```
#include <iostream>
#include <string>
#include <vector>
#include <tuple>
#include <map>
#include "Bridges.h"
#include "DataSource.h"
#include "BSTElement.h"
#include <queue>
using namespace std;
using namespace bridges;
#include <stdlib.h>
#include "data_src/ActorMovieIMDB.h"


/*
  Step 1: Get the Bridges USER ID and API KEY for your program
        by creating an account and signing up (You can use any
        email): https://bridges-cs.herokuapp.com/signup

  Step 2: User ID (Username) and Key (API Sha1 Key) can be
        found under Profile on the top left

*/

BSTElement<string, string>* insertIntoBST(string actor, string movie, BSTElement<string,
string> *root)
{
   if (root == nullptr)
   {
      root = new BSTElement<string, string>(actor);
      root->setLabel(movie);
      return root;
   }

   if (actor > root->getKey())
   {
      root->setRight(insertIntoBST(actor, movie, root->getRight()));
   }
   else
   {
      root->setLeft(insertIntoBST(actor, movie, root->getLeft()));
   }
```

```
    return root;
}

int main(int argc, char **argv)
{

    // Step 3: Test if the following print statement is being run
    cout << "Bridges: IMDB Data in a BST\n";

    // Step 4: Add your User ID and API Key as secrets on Replit

    char* mySecret1 = getenv("SECRET_HOLDING_USER_ID");
    char* mySecret2 = getenv("SECRET_HOLDING_API_KEY");

    /* Step 5: Print your User ID and API Key from secrets to the console
       to test if secrets were properly loaded
    */
    cout<<"User ID: " << mySecret1<<endl;
    cout<<"API Key: " << mySecret2<<endl;
    /*
     Step 6: Create a Bridges object by uncommenting the next line
     and inserting the values from steps 1 and 2
    */
    Bridges bridges(1, mySecret1, mySecret2);

    /*
       Step 7: Import IMDB data into a vector<ActorMovieIMDB> by
            referring to the Bridges IMDB documentation:
            https://bridgesuncc.github.io/tutorials/Data_IMDB.html
    */
    DataSource ds (&bridges);
    vector<ActorMovieIMDB> actor_list = ds.getActorMovieIMDBData(1814);

    /*
       Step 8: Open the file "insertIntoBST.txt" and copy the provided
            function for inserting an actor/movie pair into a BST.
            Paste the function into this file above your "main" function.
    */

    /*
       Step 9: Use the insertion function from the previous step to insert
            any 100 actor/movie pairs into a Bridges BST. Refer to the
            Bridges IMDB documentation:
            https://bridgesuncc.github.io/tutorials/Data_IMDB.html
```

```
*/
BSTElement<string, string> *root = nullptr;
for(int i=0;i<100;i++){
    root = insertIntoBST(actor_list[i].getActor(),actor_list[i].getMovie(),root);
}
/*
   Step 10: Visualize this tree by referring to the Bridges BST documentation:
         https://bridgesuncc.github.io/tutorials/BinarySearchTree.html
*/
// bridges.setDataStructure(root);
// bridges.visualize();
/*
   Step 11: Color each level of the tree using a level-order traversal.
         Every node in a given level should have the same color.
         Do not use the same color in two consecutive levels. A starter
         queue has been provided in case you wish to use an iterative
         implementation of BFS.

         Refer to the Bridges BST Styling documentation:
         https://bridgesuncc.github.io/tutorials/BinarySearchTree.html

*/


map<BSTElement<string, string>*, int> q;
pair<BSTElement<string, string>*, int> newPair(root, 0);
q.insert(q.begin(), newPair);

root->setColor("pink");

while(!q.empty()){
    BSTElement<string, string> *currVal = q.begin()->first;
    int currLevel = q.begin()->second;
    if(currVal->getLeft() != nullptr){
        newPair.first = currVal->getLeft();
        newPair.second = currLevel + 1;
        q.insert(q.end(), newPair);
    }
    if(currVal->getRight() != nullptr){
        newPair.first = currVal->getRight();
        newPair.second = currLevel + 1;
        q.insert(q.end(), newPair);
    }
    if(currLevel%2==0){
```

```
            currVal->setColor("pink");
        }
        else{
            currVal->setColor("blue");
        }
        q.erase(q.begin());
    }



    /*
      Step 12: Visualize the updated tree. Comment out or remove the old
               visualization code from Step 10
    */

    bridges.setDataStructure(root);
    bridges.visualize();
    return 0;
}
```