Neil Rolf

CSC300

Program 6 – Design

**Class Declarations**

typedef std::string SElement

Class BST

*private variables*

* struct TreeNode
  + SElement title; //holds title string
  + int likes; //holds likes value
  + int views; //holds views value
  + TreeNode \*left //holds pointer to left child node
  + TreeNode \*right //holds pointer to right child node
* TreeNode \*root; //pointer to root node
* int likeTotal; //holds sum of likes
* int viewTotal; //holds sum of views
* void destroySubTree(TreeNode \*); //destructor
* void insert(TreeNode \*&, TreeNode \*&); //searches branches, inserts new node alphabetically
* void displayRoot(TreeNode \*); //prints title value in root node
* void displayInOrder(TreeNode \*) const //traverses tree to print node data in alphabetical order
* void displayTotal(TreeNode \*); //calculates total likes and views

*public variables*

* BST() //constructor, sets root to null
* ~BST() //destructor, wrapper for destroySubTree(root)
* void insertNode(SElement, int, int) //inserts new node into tree (alphabetically), calls insert() to search branches
* void displayRoot() //wrapper for displayRoot(root)
* void displayInOrder() const; //wrapper for displayInOrder(root)
* void displayTotal(int \*, int \*) //passes in catch variables for totals, calls displayTotals(root);

**Driver**

enum Menu {PRINT\_ROOT\_TITLE=1, PRINT\_ALL, PRINT\_TOTAL, EXIT};

struct Node{ //temp structure for driver functions

string title;

int likes;

int views;}

void ingestData(); //reads data from file

BST tree; //class initialization

int(main){

int \*catchLikes;

int \*catchViews

//USER MENU

int selection = 0;

do{

cout << “user menu and selection prompt”;

cin.clear();

cin >> selection;

switch(selection){

case PRINT\_ROOT\_TITLE:{

tree.displayRoot();

break;}

case PRINT\_ALL:{

cout << “table headers”;

tree.displayInOrder();

break;}

case PRINT\_TOTAL:{

tree.displayTotal(catchLikes, catchViews);

cout << “Total Likes: “ << catchLikes << endl;

cout << “Total Views: “ << catchViews << endl;

break;}

case EXIT:{

break;}

}while(selection !=4);

return 0;

}

void ingestData(){

//check for file

file.open(videos.txt)

if(!infile)

exit code

//read from file to temp structure

Node tempNode;

while(infile >> tempNode.likes)

{

infile >> tempNode.views;

infile.ignore();

getline(infile, tempNode.title);

tree.insertNode(tempNode.title, tempNode.likes, tempNode.views);

}

infile.close

return;

}