootPtr : BinaryNode<ItemType>\* * count : int |
| * BinaryTree() : * BinaryTree( tree : const BinaryTree<ItemType>) : * ~BinaryTree() : virtual * operator = ( sourceTree : const BinaryTree ) : BinaryTree * isEmpty() const : bool * size() const : int * clear() : void * preOrder( visit(obj : ItemType) : void ) const : void * inOrder( visit(obj :ItemType ) : void ) const : void * postOrder( visit(obj : ItemType ) : void ) const : void * levelOrder( visit(obj : ItemType ) : void ) const : void * indentPrint( visit(obj : ItemType) : void, level : int ) : void * writeFile( write( obj : ItemType, obj : ofstream) : void ) const : bool * insert( newData : const ItemType, compare( one : const ItemType, two : const ItemType) : int ) = 0 : virtual bool * remove( data : const ItemType, compare( one : const ItemType, two : const ItemType) : int ) = 0 : virtual bool * getEntry( anEntry : const ItemType, returnedItem : ItemType,   compare( one : const ItemType, two : const ItemType) : int ) const = 0 : virtual bool * theSmallest( item : ItemType) = 0 : virtual void * theLargest( item : ItemType) = 0 : virtual void * destroyTree( nodePtr : BinaryNode<ItemType>\*) : void * copyTree( nodePtr : const BinaryNode<ItemType>\*) : BinaryNode<ItemType>\* * \_preorder( visit( obj : ItemType) : void, nodePtr : BinaryNode<ItemType>\*) const : void * \_inorder( visit( obj : ItemType) : void , nodePtr : BinaryNode<ItemType>\*) const : void * \_postorder( visit( obj : ItemType) : void, nodePtr : BinaryNode<ItemType>\*) const : void * \_indentprint(visit(obj : ItemType) : void, nodePtr : BinaryNode<ItemType>\*, level : int) : void |

|  |
| --- |
| Binary Search Tree |
| * bn : BinaryNode<ItemType>\* |
| * \_insert( nodePtr : BinaryNode<ItemType>\*, newNode : BinaryNode<ItemType>\*,   compare( one : const ItemType, two : const ItemType) : int ) : BinaryNode<ItemType>\* * \_remove( nodePtr : BinaryNode<ItemType>\*, target : const ItemType, success : bool,   compare( one : const ItemType, two : const ItemType ) : int ) : BinaryNode<ItemType>\* * deleteNode( targetNodePtr : BinaryNode<ItemType>\*) : BinaryNode<ItemType>\* * removeLeftmostNode( nodePtr : BinaryNode<ItemType>\*, successor : ItemType) : BinaryNode<ItemType>\* * findNode( nodePtr : BinaryNode<ItemType>\*, target : const ItemType,   compare( one : const ItemType , two : const ItemType ) : int ) const : BinaryNode<ItemType>\* * findSmallest( nodePtr : BinaryNode<ItemType>\*, item : const ItemType ) const : BinaryNode<ItemType>\* * findLargest( nodePtr : BinaryNode<ItemType>\*, item : const ItemType) const : BinaryNode<ItemType>\* * insert( newEntry : const ItemType, compare( one : const ItemType, two : const ItemType) : int ) : bool * remove( anEntry : const ItemType, compare( one : const ItemType, two : const ItemType) : int ) : bool * getEntry( target : const ItemType, returnedItem : ItemType,   compare( one : const ItemType, two : const ItemType) : int ) const : bool * theSmallest( item : ItemType ) : void * theLargest( item : ItemType ) : void |

|  |
| --- |
| HashTable |
| * key : string * value : Incidents * next : HashTableNode\* * head : HashTableNode\* * collisions : int\* * loadFactor : int * totalNodes : int * totalCollisions : int |
| * HashTable() : * ~HashTable() : * insertNode ( incident : Incidents, key : string ) : void * deleteNode ( key : string ) : bool * searchNode ( key : string, nodeValue : Incidents) : bool * printHashTable () : void * printStatistics() : void * computeHash( ID : string ) : int |

**Structure Chart**

****

**Data Sample**

3121,TX,11/5/2017,26,White,M

1923,CO,11/1/2017,3,White,M

1467,MD,10/18/2017,3,Black,M

1230,NV,10/1/2017,59,White,M

1111,CA,6/14/2017,3,Asian,M

1392,PA,6/7/2017,3,White,M

1650,FL,6/5/2017,5,Unknown,M

1542,OH,5/12/2017,3,White,M

1802,CA,4/18/2017,3,Black,M

2591,FL,1/6/2017,5,Latino,M

2349,WA,9/23/2016,5,Unknown,M

2231,LA,7/17/2016,3,Black,M

2498,TX,7/7/2016,5,Black,M

**Incidents.h**

//Implementation file for Incidents class

//IDE: Visual Studio 2017

#ifndef \_INCIDENTS

#define \_INCIDENTS

#include <string>

#include <iomanip>

#include <iostream>

using std::string;

using std::ostream;

class Incidents

{

private:

string ID; // unique key

string state; // location of incident

string date; // date of incident

int fatalities; // number of fataliities in incident

string race; // race of the shooter

string gender; // gender of the shooter

public:

// constructors

Incidents();

Incidents(string sID, string sState, string sDate, int sFatalities, string sRace, string sGender);

// setters

void setID(string sID);

void setState(string sState);

void setDate(string sDate);

void setFatalities(int sFatalities);

void setRace(string sRace);

void setGender(string sGender);

// getters

string getID() const;

string getState() const;

string getDate() const;

int getFatalities() const;

string getRace() const;

string getGender() const;

};

#endif

**Project Status**

|  |  |  |
| --- | --- | --- |
| Area | Programmer | Status/Comments |
| Unit 1: Team Coordination | Amrita | In action |
| Unit 2: BST Algorithms | Rasna | In progress -30% |
| Unit 3: Hash List Algorithms | Charu | In progress -10% |
| Unit 4: Screen output | Amrita | Not started yet |
| Unit 5: File I/O | Charu | In progress -10% |
| Test Plan Presentation Outline | Amrita | In progress-30% |
| Project Documentation | Rasna | In progress- 40% |