

Programming Assignment #3

CS146-08

Professor: Mike Wu

By: Eric Wu

Table of Contents

- I. Explanation of my Google Search Engine Simulator design and implementation details. (Page 3)
- II. A list of classes/subroutines/function calls and explanations of each purpose. (Pages 4 – 14)
- III. Screen shots of each simulation process/procedure (Pages 15 – 31)
- IV. How to get the program running (Pages 32 – 37)
 - a. How to run the program (Page 37)
- V. Problems encountered (Page 38 - 39)
- VI. Lessons learned (Page 40)

1) Explain/Illustrate your Google Search Engine Simulator design and implementation details.

I built upon the program that I have written for programming assignment 2. I have deleted the quicksort and bucket but kept the pair class which keeps track of the url and score and the bst class. I just added new things to the node class which keeps track of the color of the node and I also adjusted the code of the bst class accordingly so that it not implements a red black tree. I used the same webcrawler to get the URLs and I assigned a random score value to each of them. For the first part of the program, I test out the bst on the URLs. For this, the updated bst class was added 30 nodes as an argument. This new BST unlike the previous one I wrote for programming assignment 2, keeps track of the color so that the tree properties keeps the tree balanced. The updated BST also has modified delete, insertion methods. I added deletetionfixup and instertionfixup so that the color also adjusts after you add or delete a node form the red black tree. I also added the rotate left/right methods because that is one of the way that we fix the structure of the tree so that it becomes balanced. For the actual program, I ask user to input a keyword. The webcrawler gets 30 urls and gives them random score. Then I inserted the URLs and corresponding scores into the red black tree structure. I then print out the websites and scores in sorted order using the inordertreewalk method to show the sorted order and preordertreewalk so that I can draw out the tree to see if it is indeed in the proper red black tree order. I then test the red black tree by inserting websites with scores to the URL that the user inputs until they input N to stop. I then print the websites and scores in sorted order so that the user can see the new websites in the Red black tree and also in preordertreewalk so that I can again draw the tree out to see if it stills follow the redblack tree properties. I then ask user to input a page rank. And I would return the color, url, and score corresponding to that pagerank. I test the delete method on one of the website that the user wants to delete. They input the score of the rank they want to delete and then it deletes it. And then the program prints out tree again to show user that the website indeed has been deleted and again it also prints out the tree in preordertreewalk order so that I can draw the tree to see if it still follows the red black tree properties. I then ask user if they want to search for another word. If they don't the program prints the top 10 keywords.

2)A list of classes/subroutines/function calls and explanations of each purpose.

Pair Class

```
public class Pair {
    private String url;
    private int value;
    /**
     * This creates a Pair object which has a string and a value
     * @param url The URL of the pair
     * @param value The value associated with the URL
     */
    public Pair(String url, int value) {
        this.url = url; //store user inputted url into the object's url
        this.value = value; //store user inputted value into the object's value
    }
    /**
     * Gets the URL
     * @return the URL
     */
    public String getUrl() {
        return url; //return the object's url
    }
    /**
     * Gets the value
     * @return the value
     */
    public int getValue() {
        return value; //return the object's value
    }
    /**
     * Combine the URL and Value into a string
     * @return the combined URL and value string
     */
    public String toString() {
        return "Score: " + value + " URL: " + url; //return a string contraining the url and value
    }
}
```

The purpose of this pair class is so that we can store each URL with a corresponding score. I could have used a map to store the data but by making a new object to store the URL and score, I know for certain that the data would be stable.

The constructor takes in a String (which is the URL) and a integer (which is the score of the URL). I have 3 methods in this class.

One of them is **getUrl()** which returns the URL that is associated with the object.

Another one is the **getValue()** which returns the score that is associated with the object.

And finally there is the **toString()** that just returns the score and URL in one line, which just makes it easier to print out the results.

Node Class

```
/**
 * gets the parent of a node
 * @return the parent of a node
 */
public Node getParent(){
    return parent;
}
/**
 * gets the left child of a node
 * @return the left child
 */
public Node getLeft(){
    return left;
}
/**
 * gets the right child of a node
 * @return the right child
 */
public Node getRight(){
    return right;
}
/**
 * gets the key or the url and corresponding score of a node
 * @return the key of a node
 */
public Pair getKey(){
    return key;
}
public String getColor() {
    return color;
}
```

```

/**
 * gets the parent of a node
 * @return the parent of a node
 */
public Node getParent() {
    return parent;
}

/**
 * gets the left child of a node
 * @return the left child
 */
public Node getLeft() {
    return left;
}

/**
 * gets the right child of a node
 * @return the right child
 */
public Node getRight() {
    return right;
}

/**
 * gets the key or the url and corresponding score of a node
 * @return the key of a node
 */
public Pair getKey() {
    return key;
}

public String getColor() {
    return color;
}
}

```

The point of this class is because we need this in order to implement a red black tree data structure. Each node keeps track of its color, parents, left child, right child, and the key or the score and URL.

The constructor sets key to the argument which is of type Pair. Then it sets all its other connections to null.

The methods are self explanatory and I wrote comments on it. Basically they are getter and - setter methods which we can use to modify the Node's connections and retrieve data.

Red Black Tree Class

```
import java.util.*;
public class RedBlackTree {
    Node root;
    Node nil;
    private int counter = -1;
    private int counter2 = 0;
    private int counter3 = 0;
    /**
     * Creates a bst object
     */
    public RedBlackTree() {
        Pair sentinel = new Pair("sentinel", 0);
        nil = new Node(sentinel);
        nil.setColor("BLACK");
        root = nil;
    }
    /**
     * print out the tree nodes in sorted order
     * @param x the node where we want to start at usually the root of the tree
     */
    public void inOrderTreeWalk(Node x) { //used to sort the nodes in ascending order
        if(x != nil) {
            inOrderTreeWalk(x.getLeft());
            counter++;
            System.out.println("Index [" + counter + "] " + "Rank: " + (counter2 - counter) + " Color: " + x.getColor() + " " + x.getKey().toString());
            inOrderTreeWalk(x.getRight());
        }
    }
    public void preOrderTreeWalk(Node x) { //used to check if structure of tree is correct
        if(x == nil) {
            return;
        }
        System.out.println(" Color: " + x.getColor() + " " + x.getKey().toString());
        preOrderTreeWalk(x.getLeft());
        preOrderTreeWalk(x.getRight());
    }
}
```

This is the red black tree class and it has a constructor that creates a sentinel node which is the nil node with the color black and then sets the root to equal nil.

The in order tree walk method prints out the node in ascending order so that we can get its sorted order.

The pre order tree walk methods prints out the node in pre order or root, left, right, and the purpose of this method is so that we can check whether the nodes in the tree is indeed following the red black tree properties.

```

/**
 * resets the counter for the inordertreewalk that keeps track of index and pagerank
 */
public void resetCounter() {
    counter = -1;
}
/**
 * gets the url and score of a node given a pagerank
 * @param rank the page rank of the url we want to search
 * @return the url and score
 */
public Node getUrl(int rank) {
    int rankCounter = 1;
    Node x = getMax(getRoot());
    if(rank == 1) { //rank 1 is max in tree
        return x;
    } else { //start from max and go to predecessor until we hit desired pagerank
        while(rankCounter != rank) {
            x = treePredecessor(x);
            rankCounter++;
        }
    }
    return x;
}
/**
 * gets the root of the tree
 * @return the root of the tree
 */
public Node getRoot() {
    return root;
}
/**
 * get the min value of a tree could be a subtree
 * @param min the node that we want to get the min of
 * @return the min node
 */
public Node getMin(Node min){
    Node x = min;
    while(x.getLeft() != nil) {
        x = x.getLeft();
    }
    return x;
}

```

The purpose of the getUrl method is so that it can return the corresponding node to the rank that the user inputs as the argument.

The getroot method gets the root of the tree

The getMin method gets the minimum score node in the tree


```

/**
 * get the max value of a tree could be a subtree
 * @param max the node that we want the max of
 * @return the max node
 */
public Node getMax(Node max) {
    Node x = max;
    while(x.getRight() != nil) {
        x = x.getRight();
    } return x;
}

/**
 * gets the predecessor of a node or the next smallest one
 * @param predecessor the node we want the predecessor of
 * @return the predecessor of the given node
 */
public Node treePredecessor(Node predecessor) {
    Node x = predecessor;
    if(x.getLeft() != nil) {
        return getMax(x.getLeft());
    }
    Node y = x.getParent();
    while(y != nil && x == y.getLeft()) {
        x = y;
        y = y.getParent();
    } return y;
}

/**
 * gets the successor of a node or the next biggest one
 * @param successor the node we want the successor of
 * @return the successor of the given node
 */
public Node treeSuccessor(Node successor) {
    Node x = successor;
    if(x.getRight() != nil) {
        return getMin(x.getRight());
    }
    Node y = x.getParent();
    while(y != nil && x == y.getRight()) {
        x = y;
        y = y.getParent();
    } return y;
}

```

The getMax method gets the node with the maximum score

The tree predecessor method gets the predecessor of the node that the user inputs as argument or the previous smaller node

The tree successor methods gets the successor of the node that the user inputs as argument or the next bigger node.

```

/**
 * inserting a node to the tree and keeping its redblack structure
 * @param z the node that we want to insert to the tree
 */
public void redBlackTreeInsert(Node z) {
    Node y = nil;
    Node x = root;
    while(x != nil) {
        y = x;
        if(z.getKey().getValue() < x.getKey().getValue()) {
            x = x.getLeft();
        } else {
            x = x.getRight();
        }
    }
    z.setParent(y);
    if(y == nil) { //case 1
        root = z;
    } else if(z.getKey().getValue() < y.getKey().getValue()) { //case 2
        y.setLeft(z);
    } else if(z.getKey().getValue() >= y.getKey().getValue()) { //case 3
        y.setRight(z);
    }
    counter2++; //keep track of the counter
    z.setLeft(nil);
    z.setRight(nil);
    z.setColor("RED");
    redBlackInsertFixUp(z);
}

```

The `redblacktreeinsert` methods takes in a node as the argument. It then adds it into the tree so that it keeps the red black tree properties. Then we will call the `insertfixup` method so that the tree will be in redblacktree properties if it was indeed changed out of place. Notice that I increase the counter so that when a new node is added to the tree, I still know how many nodes are currently in the tree.

```

/**
 * fixes up the redblacktree after an insertion takes place
 * @param z the node that we are fixing
 */
public void redBlackInsertFixUp(Node z) {
    while(z.getParent().getColor().equals("RED")) {
        if(z.getParent() == z.getParent().getParent().getLeft()) {
            boolean done = false;
            Node y = z.getParent().getParent().getRight();
            if(y.getColor().equals("RED")) {
                z.getParent().setColor("BLACK");
                y.setColor("BLACK");
                z.getParent().getParent().setColor("RED");
                z = z.getParent().getParent();
                done = true;
            } else if(z == z.getParent().getRight()) {
                z = z.getParent();
                leftRotate(z);
            }
            if(!done) {
                z.getParent().setColor("BLACK");
                z.getParent().getParent().setColor("RED");
                rightRotate(z.getParent().getParent());
            }
        } else if(z.getParent() == z.getParent().getParent().getRight()) {
            Node y = z.getParent().getParent().getLeft();
            boolean done = false;
            if(y.getColor().equals("RED")) {
                z.getParent().setColor("BLACK");
                y.setColor("BLACK");
                z.getParent().getParent().setColor("RED");
                z = z.getParent().getParent();
                done = true;
            } else if(z == z.getParent().getLeft()) {
                z = z.getParent();
                rightRotate(z);
            }
            if(!done) {
                z.getParent().setColor("BLACK");
                z.getParent().getParent().setColor("RED");
                leftRotate(z.getParent().getParent());
            }
        }
    }
    root.setColor("BLACK");
}

```

This redblacktree insert fixup method takes in the node where the position of the inserted was added. Then it goes through the cases to fix the order and structure of the tree so that it follows the red black tree format. When the parent is not red anymore, the while loops end and then we finally set the root's color to black so that the red black tree structure is preserved.

```

/**
 * the transplant part of the delete method
 * @param u the node we want to replace
 * @param v the node we want to transplant
 */
public void redBlackTransplant(Node u, Node v) {
    if (u.getParent() == nil) {
        root = v;
    } else if (u == u.getParent().getLeft()) {
        u.getParent().setLeft(v);
    } else {
        u.getParent().setRight(v);
    }
    v.setParent(u.getParent());
}

/**
 * deleting a node from the red black tree
 * @param z the node we want to delete
 */
public void redBlackTreeDelete(Node z) {
    Node y = z;
    Node x;
    String originalColor = y.getColor();
    if (z.getLeft() == nil) { //case 1
        x = z.getRight();
        redBlackTransplant(z, z.getRight());
    } else if (z.getRight() == nil) { //case 2
        x = z.getLeft();
        redBlackTransplant(z, z.getLeft());
    } else { //case 3
        y = getMin(z.getRight());
        originalColor = y.getColor();
        x = y.getRight();
        if (y.getParent() == z) {
            x.setParent(y);
        } else {
            redBlackTransplant(y, y.getRight());
            y.setRight(z.getRight());
            y.getRight().setParent(y);
        }
        redBlackTransplant(z, y);
        y.setLeft(z.getLeft());
        y.getLeft().setParent(y);
        y.setColor(z.getColor());
    }
    if (originalColor.equals("BLACK")) {
        redBlackDeleteFixUp(x);
    }
    counter2--; //keep track of the counter
}

```

The redblacktree transplant method is the modified version of the bst transplant which is used to delete a node from the tree.

The redblacktreedelete method takes in a node that the user wants to delete as the argument. It then goes through the cases to see what to do. Then after the method goes through the cases, it calls the redblacktreefixup so that the tree keeps its redblacktree properties even after a node has been deleted. Notice that I decrement the counter so that it keeps track of the amount of nodes in the tree even when I delete a node.

```

/**
 * fixes the red black tree after a node has been deleted from the tree
 * @param x the position fo the node that was deleted
 */
public void redBlackDeleteFixUp(Node x) {
    while(x != root && x.getColor().equals("BLACK")) {
        if(x == x.getParent().getLeft()) {
            boolean done = false;
            Node w = x.getParent().getRight();
            if(w.getColor().equals("RED")) {
                w.setColor("BLACK");
                x.getParent().setColor("RED");
                leftRotate(x.getParent());
                w = x.getParent().getRight();
            } if(w.getLeft().getColor().equals("BLACK") && w.getRight().getColor().equals("BLACK")) {
                w.setColor("RED");
                x = x.getParent();
                done = true;
            } else if(w.getRight().getColor().equals("BLACK")) {
                w.getLeft().setColor("BLACK");
                w.setColor("RED");
                rightRotate(w);
                w = x.getParent().getRight();
            }
            if(!done) {
                w.setColor(x.getParent().getColor());
                x.getParent().setColor("BLACK");
                w.getRight().setColor("BLACK");
                leftRotate(x.getParent());
                x = root;
            }
        } else {

```

```

            Node w = x.getParent().getLeft();
            boolean done = false;
            if(w.getColor().equals("RED")) {
                w.setColor("BLACK");
                x.getParent().setColor("RED");
                rightRotate(x.getParent());
                w = x.getParent().getLeft();
            } if(w.getRight().getColor().equals("BLACK") && w.getLeft().getColor().equals("BLACK")) {
                w.setColor("RED");
                x = x.getParent();
                done = true;
            } else if(w.getLeft().getColor().equals("BLACK")) {
                w.getRight().setColor("BLACK");
                w.setColor("RED");
                leftRotate(w);
                w = x.getParent().getLeft();
            }
            if(!done) {
                w.setColor(x.getParent().getColor());
                x.getParent().setColor("BLACK");
                w.getLeft().setColor("BLACK");
                rightRotate(x.getParent());
                x = root;
            }
        }
    }
    x.setColor("BLACK");
}

```

The `redblackdeletefixup` method fixes up the tree after a node has been deleted from the tree so that it keeps the `redblacktree` structure. It has 2 cases which it can be and then in each case, it has 4 cases that it goes through to fix the structure of the tree.

```

}
/**
 * rotates left where the node is
 * @param x the node that we want to rotate left
 */
public void leftRotate(Node x) {
    Node y = x.getRight();
    x.setRight(y.getLeft());
    if(y.getLeft() != nil) {
        y.getLeft().setParent(x);
    }
    y.setParent(x.getParent());
    if(x.getParent() == nil) {
        root = y;
    } else if(x == x.getParent().getLeft()) {
        x.getParent().setLeft(y);
    } else {
        x.getParent().setRight(y);
    }
    y.setLeft(x);
    x.setParent(y);
}
/**
 * rotates right where the node is
 * @param x the node that we want to rotate right
 */
public void rightRotate(Node x) {
    Node y = x.getLeft();
    x.setLeft(y.getRight());
    if(y.getRight() != nil) {
        y.getRight().setParent(x);
    }
    y.setParent(x.getParent());
    if(x.getParent() == nil) {
        root = y;
    } else if(x == x.getParent().getRight()) {
        x.getParent().setRight(y);
    } else {
        x.getParent().setLeft(y);
    }
    y.setRight(x);
    x.setParent(y);
}

```

The leftrotate method rotates the node left.

The rightrotate method rotates the node right.

3) Provide a lot of screen shots of each simulation process/procedure including inputs and outputs for each of function listed in item 6 associated with the two major features listed in the end of the Problem Statements. This is to verify your codes and to check to see if your codes match the functional requirements and run correctly by yourself.

PageRank Class

```
import java.util.*;

public class Pagerank {

    public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        boolean done = false;
        LinkedHashSet<String> list = new LinkedHashSet<>();
        Iterator it = list.iterator();
        System.out.println("Testing Red Black Tree:");
        done = false;
        while (!done) {
            System.out.print("What keyword do you want to search?: ");
            String keyWord = scan.next();
            list.add(keyWord);
            Spider crawler = new Spider(); //new spider object
            crawler.search("https://arstechnica.com/", keyWord); //website we are getting URLs from
            ArrayList<String> array = crawler.getPageList(); //set the string arraylist equal to the crawler's list of URLs
            RedBlackTree bst = new RedBlackTree(); //create a bst object

            System.out.println("Printing out the URLs in the order that is inserted into the Red Black Tree");
            for (int i = 0; i < array.size(); i++) {
                Pair pair = new Pair(array.get(i), makeRandomScore()); //assign each url a random score
                Node node = new Node(pair); //node has value of url and score
                bst.redBlackTreeInsert(node); //store each node into the tree
                System.out.println(node.getKey());
            }
            System.out.println();
            System.out.println("Red Black Tree in pre order tree walk **(Used to check the structure of the RB Tree)**");
            bst.preOrderTreeWalk(bst.getRoot());
            System.out.println();
            System.out.println("Red Black Tree in order tree walk **(Sorted List)**");
            bst.inOrderTreeWalk(bst.getRoot());
            bst.resetCounter();
            System.out.println();

            boolean loopEnd = false;
            while (loopEnd != true) {
                System.out.println("What is the URL you want to add? ");
                String url = scan.next();
                System.out.println("What is the score for the URL? ");
                int score = scan.nextInt();
                scan.nextLine();
                Pair website = new Pair(url, score);
                Node newWebsite = new Node(website);
                bst.redBlackTreeInsert(newWebsite);
                System.out.println("Do you want to add more websites? (Y/N)");
                String response = scan.next();
            }
        }
    }
}
```

```

        if(response.equalsIgnoreCase("n")) {
            loopEnd = true;
        }
    }

    System.out.println("Red Black Tree in preorder tree walk with the added URLs "(Used to check structure of the RB tree)");
    bst.preOrderTreeWalk(bst.getRoot());
    System.out.println();
    System.out.println("Red Black Tree in order tree walk with the added URLs "(Sorted List)");
    bst.inOrderTreeWalk(bst.getRoot());
    bst.resetCounter();

    System.out.println("Enter the Rank of the URL that you want to delete: ");
    int urlToDelete = scan.nextInt();
    scan.nextLine();
    bst.redBlackTreeDelete(bst.getUrl(urlToDelete));
    System.out.println();

    System.out.println("Binary Search Tree in preorder tree walk after deleted URL "(Used to check structure of the RB tree)");
    bst.preOrderTreeWalk(bst.getRoot());
    System.out.println();

    System.out.println("Binary Search Tree in order tree walk after deleted URL "(Sorted List)");
    bst.inOrderTreeWalk(bst.getRoot());
    bst.resetCounter();
    System.out.println();

    System.out.print("Enter a pagerank to search for the URL: ");
    int rankToCheck = scan.nextInt();
    scan.nextLine();
    System.out.println("The score and Uri with pagerank " + rankToCheck + " is " + bst.getUrl(rankToCheck).getKey().getUri() + " with a score of " + bst.getUrl(rankToCheck).getKey().getValue());
    System.out.println();

    System.out.print("Do you want to search for any other words? (Y/N): ");
    String response = scan.nextLine().toUpperCase();
    while (response.equalsIgnoreCase("Y")) {
        done = false;
    } else {
        done = true;
    }
}

System.out.println();
System.out.println("The Top 10 Unique Searches:");
int counter = 0;
it = list.iterator();
while (it.hasNext() && counter < 10) {
    System.out.println(it.next());
}
System.out.println();
}

public static int makeRandomScore() { //creates a random score based on the 4 criterias and return it
    Random random = new Random();
    int randomFrequency = random.nextInt(100) + 1; //random frequency score of the keyword
    int randomHistory = random.nextInt(100) + 1; //random time score of how long the website has been created
    int randomRelevancy = random.nextInt(100) + 1; //random relevancy score of the website
    int randomAdvertisement = random.nextInt(100) + 1; //random score on how much website paid google
    int totalScore = randomFrequency + randomHistory + randomRelevancy + randomAdvertisement; //add up the 4 scores
    return totalScore; //return the total score
}
}

```

First I will explain how the WebCrawler works. The WebCrawler application I am using also uses jsoup and it has two classes, a Spider class and a SpiderLeg class. The spider class basically uses the spiderleg class to “crawl” on a website, trying to find if it matches with the keyword we provide. The spider class takes two arguments to begin, a website url and a keyword string from which it is going to search the website for. The spider class has three instance variables, one called MAX_PAGES_TO_SEARCH which I set to 30 because we are looking for 30 websites. The next variable is pagesVisted which is a stored in a set. And finally the last variable is called pagesToVisit which keeps track of the remaining pages needed to check. It is stored with a list. This program gets 30 URLs by navigating from the main page. It then looks around for any links on the website, and it adds them to pagesToVisit. Then it traverse down the list of pagesToVisit and if check if the URL page contains the keyword and if it does, it tells us by printing out it found it. Then it adds the URL to the pagesVisited set. In this case, I used the made a spider object with the URL as <https://arstechnica.com/> and user input as the keyword. I made this program repeat asking user for keywords until they type N and they will exit out of the program. After that, the program will print out the top 10 unique keywords that the user have

inputted. The program starts by testing the redblacktreeinsertion. After that it will go test the delete, search, and sorting. To make the program seem less clogged up by text, I have commented out the print statements from the webcrawler.

The following is the result from running the program with further explanation.

```
run:
Testing Red Black Tree:
What keyword do you want to search?: apple
Printing out the URLs in the order that is inserted into the Red BLaCK Tree
Score: 231 URL: http://arstechnica.com/?theme=dark
Score: 176 URL: https://arstechnica.com/store/
Score: 240 URL: https://arstechnica.com/search/
Score: 213 URL: http://arstechnica.com/?view=archive
Score: 195 URL: https://arstechnica.com/reviews/
Score: 213 URL: http://arstechnica.com/?theme=light
Score: 373 URL: https://arstechnica.com/about-us/
Score: 143 URL: https://arstechnica.com/staff-directory/
Score: 201 URL: https://arstechnica.com/civis/ucp.php?mode=sendpassword
Score: 190 URL: https://arstechnica.com/civis/ucp.php?mode=login&return_to=%2F
Score: 154 URL: https://arstechnica.com
Score: 192 URL: https://arstechnica.com/rss-feeds/
Score: 172 URL: https://arstechnica.com/science/
Score: 185 URL: https://arstechnica.com/advertise-with-us/
Score: 122 URL: https://arstechnica.com/?view=mobile
Score: 236 URL: https://arstechnica.com/contact-us/
Score: 246 URL: https://arstechnica.com/
Score: 152 URL: http://arstechnica.com/?view=grid
Score: 216 URL: https://arstechnica.com/tech-policy/
Score: 228 URL: https://arstechnica.com/gadgets/
Score: 61 URL: https://arstechnica.com/civis/
Score: 273 URL: https://arstechnica.com/#site-menu
Score: 216 URL: https://arstechnica.com/gaming/
Score: 181 URL: https://arstechnica.com/store/product/subscriptions/
Score: 262 URL: https://arstechnica.com/features/
Score: 104 URL: https://arstechnica.com/information-technology/
Score: 222 URL: https://arstechnica.com/reprints/
Score: 199 URL: https://arstechnica.com/cars/
Score: 227 URL: https://arstechnica.com/#main
Score: 96 URL: http://video.arstechnica.com/
```

The program begins by asking user to input a keyword to search and in this case I have inputted the word apple.

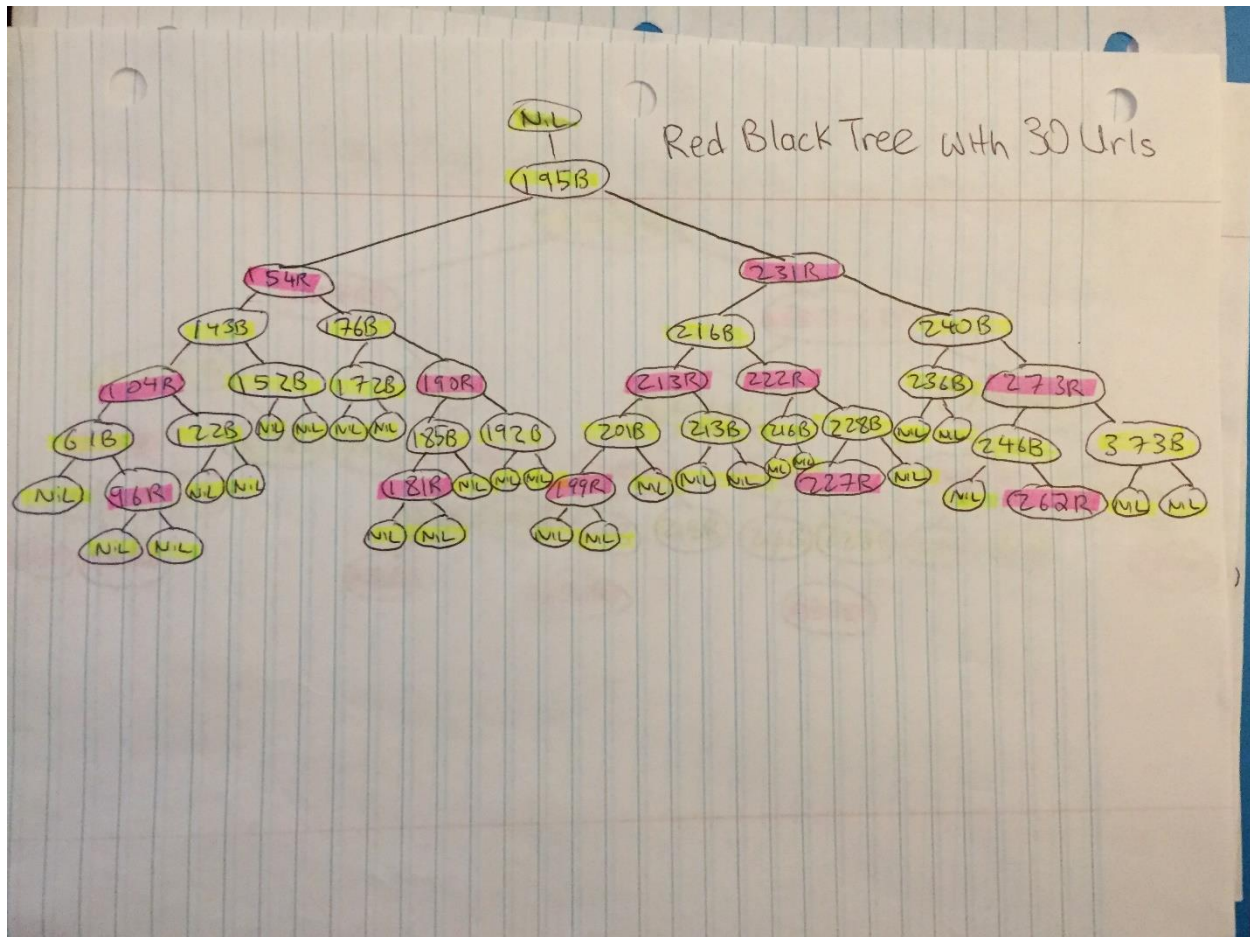
The program then prints out the nodes that are added into the red black tree.

```

Red Black Tree in pre order tree walk **(Used to check the structure of the RB Tree)**
Color: BLACK Score: 195 URL: https://arstechnica.com/reviews/
Color: RED Score: 154 URL: https://arstechnica.com
Color: BLACK Score: 143 URL: https://arstechnica.com/staff-directory/
Color: RED Score: 104 URL: https://arstechnica.com/information-technology/
Color: BLACK Score: 61 URL: https://arstechnica.com/civis/
Color: RED Score: 96 URL: http://video.arstechnica.com/
Color: BLACK Score: 122 URL: https://arstechnica.com/?view=mobile
Color: BLACK Score: 152 URL: http://arstechnica.com/?view=grid
Color: BLACK Score: 176 URL: https://arstechnica.com/store/
Color: BLACK Score: 172 URL: https://arstechnica.com/science/
Color: RED Score: 190 URL: https://arstechnica.com/civis/ucp.php?mode=login&return_to=%2F
Color: BLACK Score: 185 URL: https://arstechnica.com/advertise-with-us/
Color: RED Score: 181 URL: https://arstechnica.com/store/product/subscriptions/
Color: BLACK Score: 192 URL: https://arstechnica.com/rss-feeds/
Color: RED Score: 231 URL: http://arstechnica.com/?theme=dark
Color: BLACK Score: 216 URL: https://arstechnica.com/tech-policy/
Color: RED Score: 213 URL: http://arstechnica.com/?view=archive
Color: BLACK Score: 201 URL: https://arstechnica.com/civis/ucp.php?mode=sendpassword
Color: RED Score: 199 URL: https://arstechnica.com/cars/
Color: BLACK Score: 213 URL: http://arstechnica.com/?theme=light
Color: RED Score: 222 URL: https://arstechnica.com/reprints/
Color: BLACK Score: 216 URL: https://arstechnica.com/gaming/
Color: BLACK Score: 228 URL: https://arstechnica.com/gadgets/
Color: RED Score: 227 URL: https://arstechnica.com/#main
Color: BLACK Score: 240 URL: https://arstechnica.com/search/
Color: BLACK Score: 236 URL: https://arstechnica.com/contact-us/
Color: RED Score: 273 URL: https://arstechnica.com/#site-menu
Color: BLACK Score: 246 URL: https://arstechnica.com/
Color: RED Score: 262 URL: https://arstechnica.com/features/
Color: BLACK Score: 373 URL: https://arstechnica.com/about-us/

```

The red black tree then is printed in `preordertreewalk` so that I can draw the tree to see if it follows the red black tree structure.



We can indeed see that it follows the red black tree format

The yellow highlight represents the black colored nodes and the pink highlight represents the red colored nodes.

```

Red Black Tree in order tree walk **(Sorted List)**
Index [0] Rank: 30 Color: BLACK Score: 61 URL: https://arstechnica.com/civis/
Index [1] Rank: 29 Color: RED Score: 96 URL: http://video.arstechnica.com/
Index [2] Rank: 28 Color: RED Score: 104 URL: https://arstechnica.com/information-technology/
Index [3] Rank: 27 Color: BLACK Score: 122 URL: https://arstechnica.com/?view=mobile
Index [4] Rank: 26 Color: BLACK Score: 143 URL: https://arstechnica.com/staff-directory/
Index [5] Rank: 25 Color: BLACK Score: 152 URL: http://arstechnica.com/?view=grid
Index [6] Rank: 24 Color: RED Score: 154 URL: https://arstechnica.com
Index [7] Rank: 23 Color: BLACK Score: 172 URL: https://arstechnica.com/science/
Index [8] Rank: 22 Color: BLACK Score: 176 URL: https://arstechnica.com/store/
Index [9] Rank: 21 Color: RED Score: 181 URL: https://arstechnica.com/store/product/subscriptions/
Index [10] Rank: 20 Color: BLACK Score: 185 URL: https://arstechnica.com/advertise-with-us/
Index [11] Rank: 19 Color: RED Score: 190 URL: https://arstechnica.com/civis/ucp.php?mode=login&return_to=%2F
Index [12] Rank: 18 Color: BLACK Score: 192 URL: https://arstechnica.com/rss-feeds/
Index [13] Rank: 17 Color: BLACK Score: 195 URL: https://arstechnica.com/reviews/
Index [14] Rank: 16 Color: RED Score: 199 URL: https://arstechnica.com/cars/
Index [15] Rank: 15 Color: BLACK Score: 201 URL: https://arstechnica.com/civis/ucp.php?mode=sendpassword
Index [16] Rank: 14 Color: RED Score: 213 URL: http://arstechnica.com/?view=archive
Index [17] Rank: 13 Color: BLACK Score: 213 URL: http://arstechnica.com/?theme=light
Index [18] Rank: 12 Color: BLACK Score: 216 URL: https://arstechnica.com/tech-policy/
Index [19] Rank: 11 Color: BLACK Score: 216 URL: https://arstechnica.com/gaming/
Index [20] Rank: 10 Color: RED Score: 222 URL: https://arstechnica.com/reprints/
Index [21] Rank: 9 Color: RED Score: 227 URL: https://arstechnica.com/#main
Index [22] Rank: 8 Color: BLACK Score: 228 URL: https://arstechnica.com/gadgets/
Index [23] Rank: 7 Color: RED Score: 231 URL: http://arstechnica.com/?theme=dark
Index [24] Rank: 6 Color: BLACK Score: 236 URL: https://arstechnica.com/contact-us/
Index [25] Rank: 5 Color: BLACK Score: 240 URL: https://arstechnica.com/search/
Index [26] Rank: 4 Color: BLACK Score: 246 URL: https://arstechnica.com/
Index [27] Rank: 3 Color: RED Score: 262 URL: https://arstechnica.com/features/
Index [28] Rank: 2 Color: RED Score: 273 URL: https://arstechnica.com/#site-menu
Index [29] Rank: 1 Color: BLACK Score: 373 URL: https://arstechnica.com/about-us/

```

This prints out the list in ascending order or sorted order so that we can see the ranks of the URLs along with the index, color, score and URL

```

What is the URL you want to add?
mikewu.com
What is the score for the URL?
999
Do you want to add more websites? (Y/N)
y
What is the URL you want to add?
cs146.com
What is the score for the URL?
100
Do you want to add more websites? (Y/N)
n

```

We ask the user if they would like to input an URLS?

The user input mikewu with a score of 999

Then the program asks user if they want to add another website

The user types y for yes and then adds the website cs146.com with the score of 100

The program asks user if they want to add another website

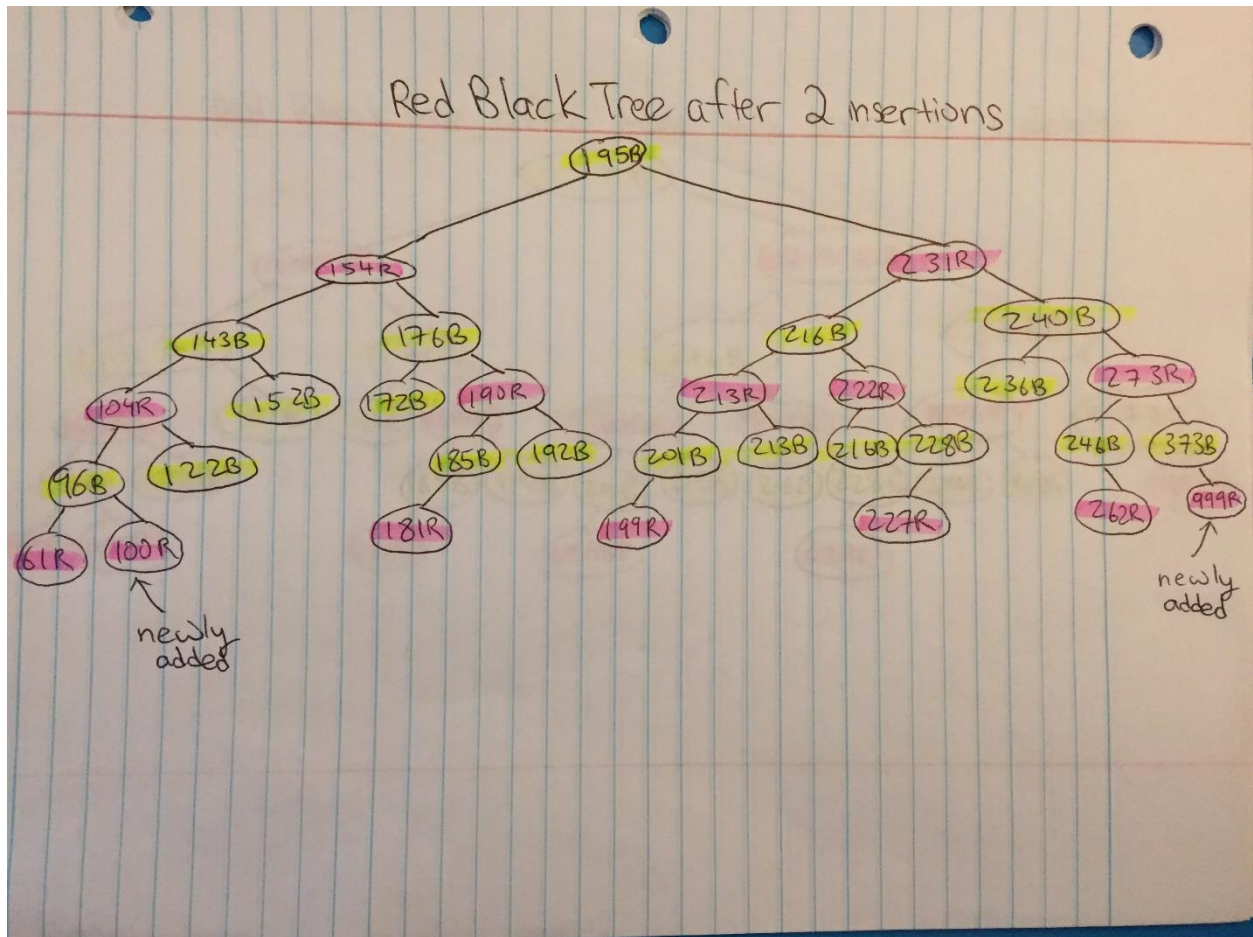
This time the user types N for no.

```

Red Black Tree in preorder tree walk with the added URLs **(Used to check structure of the RB tree)**
Color: BLACK Score: 195 URL: https://arstechnica.com/reviews/
Color: RED Score: 154 URL: https://arstechnica.com
Color: BLACK Score: 143 URL: https://arstechnica.com/staff-directory/
Color: RED Score: 104 URL: https://arstechnica.com/information-technology/
Color: BLACK Score: 96 URL: http://video.arstechnica.com/
Color: RED Score: 61 URL: https://arstechnica.com/civis/
Color: RED Score: 100 URL: cs146.com
Color: BLACK Score: 122 URL: https://arstechnica.com/?view=mobile
Color: BLACK Score: 152 URL: http://arstechnica.com/?view=grid
Color: BLACK Score: 176 URL: https://arstechnica.com/store/
Color: BLACK Score: 172 URL: https://arstechnica.com/science/
Color: RED Score: 190 URL: https://arstechnica.com/civis/ucp.php?mode=login&return_to=%2F
Color: BLACK Score: 185 URL: https://arstechnica.com/advertise-with-us/
Color: RED Score: 181 URL: https://arstechnica.com/store/product/subscriptions/
Color: BLACK Score: 192 URL: https://arstechnica.com/rss-feeds/
Color: RED Score: 231 URL: http://arstechnica.com/?theme=dark
Color: BLACK Score: 216 URL: https://arstechnica.com/tech-policy/
Color: RED Score: 213 URL: http://arstechnica.com/?view=archive
Color: BLACK Score: 201 URL: https://arstechnica.com/civis/ucp.php?mode=sendpassword
Color: RED Score: 199 URL: https://arstechnica.com/cars/
Color: BLACK Score: 213 URL: http://arstechnica.com/?theme=light
Color: RED Score: 222 URL: https://arstechnica.com/reprints/
Color: BLACK Score: 216 URL: https://arstechnica.com/gaming/
Color: BLACK Score: 228 URL: https://arstechnica.com/gadgets/
Color: RED Score: 227 URL: https://arstechnica.com/#main
Color: BLACK Score: 240 URL: https://arstechnica.com/search/
Color: BLACK Score: 236 URL: https://arstechnica.com/contact-us/
Color: RED Score: 273 URL: https://arstechnica.com/#site-menu
Color: BLACK Score: 246 URL: https://arstechnica.com/
Color: RED Score: 262 URL: https://arstechnica.com/features/
Color: BLACK Score: 373 URL: https://arstechnica.com/about-us/
Color: RED Score: 999 URL: mikewu.com

```

The red black tree then is printed in preorder treewalk so that I can draw the tree to see if it follows the red black tree structure.



We can indeed see that it follows the red black tree format even after the two URLs are added into the redblacktree.

The yellow highlight represents the black colored nodes and the pink highlight represents the red colored nodes.

```

Red Black Tree in order tree walk with the added URLs **(Sorted List)**
Index [0] Rank: 32 Color: RED Score: 61 URL: https://arstechnica.com/civis/
Index [1] Rank: 31 Color: BLACK Score: 96 URL: http://video.arstechnica.com/
Index [2] Rank: 30 Color: RED Score: 100 URL: cs146.com
Index [3] Rank: 29 Color: RED Score: 104 URL: https://arstechnica.com/information-technology/
Index [4] Rank: 28 Color: BLACK Score: 122 URL: https://arstechnica.com/?view=mobile
Index [5] Rank: 27 Color: BLACK Score: 143 URL: https://arstechnica.com/staff-directory/
Index [6] Rank: 26 Color: BLACK Score: 152 URL: http://arstechnica.com/?view=grid
Index [7] Rank: 25 Color: RED Score: 154 URL: https://arstechnica.com
Index [8] Rank: 24 Color: BLACK Score: 172 URL: https://arstechnica.com/science/
Index [9] Rank: 23 Color: BLACK Score: 176 URL: https://arstechnica.com/store/
Index [10] Rank: 22 Color: RED Score: 181 URL: https://arstechnica.com/store/product/subscriptions/
Index [11] Rank: 21 Color: BLACK Score: 185 URL: https://arstechnica.com/advertise-with-us/
Index [12] Rank: 20 Color: RED Score: 190 URL: https://arstechnica.com/civis/ucp.php?mode=login&return_to=%2F
Index [13] Rank: 19 Color: BLACK Score: 192 URL: https://arstechnica.com/rss-feeds/
Index [14] Rank: 18 Color: BLACK Score: 195 URL: https://arstechnica.com/reviews/
Index [15] Rank: 17 Color: RED Score: 199 URL: https://arstechnica.com/cars/
Index [16] Rank: 16 Color: BLACK Score: 201 URL: https://arstechnica.com/civis/ucp.php?mode=sendpassword
Index [17] Rank: 15 Color: RED Score: 213 URL: http://arstechnica.com/?view=archive
Index [18] Rank: 14 Color: BLACK Score: 213 URL: http://arstechnica.com/?theme=light
Index [19] Rank: 13 Color: BLACK Score: 216 URL: https://arstechnica.com/tech-policy/
Index [20] Rank: 12 Color: BLACK Score: 216 URL: https://arstechnica.com/gaming/
Index [21] Rank: 11 Color: RED Score: 222 URL: https://arstechnica.com/reprints/
Index [22] Rank: 10 Color: RED Score: 227 URL: https://arstechnica.com/#main
Index [23] Rank: 9 Color: BLACK Score: 228 URL: https://arstechnica.com/gadgets/
Index [24] Rank: 8 Color: RED Score: 231 URL: http://arstechnica.com/?theme=dark
Index [25] Rank: 7 Color: BLACK Score: 236 URL: https://arstechnica.com/contact-us/
Index [26] Rank: 6 Color: BLACK Score: 240 URL: https://arstechnica.com/search/
Index [27] Rank: 5 Color: BLACK Score: 246 URL: https://arstechnica.com/
Index [28] Rank: 4 Color: RED Score: 262 URL: https://arstechnica.com/features/
Index [29] Rank: 3 Color: RED Score: 273 URL: https://arstechnica.com/#site-menu
Index [30] Rank: 2 Color: BLACK Score: 373 URL: https://arstechnica.com/about-us/
Index [31] Rank: 1 Color: RED Score: 999 URL: mikewu.com

```

The program then shows the user the sorted list of the URLs again along with the 2 newly inserted URLs. We can see that the scores are indeed are in ascending order.

```
Enter the Rank of the URL that you want to delete:
```

```
8
```

The program then asks the user to enter the rank of the URL that they want to delete.

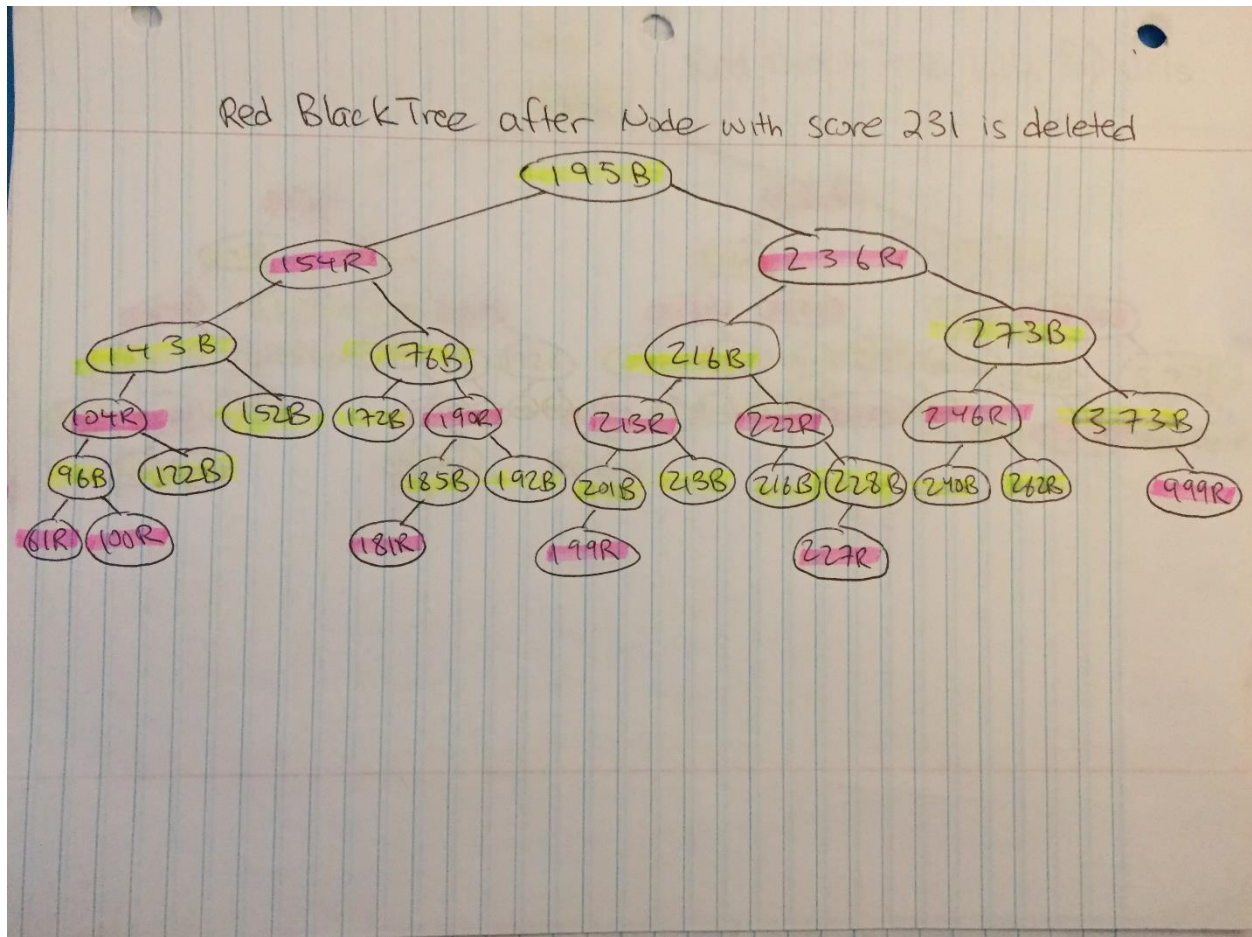
In this case the user types 8 so when we print out the tree again, it should have the URL <http://arstechnica.com/?theme=dark> omitted from the tree.

```

Binary Search Tree in preorder tree walk after deleted URL **(Used to check structure of the RB tree)**
Color: BLACK Score: 195 URL: https://arstechnica.com/reviews/
Color: RED Score: 154 URL: https://arstechnica.com
Color: BLACK Score: 143 URL: https://arstechnica.com/staff-directory/
Color: RED Score: 104 URL: https://arstechnica.com/information-technology/
Color: BLACK Score: 96 URL: http://video.arstechnica.com/
Color: RED Score: 61 URL: https://arstechnica.com/civis/
Color: RED Score: 100 URL: csl46.com
Color: BLACK Score: 122 URL: https://arstechnica.com/?view=mobile
Color: BLACK Score: 152 URL: http://arstechnica.com/?view=grid
Color: BLACK Score: 176 URL: https://arstechnica.com/store/
Color: BLACK Score: 172 URL: https://arstechnica.com/science/
Color: RED Score: 190 URL: https://arstechnica.com/civis/ucp.php?mode=login&return_to=%2F
Color: BLACK Score: 185 URL: https://arstechnica.com/advertise-with-us/
Color: RED Score: 181 URL: https://arstechnica.com/store/product/subscriptions/
Color: BLACK Score: 192 URL: https://arstechnica.com/rss-feeds/
Color: RED Score: 236 URL: https://arstechnica.com/contact-us/
Color: BLACK Score: 216 URL: https://arstechnica.com/tech-policy/
Color: RED Score: 213 URL: http://arstechnica.com/?view=archive
Color: BLACK Score: 201 URL: https://arstechnica.com/civis/ucp.php?mode=sendpassword
Color: RED Score: 199 URL: https://arstechnica.com/cars/
Color: BLACK Score: 213 URL: http://arstechnica.com/?theme=light
Color: RED Score: 222 URL: https://arstechnica.com/reprints/
Color: BLACK Score: 216 URL: https://arstechnica.com/gaming/
Color: BLACK Score: 228 URL: https://arstechnica.com/gadgets/
Color: RED Score: 227 URL: https://arstechnica.com/#main
Color: BLACK Score: 273 URL: https://arstechnica.com/#site-menu
Color: RED Score: 246 URL: https://arstechnica.com/
Color: BLACK Score: 240 URL: https://arstechnica.com/search/
Color: BLACK Score: 262 URL: https://arstechnica.com/features/
Color: BLACK Score: 373 URL: https://arstechnica.com/about-us/
Color: RED Score: 999 URL: mikewu.com

```

The red black tree then is printed in preorder tree walk so that I can draw the tree to see if it follows the red black tree structure.



We can indeed see that it follows the red black tree format even an URL was deleted from the red black tree.

The yellow highlight represents the black colored nodes and the pink highlight represents the red colored nodes.

```

Binary Search Tree in order tree walk after deleted URL **(Sorted List)**
Index [0] Rank: 31 Color: RED Score: 61 URL: https://arstechnica.com/civis/
Index [1] Rank: 30 Color: BLACK Score: 96 URL: http://video.arstechnica.com/
Index [2] Rank: 29 Color: RED Score: 100 URL: cs146.com
Index [3] Rank: 28 Color: RED Score: 104 URL: https://arstechnica.com/information-technology/
Index [4] Rank: 27 Color: BLACK Score: 122 URL: https://arstechnica.com/?view=mobile
Index [5] Rank: 26 Color: BLACK Score: 143 URL: https://arstechnica.com/staff-directory/
Index [6] Rank: 25 Color: BLACK Score: 152 URL: http://arstechnica.com/?view=grid
Index [7] Rank: 24 Color: RED Score: 154 URL: https://arstechnica.com
Index [8] Rank: 23 Color: BLACK Score: 172 URL: https://arstechnica.com/science/
Index [9] Rank: 22 Color: BLACK Score: 176 URL: https://arstechnica.com/store/
Index [10] Rank: 21 Color: RED Score: 181 URL: https://arstechnica.com/store/product/subscriptions/
Index [11] Rank: 20 Color: BLACK Score: 185 URL: https://arstechnica.com/advertise-with-us/
Index [12] Rank: 19 Color: RED Score: 190 URL: https://arstechnica.com/civis/ucp.php?mode=login&return_to=%2F
Index [13] Rank: 18 Color: BLACK Score: 192 URL: https://arstechnica.com/rss-feeds/
Index [14] Rank: 17 Color: BLACK Score: 195 URL: https://arstechnica.com/reviews/
Index [15] Rank: 16 Color: RED Score: 199 URL: https://arstechnica.com/cars/
Index [16] Rank: 15 Color: BLACK Score: 201 URL: https://arstechnica.com/civis/ucp.php?mode=sendpassword
Index [17] Rank: 14 Color: RED Score: 213 URL: http://arstechnica.com/?view=archive
Index [18] Rank: 13 Color: BLACK Score: 213 URL: http://arstechnica.com/?theme=light
Index [19] Rank: 12 Color: BLACK Score: 216 URL: https://arstechnica.com/tech-policy/
Index [20] Rank: 11 Color: BLACK Score: 216 URL: https://arstechnica.com/gaming/
Index [21] Rank: 10 Color: RED Score: 222 URL: https://arstechnica.com/reprints/
Index [22] Rank: 9 Color: RED Score: 227 URL: https://arstechnica.com/#main
Index [23] Rank: 8 Color: BLACK Score: 228 URL: https://arstechnica.com/gadgets/
Index [24] Rank: 7 Color: RED Score: 236 URL: https://arstechnica.com/contact-us/
Index [25] Rank: 6 Color: BLACK Score: 240 URL: https://arstechnica.com/search/
Index [26] Rank: 5 Color: RED Score: 246 URL: https://arstechnica.com/
Index [27] Rank: 4 Color: BLACK Score: 262 URL: https://arstechnica.com/features/
Index [28] Rank: 3 Color: BLACK Score: 273 URL: https://arstechnica.com/#site-menu
Index [29] Rank: 2 Color: BLACK Score: 373 URL: https://arstechnica.com/about-us/
Index [30] Rank: 1 Color: RED Score: 999 URL: mikewu.com

```

The program then shows the user the sorted list of the URLs again after we have deleted an URL from it. We can see that the scores are indeed are in ascending order.

In this case we can clearly see that the old rank 8 which had the URL: <http://arstechnica.com/?theme=dark> is indeed omitted from the tree and not in the list anymore.

```

Enter a pagerank to search for the URL: 8
The score and Url with pagerank 8 is https://arstechnica.com/gadgets/ with a score of 228

```

We ask the user to input the pagerank of the URL that they want to get the URL and score of.

In this case the user inputs the pagerank 8 and we can see that it does indeed match with the pagerank from the previous sorted list above. Notice that it is not the previous rank 8 that we have already deleted from the tree.

```

Do you want to search for any other word? (Y/N): y
What keyword do you want to search?: tree
Printing out the URLs in the order that is inserted into the Red BLaCk Tree
Score: 320 URL: http://arstechnica.com/?theme=dark
Score: 156 URL: https://arstechnica.com/store/
Score: 162 URL: https://arstechnica.com/search/
Score: 235 URL: http://arstechnica.com/?view=archive
Score: 188 URL: https://arstechnica.com/reviews/
Score: 235 URL: http://arstechnica.com/?theme=light
Score: 142 URL: https://arstechnica.com/about-us/
Score: 247 URL: https://arstechnica.com/staff-directory/
Score: 241 URL: https://arstechnica.com/civis/ucp.php?mode=sendpassword
Score: 202 URL: https://arstechnica.com/civis/ucp.php?mode=login&return_to=%2F
Score: 166 URL: https://arstechnica.com
Score: 206 URL: https://arstechnica.com/rss-feeds/
Score: 294 URL: https://arstechnica.com/science/
Score: 240 URL: https://arstechnica.com/advertise-with-us/
Score: 219 URL: https://arstechnica.com/?view=mobile
Score: 155 URL: https://arstechnica.com/contact-us/
Score: 262 URL: https://arstechnica.com/
Score: 274 URL: http://arstechnica.com/?view=grid
Score: 203 URL: https://arstechnica.com/tech-policy/
Score: 308 URL: https://arstechnica.com/gadgets/
Score: 158 URL: https://arstechnica.com/civis/
Score: 85 URL: https://arstechnica.com/#site-menu
Score: 253 URL: https://arstechnica.com/gaming/
Score: 249 URL: https://arstechnica.com/store/product/subscriptions/
Score: 203 URL: https://arstechnica.com/features/
Score: 203 URL: https://arstechnica.com/information-technology/
Score: 152 URL: https://arstechnica.com/reprints/
Score: 264 URL: https://arstechnica.com/cars/
Score: 277 URL: https://arstechnica.com/#main
Score: 243 URL: http://video.arstechnica.com/

```

We ask the user if they want to search for another word and in this case the user wants to search for the word tree now. The steps are the same as before so I will skip the explanation for this part of the program but I will provide the screenshots of what happens.

```

Red Black Tree in pre order tree walk **(Used to check the structure of the RB Tree)**
Color: BLACK Score: 235 URL: http://arstechnica.com/?view=archive
Color: RED Score: 188 URL: https://arstechnica.com/reviews/
Color: BLACK Score: 162 URL: https://arstechnica.com/search/
Color: RED Score: 155 URL: https://arstechnica.com/contact-us/
Color: BLACK Score: 142 URL: https://arstechnica.com/about-us/
Color: RED Score: 85 URL: https://arstechnica.com/#site-menu
Color: RED Score: 152 URL: https://arstechnica.com/reprints/
Color: BLACK Score: 156 URL: https://arstechnica.com/store/
Color: RED Score: 158 URL: https://arstechnica.com/civis/
Color: BLACK Score: 166 URL: https://arstechnica.com
Color: BLACK Score: 206 URL: https://arstechnica.com/rss-feeds/
Color: RED Score: 203 URL: https://arstechnica.com/tech-policy/
Color: BLACK Score: 202 URL: https://arstechnica.com/civis/ucp.php?mode=login&return_to=%2F
Color: BLACK Score: 203 URL: https://arstechnica.com/features/
Color: RED Score: 203 URL: https://arstechnica.com/information-technology/
Color: BLACK Score: 219 URL: https://arstechnica.com/?view=mobile
Color: RED Score: 262 URL: https://arstechnica.com/
Color: BLACK Score: 247 URL: https://arstechnica.com/staff-directory/
Color: RED Score: 240 URL: https://arstechnica.com/advertise-with-us/
Color: BLACK Score: 235 URL: http://arstechnica.com/?theme=light
Color: BLACK Score: 241 URL: https://arstechnica.com/civis/ucp.php?mode=sendpassword
Color: RED Score: 243 URL: http://video.arstechnica.com/
Color: BLACK Score: 253 URL: https://arstechnica.com/gaming/
Color: RED Score: 249 URL: https://arstechnica.com/store/product/subscriptions/
Color: BLACK Score: 294 URL: https://arstechnica.com/science/
Color: BLACK Score: 274 URL: http://arstechnica.com/?view=grid
Color: RED Score: 264 URL: https://arstechnica.com/cars/
Color: RED Score: 277 URL: https://arstechnica.com/#main
Color: BLACK Score: 320 URL: http://arstechnica.com/?theme=dark
Color: RED Score: 308 URL: https://arstechnica.com/gadgets/

```

```

Red Black Tree in order tree walk **(Sorted List)**
Index [0] Rank: 30 Color: RED Score: 85 URL: https://arstechnica.com/#site-menu
Index [1] Rank: 29 Color: BLACK Score: 142 URL: https://arstechnica.com/about-us/
Index [2] Rank: 28 Color: RED Score: 152 URL: https://arstechnica.com/reprints/
Index [3] Rank: 27 Color: RED Score: 155 URL: https://arstechnica.com/contact-us/
Index [4] Rank: 26 Color: BLACK Score: 156 URL: https://arstechnica.com/store/
Index [5] Rank: 25 Color: RED Score: 158 URL: https://arstechnica.com/civis/
Index [6] Rank: 24 Color: BLACK Score: 162 URL: https://arstechnica.com/search/
Index [7] Rank: 23 Color: BLACK Score: 166 URL: https://arstechnica.com
Index [8] Rank: 22 Color: RED Score: 188 URL: https://arstechnica.com/reviews/
Index [9] Rank: 21 Color: BLACK Score: 202 URL: https://arstechnica.com/civis/ucp.php?mode=login&return_to=%2F
Index [10] Rank: 20 Color: RED Score: 203 URL: https://arstechnica.com/tech-policy/
Index [11] Rank: 19 Color: BLACK Score: 203 URL: https://arstechnica.com/features/
Index [12] Rank: 18 Color: RED Score: 203 URL: https://arstechnica.com/information-technology/
Index [13] Rank: 17 Color: BLACK Score: 206 URL: https://arstechnica.com/rss-feeds/
Index [14] Rank: 16 Color: BLACK Score: 219 URL: https://arstechnica.com/?view=mobile
Index [15] Rank: 15 Color: BLACK Score: 235 URL: http://arstechnica.com/?view=archive
Index [16] Rank: 14 Color: BLACK Score: 235 URL: http://arstechnica.com/?theme=light
Index [17] Rank: 13 Color: RED Score: 240 URL: https://arstechnica.com/advertise-with-us/
Index [18] Rank: 12 Color: BLACK Score: 241 URL: https://arstechnica.com/civis/ucp.php?mode=sendpassword
Index [19] Rank: 11 Color: RED Score: 243 URL: http://video.arstechnica.com/
Index [20] Rank: 10 Color: BLACK Score: 247 URL: https://arstechnica.com/staff-directory/
Index [21] Rank: 9 Color: RED Score: 249 URL: https://arstechnica.com/store/product/subscriptions/
Index [22] Rank: 8 Color: BLACK Score: 253 URL: https://arstechnica.com/gaming/
Index [23] Rank: 7 Color: RED Score: 262 URL: https://arstechnica.com/
Index [24] Rank: 6 Color: RED Score: 264 URL: https://arstechnica.com/cars/
Index [25] Rank: 5 Color: BLACK Score: 274 URL: http://arstechnica.com/?view=grid
Index [26] Rank: 4 Color: RED Score: 277 URL: https://arstechnica.com/#main
Index [27] Rank: 3 Color: BLACK Score: 294 URL: https://arstechnica.com/science/
Index [28] Rank: 2 Color: RED Score: 308 URL: https://arstechnica.com/gadgets/
Index [29] Rank: 1 Color: BLACK Score: 320 URL: http://arstechnica.com/?theme=dark

```

```

What is the URL you want to add?
mikewu.com
What is the score for the URL?
999
Do you want to add more websites? (Y/N)
N

```

```

Red Black Tree in preorder tree walk with the added URLs **(Used to check structure of the RB tree)**
Color: BLACK Score: 235 URL: http://arstechnica.com/?view=archive
Color: RED Score: 188 URL: https://arstechnica.com/reviews/
Color: BLACK Score: 162 URL: https://arstechnica.com/search/
Color: RED Score: 155 URL: https://arstechnica.com/contact-us/
Color: BLACK Score: 142 URL: https://arstechnica.com/about-us/
Color: RED Score: 85 URL: https://arstechnica.com/#site-menu
Color: RED Score: 152 URL: https://arstechnica.com/reprints/
Color: BLACK Score: 156 URL: https://arstechnica.com/store/
Color: RED Score: 158 URL: https://arstechnica.com/civis/
Color: BLACK Score: 166 URL: https://arstechnica.com
Color: BLACK Score: 206 URL: https://arstechnica.com/rss-feeds/
Color: RED Score: 203 URL: https://arstechnica.com/tech-policy/
Color: BLACK Score: 202 URL: https://arstechnica.com/civis/ucp.php?mode=login&return_to=%2F
Color: BLACK Score: 203 URL: https://arstechnica.com/features/
Color: RED Score: 203 URL: https://arstechnica.com/information-technology/
Color: BLACK Score: 219 URL: https://arstechnica.com/?view=mobile
Color: RED Score: 262 URL: https://arstechnica.com/
Color: BLACK Score: 247 URL: https://arstechnica.com/staff-directory/
Color: RED Score: 240 URL: https://arstechnica.com/advertise-with-us/
Color: BLACK Score: 235 URL: http://arstechnica.com/?theme=light
Color: BLACK Score: 241 URL: https://arstechnica.com/civis/ucp.php?mode=sendpassword
Color: RED Score: 243 URL: http://video.arstechnica.com/
Color: BLACK Score: 253 URL: https://arstechnica.com/gaming/
Color: RED Score: 249 URL: https://arstechnica.com/store/product/subscriptions/
Color: BLACK Score: 294 URL: https://arstechnica.com/science/
Color: BLACK Score: 274 URL: http://arstechnica.com/?view=grid
Color: RED Score: 264 URL: https://arstechnica.com/cars/
Color: RED Score: 277 URL: https://arstechnica.com/#main
Color: BLACK Score: 320 URL: http://arstechnica.com/?theme=dark
Color: RED Score: 308 URL: https://arstechnica.com/gadgets/
Color: RED Score: 999 URL: mikewu.com

```



```

Red Black Tree in order tree walk with the added URLs **(Sorted List)**
Index [0] Rank: 31 Color: RED Score: 85 URL: https://arstechnica.com/#site-menu
Index [1] Rank: 30 Color: BLACK Score: 142 URL: https://arstechnica.com/about-us/
Index [2] Rank: 29 Color: RED Score: 152 URL: https://arstechnica.com/reprints/
Index [3] Rank: 28 Color: RED Score: 155 URL: https://arstechnica.com/contact-us/
Index [4] Rank: 27 Color: BLACK Score: 156 URL: https://arstechnica.com/store/
Index [5] Rank: 26 Color: RED Score: 158 URL: https://arstechnica.com/civis/
Index [6] Rank: 25 Color: BLACK Score: 162 URL: https://arstechnica.com/search/
Index [7] Rank: 24 Color: BLACK Score: 166 URL: https://arstechnica.com
Index [8] Rank: 23 Color: RED Score: 188 URL: https://arstechnica.com/reviews/
Index [9] Rank: 22 Color: BLACK Score: 202 URL: https://arstechnica.com/civis/ucp.php?mode=login&return_to=%2F
Index [10] Rank: 21 Color: RED Score: 203 URL: https://arstechnica.com/tech-policy/
Index [11] Rank: 20 Color: BLACK Score: 203 URL: https://arstechnica.com/features/
Index [12] Rank: 19 Color: RED Score: 203 URL: https://arstechnica.com/information-technology/
Index [13] Rank: 18 Color: BLACK Score: 206 URL: https://arstechnica.com/rss-feeds/
Index [14] Rank: 17 Color: BLACK Score: 219 URL: https://arstechnica.com/?view=mobile
Index [15] Rank: 16 Color: BLACK Score: 235 URL: http://arstechnica.com/?view=archive
Index [16] Rank: 15 Color: BLACK Score: 235 URL: http://arstechnica.com/?theme=light
Index [17] Rank: 14 Color: RED Score: 240 URL: https://arstechnica.com/advertise-with-us/
Index [18] Rank: 13 Color: BLACK Score: 241 URL: https://arstechnica.com/civis/ucp.php?mode=sendpassword
Index [19] Rank: 12 Color: RED Score: 243 URL: http://video.arstechnica.com/
Index [20] Rank: 11 Color: BLACK Score: 247 URL: https://arstechnica.com/staff-directory/
Index [21] Rank: 10 Color: RED Score: 249 URL: https://arstechnica.com/store/product/subscriptions/
Index [22] Rank: 9 Color: BLACK Score: 253 URL: https://arstechnica.com/gaming/
Index [23] Rank: 8 Color: RED Score: 262 URL: https://arstechnica.com/
Index [24] Rank: 7 Color: RED Score: 264 URL: https://arstechnica.com/cars/
Index [25] Rank: 6 Color: BLACK Score: 274 URL: http://arstechnica.com/?view=grid
Index [26] Rank: 5 Color: RED Score: 277 URL: https://arstechnica.com/#main
Index [27] Rank: 4 Color: BLACK Score: 294 URL: https://arstechnica.com/science/
Index [28] Rank: 3 Color: RED Score: 308 URL: https://arstechnica.com/gadgets/
Index [29] Rank: 2 Color: BLACK Score: 320 URL: http://arstechnica.com/?theme=dark
Index [30] Rank: 1 Color: RED Score: 999 URL: mikewu.com

```

Enter the Rank of the URL that you want to delete:

2

```

Binary Search Tree in preorder tree walk after deleted URL **(Used to check structure of the RB tree)**
Color: BLACK Score: 235 URL: http://arstechnica.com/?view=archive
Color: RED Score: 188 URL: https://arstechnica.com/reviews/
Color: BLACK Score: 162 URL: https://arstechnica.com/search/
Color: RED Score: 155 URL: https://arstechnica.com/contact-us/
Color: BLACK Score: 142 URL: https://arstechnica.com/about-us/
Color: RED Score: 85 URL: https://arstechnica.com/#site-menu
Color: RED Score: 152 URL: https://arstechnica.com/reprints/
Color: BLACK Score: 156 URL: https://arstechnica.com/store/
Color: RED Score: 158 URL: https://arstechnica.com/civis/
Color: BLACK Score: 166 URL: https://arstechnica.com
Color: BLACK Score: 206 URL: https://arstechnica.com/rss-feeds/
Color: RED Score: 203 URL: https://arstechnica.com/tech-policy/
Color: BLACK Score: 202 URL: https://arstechnica.com/civis/ucp.php?mode=login&return_to=%2F
Color: BLACK Score: 203 URL: https://arstechnica.com/features/
Color: RED Score: 203 URL: https://arstechnica.com/information-technology/
Color: BLACK Score: 219 URL: https://arstechnica.com/?view=mobile
Color: RED Score: 262 URL: https://arstechnica.com/
Color: BLACK Score: 247 URL: https://arstechnica.com/staff-directory/
Color: RED Score: 240 URL: https://arstechnica.com/advertise-with-us/
Color: BLACK Score: 235 URL: http://arstechnica.com/?theme=light
Color: BLACK Score: 241 URL: https://arstechnica.com/civis/ucp.php?mode=sendpassword
Color: RED Score: 243 URL: http://video.arstechnica.com/
Color: BLACK Score: 253 URL: https://arstechnica.com/gaming/
Color: RED Score: 249 URL: https://arstechnica.com/store/product/subscriptions/
Color: BLACK Score: 294 URL: https://arstechnica.com/science/
Color: BLACK Score: 274 URL: http://arstechnica.com/?view=grid
Color: RED Score: 264 URL: https://arstechnica.com/cars/
Color: RED Score: 277 URL: https://arstechnica.com/#main
Color: BLACK Score: 999 URL: mikewu.com
Color: RED Score: 308 URL: https://arstechnica.com/gadgets/

```

```

Binary Search Tree in order tree walk after deleted URL **(Sorted List)**
Index [0] Rank: 30 Color: RED Score: 85 URL: https://arstechnica.com/#site-menu
Index [1] Rank: 29 Color: BLACK Score: 142 URL: https://arstechnica.com/about-us/
Index [2] Rank: 28 Color: RED Score: 152 URL: https://arstechnica.com/reprints/
Index [3] Rank: 27 Color: RED Score: 155 URL: https://arstechnica.com/contact-us/
Index [4] Rank: 26 Color: BLACK Score: 156 URL: https://arstechnica.com/store/
Index [5] Rank: 25 Color: RED Score: 158 URL: https://arstechnica.com/civis/
Index [6] Rank: 24 Color: BLACK Score: 162 URL: https://arstechnica.com/search/
Index [7] Rank: 23 Color: BLACK Score: 166 URL: https://arstechnica.com
Index [8] Rank: 22 Color: RED Score: 188 URL: https://arstechnica.com/reviews/
Index [9] Rank: 21 Color: BLACK Score: 202 URL: https://arstechnica.com/civis/ucp.php?mode=login&return_to=%2F
Index [10] Rank: 20 Color: RED Score: 203 URL: https://arstechnica.com/tech-policy/
Index [11] Rank: 19 Color: BLACK Score: 203 URL: https://arstechnica.com/features/
Index [12] Rank: 18 Color: RED Score: 203 URL: https://arstechnica.com/information-technology/
Index [13] Rank: 17 Color: BLACK Score: 206 URL: https://arstechnica.com/rss-feeds/
Index [14] Rank: 16 Color: BLACK Score: 219 URL: https://arstechnica.com/?view=mobile
Index [15] Rank: 15 Color: BLACK Score: 235 URL: http://arstechnica.com/?view=archive
Index [16] Rank: 14 Color: BLACK Score: 235 URL: http://arstechnica.com/?theme=light
Index [17] Rank: 13 Color: RED Score: 240 URL: https://arstechnica.com/advertise-with-us/
Index [18] Rank: 12 Color: BLACK Score: 241 URL: https://arstechnica.com/civis/ucp.php?mode=sendpassword
Index [19] Rank: 11 Color: RED Score: 243 URL: http://video.arstechnica.com/
Index [20] Rank: 10 Color: BLACK Score: 247 URL: https://arstechnica.com/staff-directory/
Index [21] Rank: 9 Color: RED Score: 249 URL: https://arstechnica.com/store/product/subscriptions/
Index [22] Rank: 8 Color: BLACK Score: 253 URL: https://arstechnica.com/gaming/
Index [23] Rank: 7 Color: RED Score: 262 URL: https://arstechnica.com/
Index [24] Rank: 6 Color: RED Score: 264 URL: https://arstechnica.com/cars/
Index [25] Rank: 5 Color: BLACK Score: 274 URL: http://arstechnica.com/?view=grid
Index [26] Rank: 4 Color: RED Score: 277 URL: https://arstechnica.com/#main
Index [27] Rank: 3 Color: BLACK Score: 294 URL: https://arstechnica.com/science/
Index [28] Rank: 2 Color: RED Score: 308 URL: https://arstechnica.com/gadgets/
Index [29] Rank: 1 Color: BLACK Score: 999 URL: mikewu.com

```

```

Enter a pagerank to search for the URL: 1
The score and Url with pagerank 1 is mikewu.com with a score of 999

```

```

Do you want to search for any other word? (Y/N): N

The Top 10 Unique Searches:
apple
tree

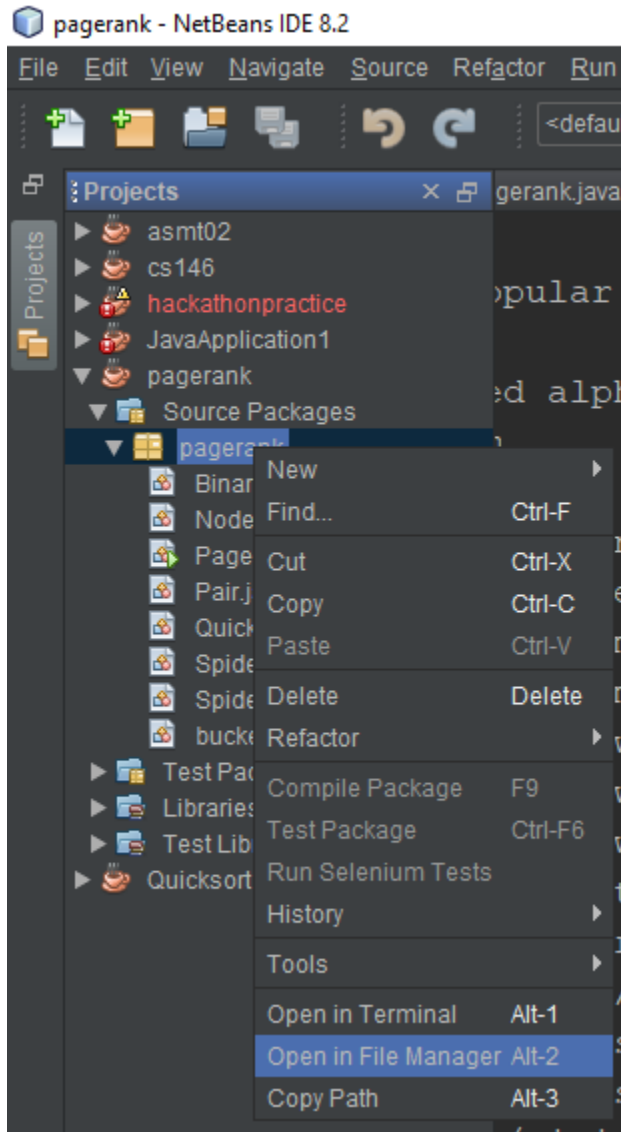
BUILD SUCCESSFUL (total time: 52 minutes 53 seconds)

```







The program then asks the user if they want to search for another word. In this case the user types N for no and then the program ends by printing out the top 10 unique searches. In this case however, we only searched for two words so it prints out only two. We can see that it does match with the previous words that we had inputted, which are apple and tree.

4)The procedure (step by step) of how to unzip your files, install your application, and run/test your codes.

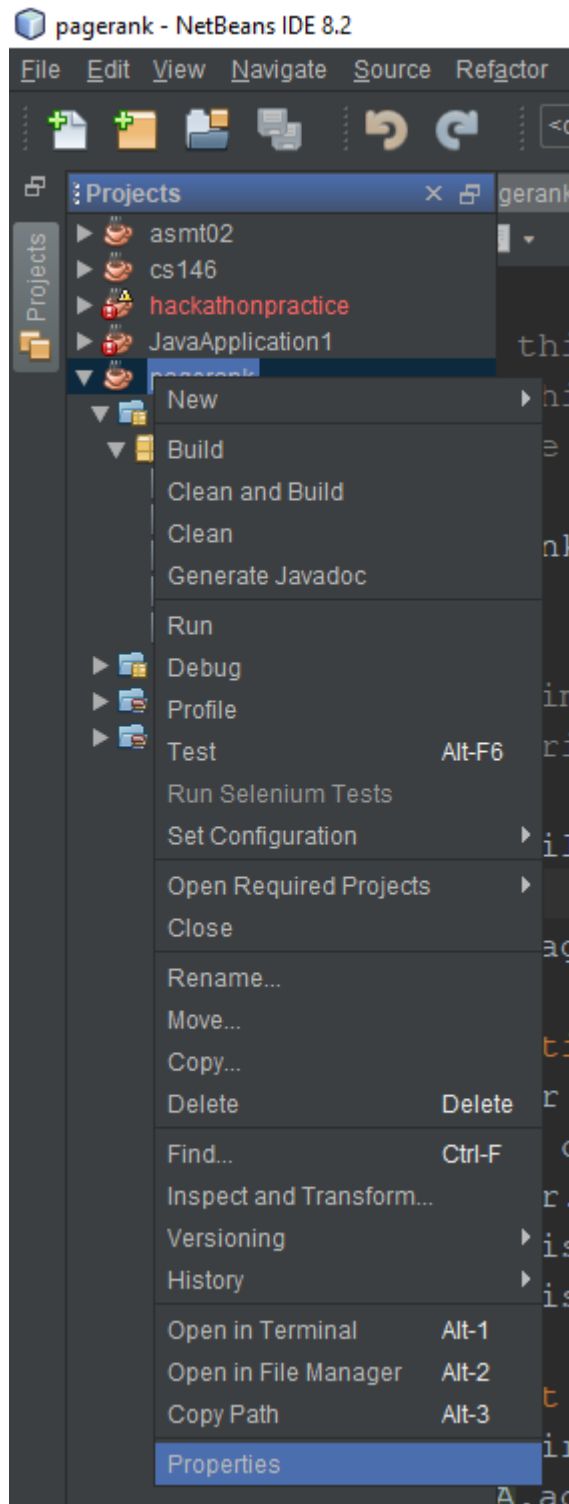
I am using netbeans to write my code. First begin by unzipping the files. Drag out everything from the folder and drag it into your desktop. Then open up your IDE's file path.



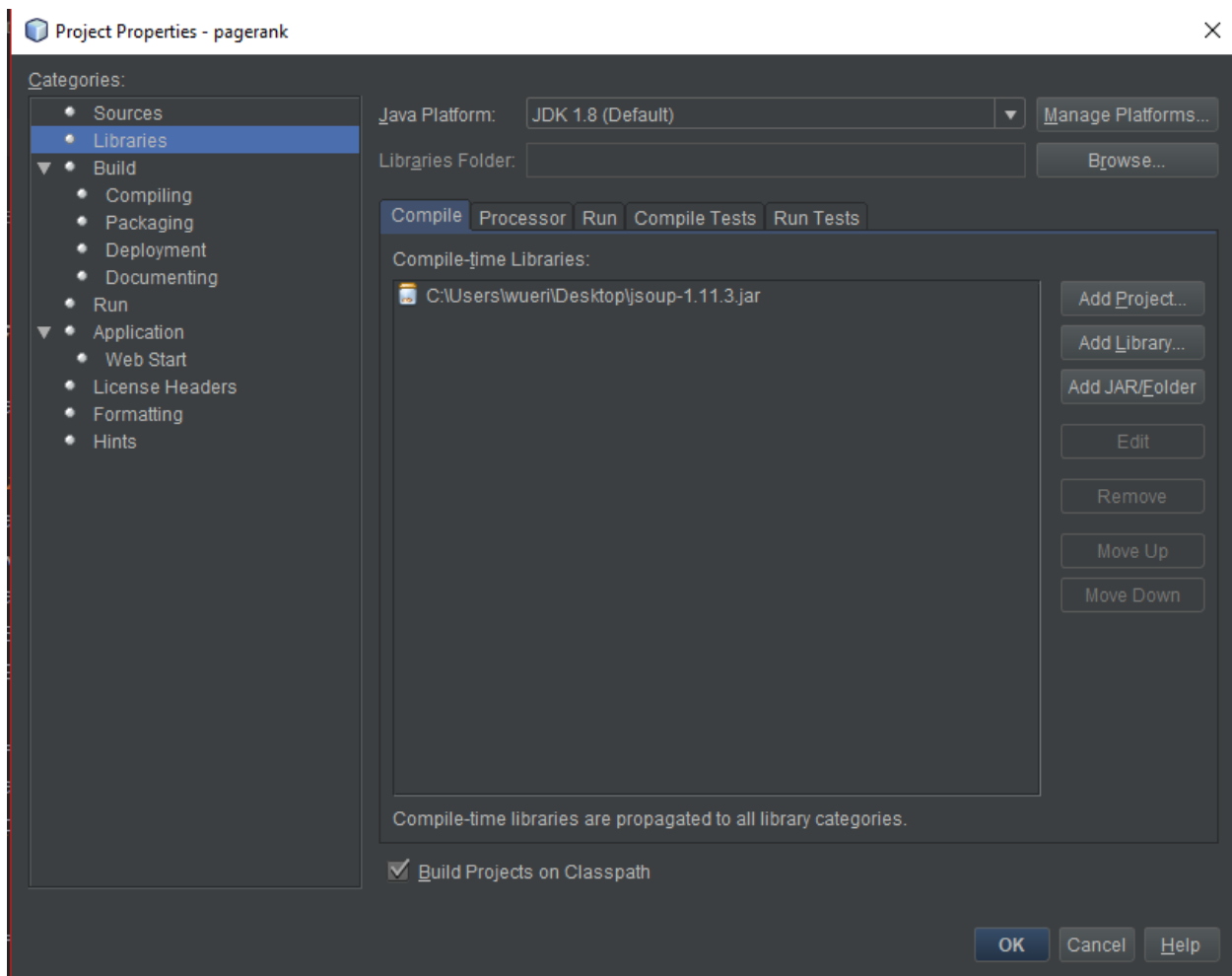
Then drag everything from the code folder into the file path.

OS (C:) > Users > wueri > Documents > NetBeansProjects > pagerank > src > pagerank				Search pagerank
Name	Date modified	Type	Size	
 Node.java	11/27/2018 4:46 PM	JAVA File	2 KB	
 Pagerank.java	11/30/2018 6:59 PM	JAVA File	6 KB	
 Pair.java	11/27/2018 4:46 PM	JAVA File	2 KB	
 RedBlackTree.java	11/30/2018 6:59 PM	JAVA File	13 KB	
 Spider.java	11/27/2018 11:28 PM	JAVA File	3 KB	
 SpiderLeg.java	11/27/2018 11:11 PM	JAVA File	4 KB	

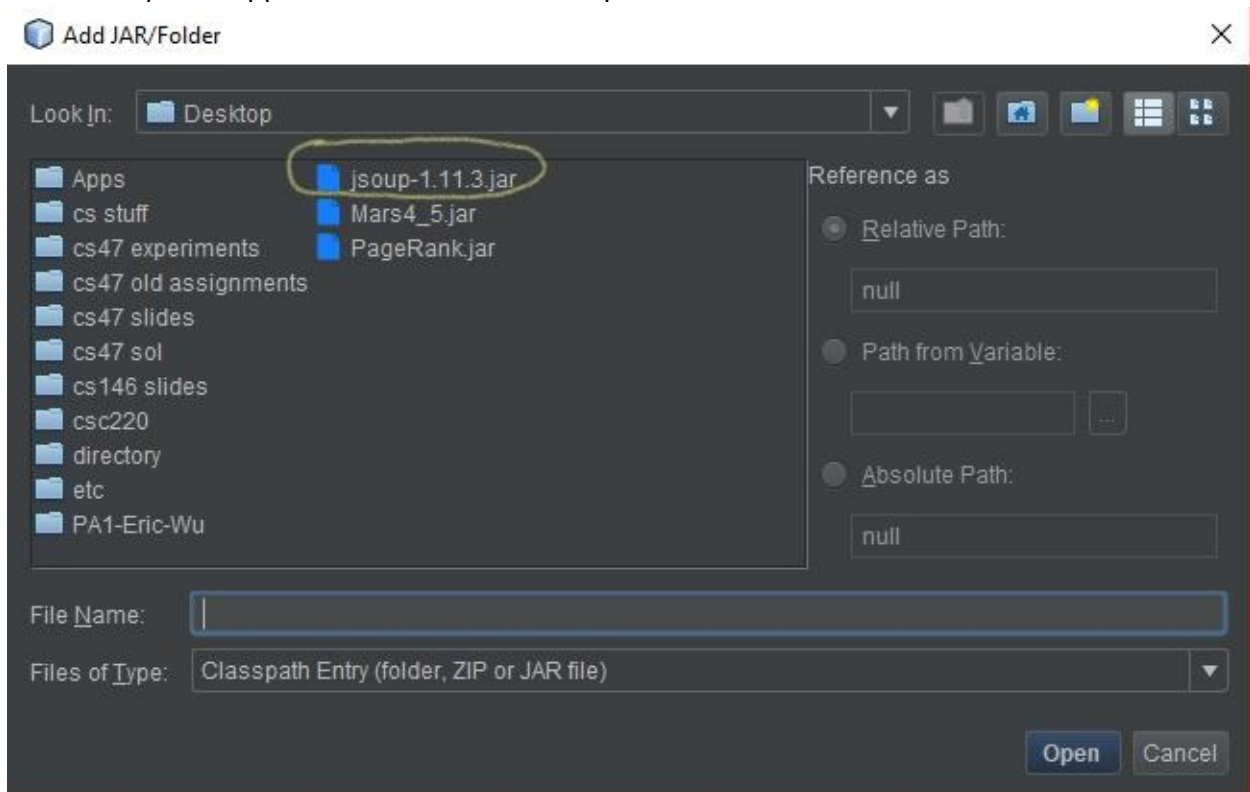
Right click on the project and click on properties.



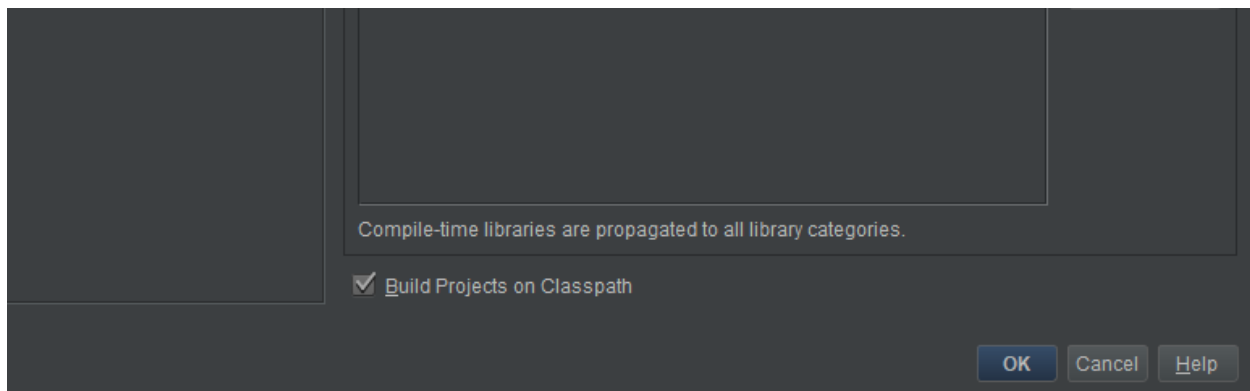
Click on libraries on the left hand side and then click on the Add JAR/Folder on the right side.



Select the jsoup-1.11.3.jar file that you have unzipped along with the code. It can be found in the desktop if unzipped the files to the desktop.



Then click on the Ok button.



You are done with setting up the program! Now proceed to run the main class, Pagerank.java.

To Run the Jar file make sure it is in you desktop.

Then open command prompt and type

```
cd desktop
java -jar PageRank.jar
```

Overview on how to use the program

The program begins by asking user to input a keyword to test the redblacktree.

Then it ask user to add urls along with scores to add to the redblacktree

Then it ask user to enter the pagerank of the url they want to delete

Then it ask the user to enter the pagerank of the url they want to check

The program then ask you if you want to search for any other word. If you type Y, it will repeat the steps over again, but if you type N, the program will move on to the next part.

The program ends by printing out the top 10 unique searches.

5) Problems encountered during the implementation

I have still encountered many problems while trying to write this program even though I was building on old code. Because I didn't want to write the redblacktree code from scratch I just added it to the old Binary search tree I wrote because it was similar. This might not have been a good idea because I encountered a lot of errors and I spent a lot of time trying to debug it. The first problem I encountered is the null error from the redblacktree. Because I did not implement the redblacktree from scratch, I had to meticulously go through the code in order to find out what I did wrong from the old bst code I wrote. I am also proud to say that the debugger was my best friend while doing the programming assignment because it helped me find out the reason why I had the null error in the program. It turned out that a line of code completely messed up my program. Another problem I had was that the program was not running correctly. Sometimes the program will work correctly, but sometimes it will also not print out anything and the program will be continuously running. I had no idea what was wrong since I had not work with nodes in quite a while so I had to bring out my old textbook and read that until I got how to do it. I also had to refer back to my professor's slides and book many times before I got the correct java code down. It turned out that I have implemented the the cases wrongly and therefore the program was doing something wrong. Another problem I had was the way I implemented the user interface. I used the code I wrote before for programming assignment 1 and it doesn't translate over perfectly. I had to remove many lines and add many so if I started blank, I might have finished the program even faster. It was also hard for me to understand what some part

of my old code did because of pointless variable names and lack of detailed comments. I have encountered many problems while trying to finish the programming assignment but I hope that I can learn from the mistakes I have made and use that experience and apply it to other projects I will be working on in the future.

6) Lessons Learned

I have learned many things from doing this programming assignment. For the third time, it would be time management. I have procrastinated on completing this assignment and that turned out to be a bad idea. For the past few days, I have been getting little sleep because I am trying to finish the project and also juggle with work from other classes and the fact that finals are coming up makes this even more stressful. The next time I am given an assignment, I will start it as soon as possible, so if I have any major bugs with the program like the null errors and the fact that nothing was printing out, I will have time to fix it and also ask my professor for help. I would also be less stressed out because I should be doing a decent amount of work on the assignment each day over the course of several week and not all in several days. Having dislocated my shoulder and therefore recovering from it not being able to type with two hands as fast as before also made this really difficult for me. I also learned the importance of writing good comments and using relevant variable names so that I can better understand the code the next time I look at it. Another thing I learned was that I need to practice my programming skills and concepts. I don't usually review concepts once I learned them so I usually forget how to do them. For example when making the red black tree insertion and deletion I had some trouble implementing it so that it doesn't have the null errors and the cases were also hard for me to implement correctly, so reviewing old concepts is really important in being a successful student and programmer. I am planning on working on projects alongside with classes so that I could always keep my programming knowledge and skills sharp.