

PA3 Hashing and Red Black Tree Report  
JianBin Wu

## Design and Implementation

### *Exercise 1 (Google Search Engine Most Popular Keywords)*

For Hashing with Chaining with a Division method, I used the LinkedList for the hash table and Linked List for each slot to handle the collision. In the chainedHashFunction method I converted each letter in the keyword into ascii code with base 26 and added them up. Then, convert the this result to binary and substring the last 16 bits (r0). After that, substring the significant bits from r0 and convert back to decimal. That will be the hashResu. It also allowed users to insert a keyword, search keyword and delete a keyword in the hash table.

For the Multiplication method, I used the LinkedList for the hash table and Linked List for each slot to handle the collision, too. In the Multiplication Hash Function method I converted each letter in the keyword into ascii code and added them up. Then, I use the multiplication process in bit format. I converted the result in binary and substring the r0. Then, substring the significant bits for r0 and convert it back to decimal. It will be the hashing result. Then, inserted the keyword with the hashing result. It also allowed users to insert a keyword, search a keyword and delete a keyword in the hash table.

For the Double Hashing method I used the LinkedList for the hash table and double hashing to handle the collision. The first hashing function will be  $h(k) = k \bmod 57$ . And the second hashing function  $h(x) = 1 + (k \bmod 13)$ . For each hashing function I converted each letter in the keyword into ascii code and added them up. Then, I insert the keyword with the first hashing function. And use  $h(k,i) = ((h1(k) + i \cdot h2(k)) \bmod m)$  to assign the keyword to another slot if a collision happened. It also allowed users to insert a keyword and delete a keyword in the hash table.

### *Exercise 2 (Google Search Engine PageRank Priority Queue)*

For the Google Search Engine queue I used the RB tree properties to build up a RB tree. I created a class named RBNode that constructs a node for the RB tree. Each node in the RBtree will have a url string, pagerank integer and color string. It has getter and setter methods in this class. Then, using the webCrawler to get the first 20 keywords. I also use the HashSet to store the score when generating a random pageRank score,so it won't have duplicate values. Then, create 20 RB nodes and assign each a score and url. After that, build up a RB tree using RBinsert method as well as RBInsertFixup method. Users are allow to insert nodes to and delete nodes from the RBTree and it will follow the RBTree properties.

## Classes, Subroutines and Function calls

HashingWithChaining class- Implementation of Hashing with Chaining method using a linked list. Using the Division method to insert the top 50 keywords into a size 17 of the hash table.

- HashingWithChaining() - Initialize hash table with size 17.
- chainedHashFunction(String s) - Hashing Function using division method.
- chainedHashInsert(String s) - Insert keyword into hash table.
- chainedHashSearch(String s) - Search a keyword in the hashing table.
- chainedHashDelete(String s) - Delete an existing keyword.

MultiplicationMethod class - Simulate a 16 bit-computer using the multiplication method to insert the top 50 keywords into a size 64 of the hash table.

- MultiplicationMethod() - Initialize hash table with size 64.
- multiplicationHashFunction(String s) - Hashing Function using multiplication method.
- multiplicationInsert(String s) - Insert keyword into hash table
- multiplicationSearch(String s) - Search a keyword in the hashing table.
- multiplicationDelete(String s) - Delete an existing keyword

DoubleHashing class - Implementation of Double Hashing method to insert the top 50 keywords into a size of 57 hash table.

- DoubleHashing() - Initialize hash table with size 57.
- firstHashingFunction(String s) - First Hashing Function.
- secondHashingFunction(String s) - Second hashing function.
- doubleHashingInsert(String s) - Insert keyword into hash table
- doubleHashingSearch(String s) - Search a keyword in the hashing table.

WebCrawler class - A web crawler to collect urls from Google Search engine using keywords.

- search() - This method start the search.
- getDomainName(String url) - The method will use pattern and matcher to extract the domain.
- getUrls() - This method get the set of urls result.
- crawl(String url) - This method will crawl the links and put them into a set to keep.
- searchForWord(String searchWord) - This method will check if the website contains a keyword.

RBNode<T> class - Node object for Red Black Tree . Contain a url, a page rank score and color.

- RBNode(T url,int pageRank) - Construct a Red Black Node with url and page rank score.
- getColor() - Get the color of the node.
- T getUrl() - Get the url of the node.
- getPageRank() - Get the page rank score of the node.
- setColor(String color) - Set the color of the node to different color.
- getLeftChild() - Get the left child of the node.
- getRightChild() - Get the right child of the node.
- setRightChild(RBNode<T> rightChild) - Set the right child of the node to other node.

- getParent() - Get the parent of the node.
- setParent(RBNode<T> parent) - Set the parent of the node to other node.
- setLeftChild(RBNode<T> leftChild) - Set the left child of the node to other node

## RBTree class

- RBTree() - Construct an empty Red Black Tree
- leftRotate(RBTree T , RBNode x) - Left rotation for a node
- rightRotate(RBTree T , RBNode x) - Right rotation for a node
- RBInsert(RBTree T, RBNode z) - Insert a node into the Red Black Tree
- RBInsertFixup(RBTree T, RBNode z) - Fix the Red Black Tree after the node inserted, so the Red Black Tree follow the properties.
- RBTransplant(RBTree T, RBNode u, RBNode v) - Transplant/Delete a node with other node
- TreeMinimum(RBNode x) - Find the left-most node of a node
- RBDelete(RBTree T, RBNode z) - Delete a node into the Red Black Tree
- RBDeleteFixup(RBTree T, RBNode x) - Fix the Red Black Tree after the node deleted, so the Red Black Tree follow the properties.
- RBSort (RBNode x) - Sorting the data of each node in a Red Black Tree from large to small (Reverse InorderWalk)
- iterativeTreeSearch(RBNode x , int pageRank) - Search for a node
- printRBTree(RBNode x) - print Red Black Tree in Inorder walk with parent, left, right children of the node.

## Self-testing Screenshots

*Exercise 1 (Google Search Engine Most Popular Keywords)*

## HashingWithChaining

```
public static void main (String args[]) {

    new HashingWithChaining();
    Scanner scanner = new Scanner(System.in);
    String[] keyWords = {"facebook", "youtube", "gmail", "google", "weather",
        "ebay", "yahoo", "walmart", "yahoo mail", "google translate", "google maps",
        "craigslist", "netflix", "google docs", "translate", "home depot", "news", "fox news",
        "calculator", "usps tracking", "cnn", "hotmail", "google drive", "maps", "paypal",
        "lowes", "instagram", "msn", "amazon prime", "target", "espn", "zillow", "bank of america",
        "wells fargo", "twitter", "google classroom", "indeed", "best buy", "speed test", "linkedin",
        "aol mail", "hulu", "you tube", "pinterest", "trump", "nba", "roblox", "capital one", "traductor"
    };
    //insert keywords to hash table
    for(String s : keyWords)
    {
        chainedHashInsert(s);
    }
}
```

In the beginning of the program, it generates a hash table (size 17) with 50 key words using the chainedHashInsert method.

```
/**
 * Insert keyword into hash table
 * @param s keyword string
 */
public static void chainedHashInsert(String s) {
    int hashResult = chainedHashFunction(s);
    hashTable.get(hashResult).add(s);
}
```

To insert the keyword, it gets the hashResult from chainedHashFunction and insert to the target slot.

```
*/
public static int chainedHashFunction(String s) {
    int stringLength = s.length();
    double hashResult = 0;
    double temp = 0;
    //Transform the keyword into ascii code, base 26 and add them up.
    while(stringLength != 0) {
        for (int i = 0; i < s.length(); i++) {
            temp = s.charAt(i);
            temp = temp * Math.pow(26, i);
            hashResult = hashResult + temp;
            stringLength--;
        }
    }
    //Hashing function  $h(k) = k \bmod m$ 
    hashResult = hashResult % 17;
    int hashResults = (int) Math.round(hashResult);
    return hashResults;
}
```

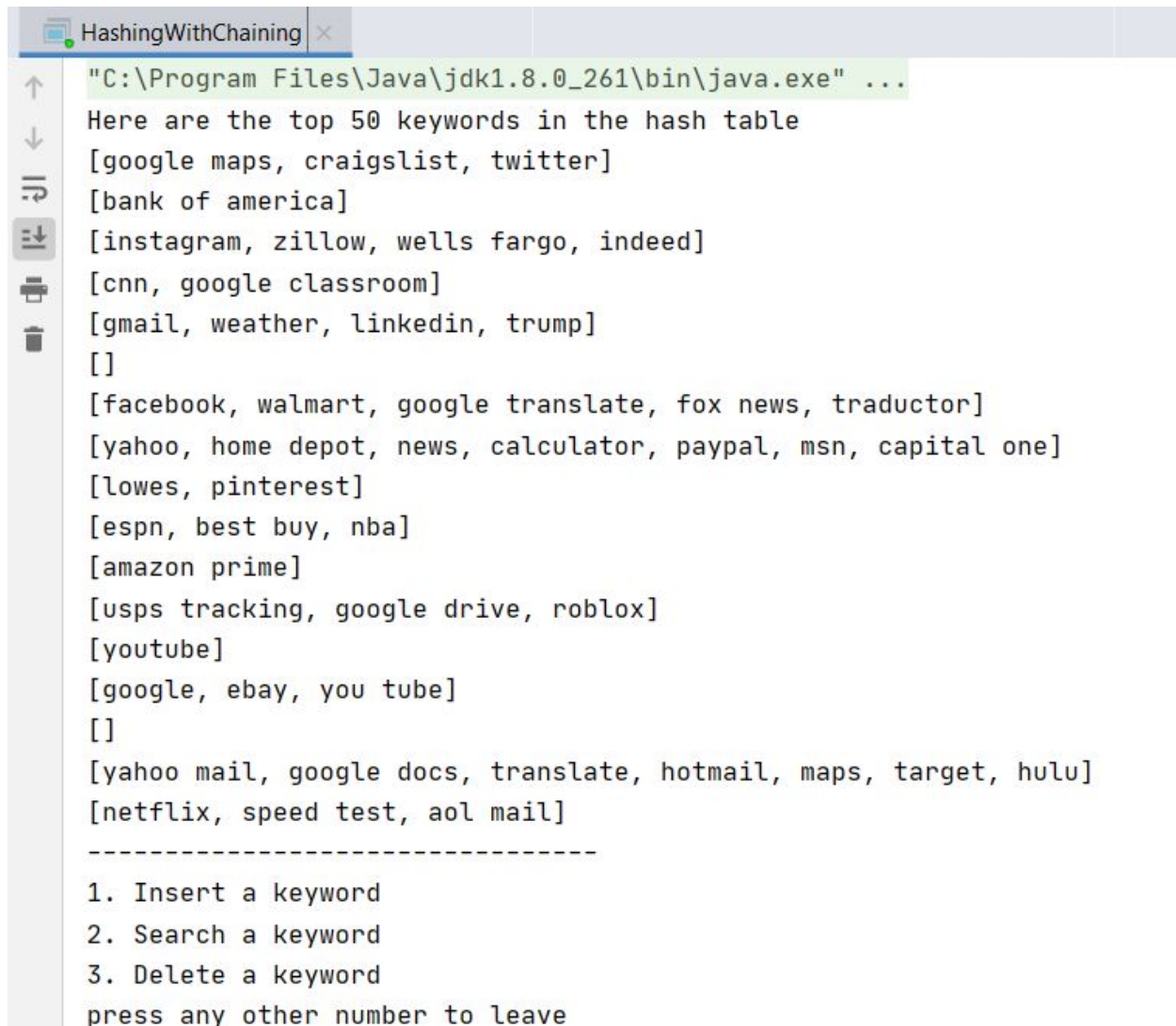
In the chainedHashingFunction, it converted each letter in the keyword into ascii code with base 26 and added them up. Then, using the hashing function  $h(k) = k \bmod m$ , where k is the converted result and m is the size of the table. Since the hashResult are double and the index of

the LinkedList are integer, I casted the hashResult and return the hashingResults with rounded integer.

After that, print out the hash table using for loop

```
System.out.println("Here are the top 50 keywords in the hash table");
for(int i = 0 ; i < hashTable.size(); i++){
    System.out.println(hashTable.get(i));
}
System.out.println("-----");
System.out.println("1. Insert a keyword");
System.out.println("2. Search a keyword");
System.out.println("3. Delete a keyword");
System.out.println("press any other number to leave");
```

OutPut:



```
HashingWithChaining x
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
Here are the top 50 keywords in the hash table
[google maps, craigslist, twitter]
[bank of america]
[instagram, zillow, wells fargo, indeed]
[cnn, google classroom]
[gmail, weather, linkedin, trump]
[]
[facebook, walmart, google translate, fox news, traductor]
[yahoo, home depot, news, calculator, paypal, msn, capital one]
[lowes, pinterest]
[espn, best buy, nba]
[amazon prime]
[usps tracking, google drive, roblox]
[youtube]
[google, ebay, you tube]
[]
[yahoo mail, google docs, translate, hotmail, maps, target, hulu]
[netflix, speed test, aol mail]
-----
1. Insert a keyword
2. Search a keyword
3. Delete a keyword
press any other number to leave
```

If user enter 1, it will allowed user to enter a keyword and insert in the hash table. In this case I will enter ssss. After I had entered the ssss, chainedHashInsert will be called as the previous part. And the keyword will be inserted to the hash table in slot 1. And also print out the hash table with the for loop as well.

```
if(selection == 1) {  
    System.out.println("Enter keyword: ");  
    String s = scanner.next();  
    chainedHashInsert(s);  
    System.out.println("Inserted");  
    for(int i = 0 ; i < hashTable.size(); i++){  
        System.out.println(hashTable.get(i));  
    }  
    System.out.println("-----");  
    System.out.println("1. Insert a keyword");  
    System.out.println("2. Search a keyword");  
    System.out.println("3. Delete a keyword");  
    System.out.println("press any other number to leave")  
}
```

Output:



```
3. Delete a keyword
press any other number to leave
1
Enter keyword:
ssss
Inserted
[google maps, craigslist, twitter]
[bank of america, ssss]
[instagram, zillow, wells fargo, indeed]
[cnn, google classroom]
[gmail, weather, linkedin, trump]
[]
[facebook, walmart, google translate, fox news, traductor]
[yahoo, home depot, news, calculator, paypal, msn, capital one]
[lowes, pinterest]
[espn, best buy, nba]
[amazon prime]
[usps tracking, google drive, roblox]
[youtube]
[google, ebay, you tube]
[]
[yahoo mail, google docs, translate, hotmail, maps, target, hulu]
[netflix, speed test, aol mail]
-----
1. Insert a keyword
2. Search a keyword
3. Delete a keyword
press any other number to leave
|
```

If the user selects 2, it allows the user to search an existing keyword.



```

if(selection == 2){
    System.out.println("Enter keyword: ");
    String s = scanner.next();
    String a = chainedHashSearch(s);
    System.out.println(a);
    System.out.println("-----");
    for(int i = 0 ; i < hashTable.size(); i++){
        System.out.println(hashTable.get(i));
    }
    System.out.println("-----");
    System.out.println("1. Insert a keyword");
    System.out.println("2. Search a keyword");
    System.out.println("3. Delete a keyword");
    System.out.println("press any other number to leave");
}

```

User will be allowed to enter the keyword. Then the chainedHashSearch method will be call.

```

public static String chainedHashSearch(String s) {
    int hashResult = chainedHashFunction(s);
    boolean isFound = false;
    for(int i = 0; i < hashTable.get(hashResult).size(); i++){
        if(hashTable.get(hashResult).get(i).equals(s)){
            isFound = true;
        }
    }
    if(isFound){
        return "Slot: " + hashResult;
    }
    else
    {
        return "Not Found";
    }
}
}

```

In the chainedHashSearch method, it get the hash result of the keyword and try to search the keyword is in slots the hash table or not. If the keyword is in the hash table, it will return the number of the slot where the keyword located, otherwise it will return “found”

In this case, I will search the keyword “aaaa”, which not in the table.

Output

```
-----  
1. Insert a keyword  
2. Search a keyword  
3. Delete a keyword  
press any other number to leave  
2  
Enter keyword:  
aaaa  
Not Found  
-----
```

Then I search a keyword that is in the table, “ssss”, the one I just entered and it will return slot 1

Output

```
-----  
1. Insert a keyword  
2. Search a keyword  
3. Delete a keyword  
press any other number to leave  
2  
Enter keyword:  
ssss  
Slot: 1
```

If the user select 3, it allow user to delete an existed keyword.

```

if(selection == 3){
    System.out.println("Enter keyword: ");
    String s = scanner.next();
    chainedHashDelete(s);
    System.out.println("Deleted");
    System.out.println("-----");
    for(int i = 0 ; i < hashTable.size(); i++){
        System.out.println(hashTable.get(i));
    }
    System.out.println("-----");
    System.out.println("1. Insert a keyword");
    System.out.println("2. Search a keyword");
    System.out.println("3. Delete a keyword");
    System.out.println("press any other number to leave");
}

```

User will be allowed to enter the keyword. Then the chainedHashDelete method will be call.

```

public static void chainedHashDelete(String s) {
    int haxResult = chainedHashFunction(s);
    for(int i = 0; i < hashTable.get(haxResult).size(); i++){
        if(hashTable.get(haxResult).get(i).equals(s)){
            hashTable.get(haxResult).remove(i);
        }
    }
}

```

In the chainedHashDelete method, it get the hash result of the keyword and try to look for the keyword is in slots the hash table and delete the keyword

In this case, I will delete "ssss"

Output

3

Enter keyword:

ssss

Deleted

-----

[google maps, craigslist, twitter]

[bank of america]

[instagram, zillow, wells fargo, indeed]

[cnn, google classroom]

[gmail, weather, linkedin, trump]

[]

[facebook, walmart, google translate, fox news, traductor]

[yahoo, home depot, news, calculator, paypal, msn, capital one]

[lowes, pinterest]

[espn, best buy, nba]

[amazon prime]

[usps tracking, google drive, roblox]

[youtube]

[google, ebay, you tube]

[]

[yahoo mail, google docs, translate, hotmail, maps, target, hulu]

[netflix, speed test, aol mail]

-----

Ssss had deleted from slot 1

3. Delete a keyword

press any other number to leave

6

Thanks for using! Bye!

When press other number, 6 in this case, it will end the application.

## Multiplication method

```

public static void main (String[] args) {

    new MultiplicationMethod();
    Scanner scanner = new Scanner(System.in);

    String[] keyWords = {"facebook", "youtube", "gmail", "google", "weather",
        "ebay", "yahoo", "walmart", "yahoo mail", "google translate", "google maps",
        "craigslist", "netflix", "google docs", "translate", "home depot", "news", "fox news",
        "calculator", "usps tracking", "cnn", "hotmail", "google drive", "maps", "paypal",
        "lowes", "instagram", "msn", "amazon prime", "target", "espn", "zillow", "bank of america",
        "wells fargo", "twitter", "google classroom", "indeed", "best buy", "speed test", "linkedin",
        "aol mail", "hulu", "you tube", "pinterest", "trump", "nba", "roblox", "capital one", "traductor"
    };

    //insert keywords to hash table
    for(String s : keyWords)
    {
        multiplicationInsert(s);
    }
}

```

In the beginning of the program, it generates a hash table (size 64) with 50 key words using the multiplicationInsert method.

```

/**
 * Insert keyword into hash table
 * @param s keyword string
 */
public static void multiplicationInsert(String s) {
    int haxResult = multiplicationHashFunction(s);
    hashTable.get(haxResult).add(s);
}

```

To insert the keyword, it gets the hashResult from multiplicationHashFunction and insert to the target slot.



```

public static int multiplicationHashFunction(String s) {
    int stringLength = s.length();
    double hashResult = 0;
    double temp = 0;
    //Transform the keyword into ascii code and add them up.
    while(stringLength != 0) {
        for (int i = 0; i < s.length(); i++) {
            temp= s.charAt(i);
            hashResult = hashResult + temp;
            stringLength--;
        }
    }

    //multiplication process in bit format.
    double k = hashResult;
    int m = 64;
    int p = 6;
    double w = 16;
    double A = (Math.sqrt(5.0)-1)/2 ;
    double S = A * Math.pow(2.0,w);
    int binaryResult = (int) Math.round( k * S);
    String binary = Integer.toBinaryString(binaryResult);
    //get R0
    int last16Bit = binary.length() - 16;
    String last16Bits = binary.substring(last16Bit, binary.length());
    // first six significant bits
    String significantBits = last16Bits.substring(0,p);
    // convert back to decimal
    int hashResults = Integer.parseInt(significantBits, radix: 2);
    return hashResults;
}

```

In the chainedHashFunction method I converted each letter in the keyword into ascii code with base 26 and added them up. Then, convert the this result to binary and substring the last 16 bits (r0). After that, substring the significant bits from r0 and convert back to decimal. That will be the hashResult.

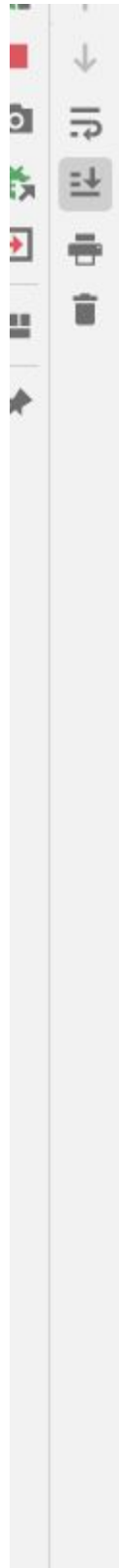
After that, print out the hash table using for loop

```
System.out.println("Here are the top 50 keywords in the hash table");  
for(int i = 0 ; i < hashTable.size(); i++){  
    System.out.println(hashTable.get(i));  
}  
System.out.println("-----");  
System.out.println("1. Insert a keyword");  
System.out.println("2. Search a keyword");  
System.out.println("3. Delete a keyword");  
System.out.println("press any other number to leave");
```

Output:



```
prites
MultiplicationMethod x RBTree x
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe" ...
Here are the top 50 keywords in the hash table
[]
[news, instagram, wells fargo]
[bank of america, aol mail, trump]
[]
[google maps]
[]
[]
[]
[google docs, roblox]
[cnn, traductor]
[]
[home depot]
[best buy]
[yahoo]
[]
[]
[]
[]
[]
[]
[indeed]
[]
[]
[]
[amazon prime]
[usps tracking, lowes, twitter]
[capital one]
[google translate, msn]
[]
```



```
[capital one]
[google translate, msn]
[]
[you tube]
[]
[facebook]
[nba]
[hotmail, google classroom]
[]
[]
[]
[]
[fox news, maps, speed test]
[gmail, craigslist]
[]
[hulu]
[]
[youtube]
[google, espn]
[walmart]
[ebay]
[]
[weather]
[]
[]
[]
[calculator]
[]
[linkedin, pinterest]
[paypal, target]
[]
```

```

[]
[]
[calculator]
[]
[linkedin, pinterest]
[paypal, target]
[]
[]
[]
[zillow]
[yahoo mail, netflix]
[translate, google drive]
[]
[]
-----
1. Insert a keyword
2. Search a keyword
3. Delete a keyword
press any other number to leave

```

If user enter 1, it will allowed user to enter a keyword and insert in the hash table. In this case I will enter hey. After I had entered the hey, multiplicationInsert will be called as the previous part. And the keyword will be inserted to the hash table in slot 1. And also print out the hash table with the for loop as well.

Output:

1. Insert a keyword
  2. Search a keyword
  3. Delete a keyword
- press any other number to leave

1

Enter keyword:

hey

Inserted

[]

[news, instagram, wells fargo]

[bank of america, aol mail, trump]

[]

[google maps]

[]

[]

[]

[google docs, roblox]

[cnn, traductor]

[]

[home depot]

[best buy]

[yahoo]

[]

[]

[]

[]

[]

[]

[indeed]

[]

---

```
{}  
[indeed]  
[]  
[]  
[]  
[amazon prime]  
[usps tracking, lowes, twitter]  
[capital one]  
[google translate, msn]  
[]  
[you tube]  
[hey]  
[facebook]  
[nba]  
[hotmail, google classroom]  
[]  
[]  
[]  
[]  
[fox news, maps, speed test]  
[gmail, craigslist]  
[]  
[hulu]  
[]  
[youtube]  
[google, espn]  
[walmart]  
[ebay]  
[]
```

```
[weather]
[ebay]
[]
[weather]
[]
[]
[]
[calculator]
[]
[linkedin, pinterest]
[paypal, target]
[]
[]
[]
[zillow]
[yahoo mail, netflix]
[translate, google drive]
[]
[]
```

If the user selects 2, it allows the user to search an existing keyword.

```

    }
    if(selection == 2){
        System.out.println("Enter keyword: ");
        String s = scanner.next();
        String a = multiplicationSearch(s);
        System.out.println(a);
        System.out.println("-----");
        for(int i = 0 ; i < hashTable.size(); i++){
            System.out.println(hashTable.get(i));
        }
        System.out.println("-----");
        System.out.println("1. Insert a keyword");
        System.out.println("2. Search a keyword");
        System.out.println("3. Delete a keyword");
        System.out.println("press any other number to leave");
    }
}

```

User will be allowed to enter the keyword. Then the multiplicationSearch method will be call.

```

    ^/
    public static String multiplicationSearch(String s) {
        int haxResult = multiplicationHashFunction(s);
        boolean isFound = false;
        for(int i = 0; i < hashTable.get(haxResult).size(); i++){
            if(hashTable.get(haxResult).get(i).equals(s)){
                isFound = true;
            }
        }
        if(isFound){
            return "Slot: " + haxResult;
        }
        else
        {
            return "Not Found";
        }
    }
}

```



In the multiplicationSearch method, it get the hash result of the keyword and try to search the keyword is in slots the hash table or not. If the keyword is in the hash table, it will return the number of the slot where the keyword located, otherwise it will return "found"

In this case, I will search the keyword "youtube", which in the table and it will return slot 43

Output:

```
1. Insert a keyword
2. Search a keyword
3. Delete a keyword
press any other number to leave
2
Enter keyword:
youtube
Slot: 43
-----
[]
```

If the user select 3, it allow user to delete an existed keyword.

```
if(selection == 3){
    System.out.println("Enter keyword: ");
    String s = scanner.next();
    multiplicationDelete(s);
    System.out.println("Deleted");
    System.out.println("-----");
    for(int i = 0 ; i < hashTable.size(); i++){
        System.out.println(hashTable.get(i));
    }
    System.out.println("-----");
    System.out.println(hashTable);
    System.out.println("1. Insert a keyword");
    System.out.println("2. Search a keyword");
    System.out.println("3. Delete a keyword");
    System.out.println("press any other number to leave");
}
```

User will be allowed to enter the keyword. Then the multiplicationDelete method will be call.

```
/**
 * Delete an existing keyword
 * @param s keyword to delete
 */
public static void multiplicationDelete(String s) {
    int haxResult = multiplicationHashFunction(s);
    for(int i = 0; i < hashTable.get(haxResult).size(); i++){
        if(hashTable.get(haxResult).get(i).equals(s)){
            hashTable.get(haxResult).remove(i);
        }
    }
}
```

In the multiplicationDelete method, it get the hash result of the keyword and try to look for the keyword is in slots the hash table and delete the keyword

In this case, I will delete “news” which in slot 1

Output:

```
3. Delete a keyword
press any other number to leave
3
Enter keyword:
news
Deleted
-----
[]
[instagram, wells fargo]
[bank of america, aol mail, trump]
[]
[google maps]
[]
[]
[]
[google docs, roblox]
[cnn, traductor]
[]
[home depot]
[best buy]
[yahoo]
[]
```

As we can see that news had removed from the slot 1 and only instagram and wells fargo left.

```
3. Delete a keyword
press any other number to leave
6
Thanks for using! Bye!
```

When press other number, 6 in this case, it will end the application.

### Double Hashing method

```

public static void main (String args[]) {
    Scanner scanner = new Scanner(System.in);
    new DoubleHashing();

    String[] keyWords = {"facebook", "youtube", "gmail", "google", "weather",
        "ebay", "yahoo", "walmart", "yahoo mail", "google translate", "google maps",
        "craigslist", "netflix", "google docs", "translate", "home depot", "news", "fox news",
        "calculator", "usps tracking", "cnn", "hotmail", "google drive", "maps", "paypal",
        "lowes", "instagram", "msn", "amazon prime", "target", "espn", "zillow", "bank of america",
        "wells fargo", "twitter", "google classroom", "indeed", "best buy", "speed test", "linkedin",
        "aol mail", "hulu", "you tube", "pinterest", "trump", "nba", "roblox", "capital one", "traductor"
    };
    //insert keywords to hash table
    for(String s : keyWords)
    {
        doubleHashingInsert(s);
    }
    System.out.println("Here are the top 50 keywords in the hash table");
    for(int i = 0 ; i < hashTable.size(); i++){
        System.out.println(hashTable.get(i));
    }
    System.out.println("1. Insert a keyword");
    System.out.println("2. Search a keyword");
    System.out.println("press any other number to leave");
}

```

In the beginning of the program, it generates a hash table (size 57) with 50 key words using the doubleHashingInsert method.

doubleHashingInsert method will base on the two functions, firstHashingFunction and secondHashingFunction.

```
public static int firstHashingFunction(String s){
    int stringLength = s.length();
    double hashResult = 0;
    double temp = 0;
    //Transform the keyword into ascii code, base 26 and add them up.
    while(stringLength != 0) {
        for (int i = 0; i < s.length(); i++) {
            temp= s.charAt(i);
            temp = temp * Math.pow(26, i);
            hashResult = hashResult + temp;
            stringLength--;
        }
    }
    // first hashing function K mod 57
    hashResult = hashResult % 57;
    int hashResults = (int) Math.round(hashResult);
    return hashResults;
}
```

```

public static int secondHashingFunction(String s){
    int stringLength = s.length();
    double hashResult = 0;
    double temp = 0;|
    //Transform the keyword into ascii code, base 26
    while(stringLength != 0) {
        for (int i = 0; i < s.length(); i++) {
            temp= s.charAt(i);
            temp = temp * Math.pow(26, i);
            hashResult = hashResult + temp;
            stringLength--;
        }
    }

    // second hashing function 1 + (k mod 13)
    hashResult = 1 + (hashResult % 13);
    int hashResults = (int) Math.round(hashResult);
    return hashResults;
}

```

Both of the hash functions works the same way but using different equation to get the result. The first one will be  $h(k) = k \bmod 57$ , the second one will be  $h(k) = 1 + (k \bmod 13)$

```

public static void doubleHashingInsert(String s) {
    //first hashing function result
    int firstHashResult = firstHashingFunction(s);
    int secondHashResult;
    int hashResult = firstHashResult;
    int probe = 0;
    if (hashTable.get(firstHashResult).equals("null")) {
        hashTable.set(firstHashResult, s);
    } else {
        while (!hashTable.get(hashResult).equals("null")) {
            probe++;
            //second hashing function result
            secondHashResult = secondHashingFunction(s);

            //double hashing  $h(k,i) = ((h1(k) + i h2(k)) \bmod m$ 
            hashResult = (firstHashResult + (probe * secondHashResult)) % 57;
        }
        hashTable.set(hashResult, s);
    }
}
}

```

In the double hashingInsert method, It insert to the slot if the slot is empty. When, collision happens, it used  $h(k,i) = ((h1(k) + i h2(k)) \bmod m$ , where h1 is the first hashing function and h2 is the second hashing function and i will be the probe, to get the hash result. And insert into the keyword into LinkedList.

Output:





```
"C:\Program Files\Java\jdk1.8.0_261\bin\java.exe"
```

```
Here are the top 50 keywords in the hash table
```

```
null
```

```
gmail
```

```
traductor
```

```
null
```

```
google classroom
```

```
youtube
```

```
espn
```

```
indeed
```

```
netflix
```

```
craigslist
```

```
hulu
```

```
google docs
```

```
target
```

```
null
```

```
twitter
```

```
you tube
```

```
paypal
```

```
trump
```

```
ebay
```

```
roblox
```

```
yahoo
```

```
walmart
```

```
lowes
```

```
null
```

```
translate
```

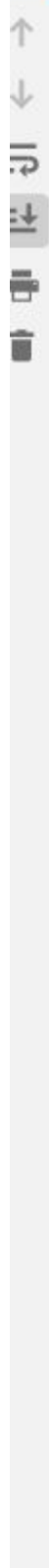
```
amazon prime
```

```
instagram
```

```
cnn
```

```
hotmail
```

```
weather
```



```
weather
capital one
null
fox news
zillow
bank of america
linkedin
pinterest
news
google translate
null
speed test
best buy
google drive
google
null
facebook
home depot
google maps
yahoo mail
aol mail
nba
maps
null
msn
usps tracking
calculator
wells fargo
1. Insert a keyword
2. Search a keyword
press any other number to leave
```

If user enter 1, it will allowed user to enter a keyword and insert in the hash table. In this case I will enter code. After I had entered the code, doubleHashingInsert will be called as the previous part. And the keyword will be inserted to the hash table in slot 1. And also print out the hash table with the for loop as well.

```
if(selection == 1) {  
    System.out.println("Enter keyword: ");  
    String s = scanner.next();  
    doubleHashingInsert(s);  
    System.out.println("Inserted");  
    for(int i = 0 ; i < hashTable.size(); i++){  
        System.out.println(hashTable.get(i));  
    }  
    System.out.println("-----");  
    System.out.println("1. Insert a keyword");  
    System.out.println("2. Search a keyword");  
    System.out.println("press any other number to leave");  
}
```

Output:

1. Insert a keyword
  2. Search a keyword
- press any other number to leave

1

Enter keyword:

code

Inserted

null

gmail

traductor

null

google classroom

youtube

espn

indeed

netflix

craigslist

hulu

google docs

target

null

twitter

you tube

paypal

trump

ebay

roblox

yahoo

walmart

lowes

walmart  
lowes  
null  
translate  
amazon prime  
instagram  
cnn  
hotmail  
weather  
capital one  
null  
fox news  
zillow  
bank of america  
linkedin  
pinterest  
news  
google translate  
null  
speed test  
best buy  
google drive  
google  
code   
facebook  
home depot  
google maps  
yahoo mail  
aol mail  
nba  
maps  
null

```
yahoo mail
aol mail
nba
maps
null
msn
usps tracking
calculator
wells fargo
-----
1. Insert a keyword
2. Search a keyword
press any other number to leave
```

As we can see, the keyword “code” had inserted into the empty slot after the word google.

If the user selects 2, it allows the user to search an existing keyword.

```
}
if(selection == 2){
    System.out.println("Enter keyword: ");
    String s = scanner.next();
    String a = doubleHashingSearch(s);
    System.out.println(a);
    System.out.println("-----");
    for(int i = 0 ; i < hashTable.size(); i++){
        System.out.println(hashTable.get(i));
    }
    System.out.println("-----");
    System.out.println("1. Insert a keyword");
    System.out.println("2. Search a keyword");
    System.out.println("press any other number to leave");
}
selection = scanner.nextInt();
```

User will be allowed to enter the keyword. Then the doubleHashingSearch method will be call.

```

    π/
    public static String doubleHashingSearch(String s){
        int slot = 0;
        boolean isFound = false;
        for(int i = 0; i < hashTable.size(); i++){
            if(hashTable.get(i).equals(s)){
                slot = i ;
                isFound = true;
            }
        }
        if (isFound){
            return "Slot: " + slot;
        }
        else{
            return "Not Found";
        }
    }
}

```

In the doubleHashingSearch method, it get the hash result of the keyword and try to search the keyword is in slots the hash table or not. If the keyword is in the hash table, it will return the number of the slot where the keyword located, otherwise it will return “found”

In this case, I will search the keyword “nba”, which in the table and it will return 50



```
nba
maps
null
msn
usps tracking
calculator
wells fargo
-----
1. Insert a keyword
2. Search a keyword
press any other number to leave
2
Enter keyword:
nba
Slot: 50
```

When press other numbers, 6 in this case, it will end the application.

```
wells fargo
-----
1. Insert a keyword
2. Search a keyword
press any other number to leave
6
Thanks for using! Bye!
```

## Exercise 2 (Google Search Engine PageRank Priority Queue)

```
public static void main(String args[]){
    RBTree t = new RBTree();
    Scanner scanner = new Scanner(System.in);
    System.out.println("Welcome to Google Search Engine Simulator");
    System.out.println("-----");
    System.out.println("Please enter the keyword.");
    String keyword = scanner.nextLine();
    WebCrawler crawler = new WebCrawler (keyword);
    crawler.search();
    Set<Integer> s = new HashSet<>();

    //generate scores and store in a HashSet
    while(s.size() < 20){
        Random randomScore = new Random();
        int pageRank = randomScore.nextInt( bound: 100) + 1;
        s.add(pageRank);
    }
    //save the score in the HashSet into ArrayList
    ArrayList<Integer> scores = new ArrayList<>();
    for (Integer a : s) {
        scores.add(a);
    }
}
```

In the beginning of the program, it constructed an empty Red Black Tree. Then, allow user to search the keyword using the web crawler. After that it will generate 20 random rankScore and store it in a hashset, so there will be no duplicates. After that it stores the scores in an ArrayList. The website being crawled will store in an ArrayList too. Then, it will print out the urls and page rank score. In this case, I will search sjsu

```
//print out clawing result
for(int i = 0 ; i < 20; i++){
    System.out.println(urls.get(i) + " PageRank:" + scores.get(i));
}
System.out.println("-----");
```

Output:

**\*\*Visiting\*\*** Received web page at <https://google.com/search?q=sjsu>  
Found (145) links

Searching for the word sjsu...

**\*\*Success\*\*** Word sjsu found at <https://google.com/search?q=sjsu&nl>

**\*\*Done\*\*** Visited 90 web page(s)

Here are the first 20 URL links with score:

[www.sjsu.edu/&sa=U&ved=2ahU](http://www.sjsu.edu/&sa=U&ved=2ahU) PageRank:34  
[www.myikc.com/sjsu-grad/&sa=](http://www.myikc.com/sjsu-grad/&sa=) PageRank:99  
[www.wscuc.org/institutions/](http://www.wscuc.org/institutions/) PageRank:36  
[www.gradreports.com/college](http://www.gradreports.com/college) PageRank:38  
[sjsujmc.com/&sa=U&ved=2ahUK](http://sjsujmc.com/&sa=U&ved=2ahUK) PageRank:39  
[twitter.com/SJSU%3Fref\\_src%](https://twitter.com/SJSU%3Fref_src%3D) PageRank:41  
[www.mathworks.com/academia/](http://www.mathworks.com/academia/) PageRank:11  
[www.applitrack.com/sjsu/onl](http://www.applitrack.com/sjsu/onl) PageRank:44  
[www.cbssports.com/college-f](http://www.cbssports.com/college-f) PageRank:17  
[www.niche.com/colleges/san-](http://www.niche.com/colleges/san-) PageRank:49  
[ideassjsu.org/&sa=U&ved=2ah](http://ideassjsu.org/&sa=U&ved=2ah) PageRank:53  
[sjsunews.com/&sa=U&ved=2ahU](http://sjsunews.com/&sa=U&ved=2ahU) PageRank:23  
[www.calstateonline.net/Cal-](http://www.calstateonline.net/Cal-) PageRank:88  
[en.wikipedia.org/wiki/Mary\\_](https://en.wikipedia.org/wiki/Mary_) PageRank:24  
[www.youtube.com/user/sjsu&s](https://www.youtube.com/user/sjsu&s) PageRank:56  
[www.ed2go.com/sjsu/&sa=U&ve](http://www.ed2go.com/sjsu/&sa=U&ve) PageRank:58  
[en.wikipedia.org/wiki/Weste](https://en.wikipedia.org/wiki/Weste) PageRank:26  
[www.mercurynews.com/2020/11](http://www.mercurynews.com/2020/11) PageRank:27  
[www.mercurynews.com/2020/12](http://www.mercurynews.com/2020/12) PageRank:30  
[www.sjsu.edu/academics/&sa=](http://www.sjsu.edu/academics/&sa=) PageRank:62



Then, it construct an RedBlack tree using RBInsert(),

```
//save the clawing result into an ArrayList
ArrayList<String> urls = new ArrayList<>();
for (String w : crawler.getUrls()){
    urls.add(w);
}
//print out clawing result
for(int i = 0 ; i < 20; i++){
    System.out.println(urls.get(i) + " PageRank:" + scores.get(i));
}
System.out.println("-----");
ArrayList<RBNode> nodes = new ArrayList<>();
//creat node with urls and pageRank score and create a Red Black Tree
for(int i = 0 ; i < 20; i++){
    RBNode node = new RBNode(urls.get(i),scores.get(i));
    t.RBInsert(t,node);
}
```

Then, print out the RedBlack Tree

```
t
System.out.println("Here is the Red Black Tree in Inorder walk");
t.printRBTree(t.root);
System.out.println("-----");
System.out.println("Press 1 to insert a node");
System.out.println("Press 2 to delete a node");
System.out.println("Press 3 to sort the websites");
System.out.println("-----");
```

Output:

```
-----
Here is the Red Black Tree in Inorder walk
Current: www.mathworks.com/academia/ 11 BLACK | Parent: www.cbssports.com/college-f 17 BLACK | Left Child: null 0 BLACK | Right Child: null 0 BLACK
Current: www.cbssports.com/college-f 17 BLACK | Parent: en.wikipedia.org/wiki/Mary_ 24 RED | Left Child: www.mathworks.com/academia/ 11 BLACK | Right Child: sjsunews.com/&sa=U&ved=2ahU 23 BLACK | Parent: www.cbssports.com/college-f 17 BLACK | Left Child: null 0 BLACK | Right Child: null 0 BLACK
Current: sjsunews.com/&sa=U&ved=2ahU 23 BLACK | Parent: www.cbssports.com/college-f 17 BLACK | Left Child: null 0 BLACK | Right Child: null 0 BLACK
Current: en.wikipedia.org/wiki/Mary_ 24 RED | Parent: sjsujmc.com/&sa=U&ved=2ahU 39 BLACK | Left Child: www.cbssports.com/college-f 17 BLACK | Right Child: www.wscuc.org/institutions/ 36 BLACK | Parent: en.wikipedia.org/wiki/Weste 26 BLACK | Parent: www.mercurynews.com/2020/11 27 RED | Left Child: null 0 BLACK | Right Child: null 0 BLACK
Current: www.mercurynews.com/2020/11 27 RED | Parent: www.wscuc.org/institutions/ 36 BLACK | Left Child: en.wikipedia.org/wiki/Weste 26 BLACK | Right Child: www.sjsu.edu/&sa=U&ved=2ahU 30 RED | Parent: www.sjsu.edu/&sa=U&ved=2ahU 34 BLACK | Left Child: null 0 BLACK | Right Child: null 0 BLACK
Current: www.sjsu.edu/&sa=U&ved=2ahU 34 BLACK | Parent: www.mercurynews.com/2020/11 27 RED | Left Child: www.mercurynews.com/2020/12 30 RED | Right Child: null 0 BLACK
Current: www.wscuc.org/institutions/ 36 BLACK | Parent: en.wikipedia.org/wiki/Mary_ 24 RED | Left Child: www.mercurynews.com/2020/11 27 RED | Right Child: www.gradreports.com/college 38 BLACK | Parent: www.wscuc.org/institutions/ 36 BLACK | Left Child: null 0 BLACK | Right Child: null 0 BLACK
Current: www.gradreports.com/college 38 BLACK | Parent: www.wscuc.org/institutions/ 36 BLACK | Left Child: null 0 BLACK | Right Child: null 0 BLACK
Current: sjsujmc.com/&sa=U&ved=2ahU 39 BLACK | Parent: null 0 BLACK | Left Child: en.wikipedia.org/wiki/Mary_ 24 RED | Right Child: ideassjsu.org/&sa=U&ved=2ah 53 BLACK | Parent: www.applitrack.com/sjsu/onl 44 RED | Left Child: null 0 BLACK | Right Child: null 0 BLACK
Current: twitter.com/SJSU%3Fref_src% 41 BLACK | Parent: www.applitrack.com/sjsu/onl 44 RED | Left Child: null 0 BLACK | Right Child: null 0 BLACK
Current: www.applitrack.com/sjsu/onl 44 RED | Parent: ideassjsu.org/&sa=U&ved=2ah 53 BLACK | Left Child: twitter.com/SJSU%3Fref_src% 41 BLACK | Right Child: www.niche.com/colleges/san- 49 BLACK | Parent: www.applitrack.com/sjsu/onl 44 RED | Left Child: null 0 BLACK | Right Child: null 0 BLACK
Current: www.niche.com/colleges/san- 49 BLACK | Parent: www.applitrack.com/sjsu/onl 44 RED | Left Child: null 0 BLACK | Right Child: null 0 BLACK
Current: ideassjsu.org/&sa=U&ved=2ah 53 BLACK | Parent: sjsujmc.com/&sa=U&ved=2ahU 39 BLACK | Left Child: www.applitrack.com/sjsu/onl 44 RED | Right Child: www.calstateonline.net/Cal- 88 RED | Parent: www.ed2go.com/sjsu/&sa=U&ved=2ah 58 BLACK | Left Child: null 0 BLACK | Right Child: null 0 BLACK
Current: www.youtube.com/user/sjsu&s 56 RED | Parent: www.ed2go.com/sjsu/&sa=U&ved=2ah 58 BLACK | Left Child: null 0 BLACK | Right Child: null 0 BLACK
Current: www.ed2go.com/sjsu/&sa=U&ved=2ah 58 BLACK | Parent: www.calstateonline.net/Cal- 88 RED | Left Child: www.youtube.com/user/sjsu&s 56 RED | Right Child: www.sjsu.edu/academics/ 62 RED | Parent: www.ed2go.com/sjsu/&sa=U&ved=2ah 58 BLACK | Left Child: null 0 BLACK | Right Child: null 0 BLACK
Current: www.sjsu.edu/academics/ 62 RED | Parent: www.ed2go.com/sjsu/&sa=U&ved=2ah 58 BLACK | Left Child: null 0 BLACK | Right Child: null 0 BLACK
Current: www.calstateonline.net/Cal- 88 RED | Parent: ideassjsu.org/&sa=U&ved=2ah 53 BLACK | Left Child: www.ed2go.com/sjsu/&sa=U&ved=2ah 58 BLACK | Right Child: www.mykic.com/sjsu-gr 99 BLACK | Parent: www.calstateonline.net/Cal- 88 RED | Left Child: null 0 BLACK | Right Child: null 0 BLACK
Current: www.mykic.com/sjsu-gr 99 BLACK | Parent: www.calstateonline.net/Cal- 88 RED | Left Child: null 0 BLACK | Right Child: null 0 BLACK
-----
Press 1 to insert a node
```

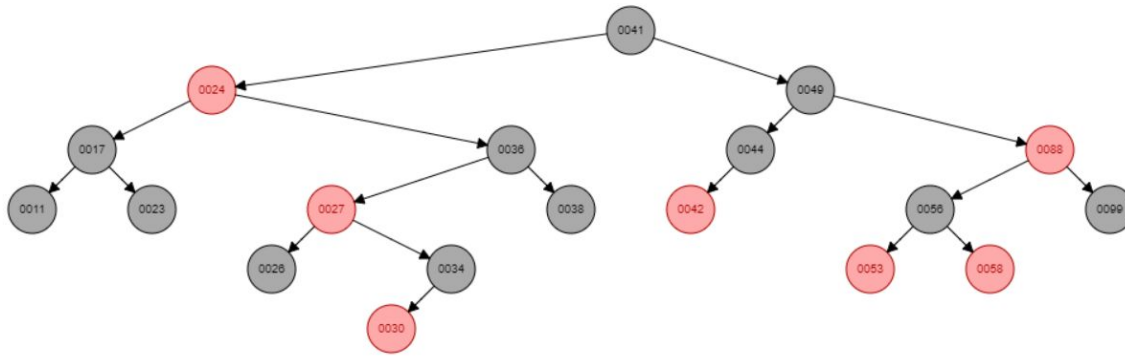
Here is a better view

```

-----
Here is the Red Black Tree in Inorder walk
Current: www.mathworks.com/academia/ 11 BLACK |
Current: www.cbssports.com/college-f 17 BLACK |
Current: sjsunews.com/&sa=U&ved=2ahU 23 BLACK |
Current: en.wikipedia.org/wiki/Mary\_ 24 RED | Pa
Current: en.wikipedia.org/wiki/Weste 26 BLACK |
Current: www.mercurynews.com/2020/11 27 RED | Pa
Current: www.mercurynews.com/2020/12 30 RED | Pa
Current: www.sjsu.edu/&sa=U&ved=2ahU 34 BLACK |
Current: www.wscuc.org/institutions/ 36 BLACK |
Current: www.gradreports.com/college 38 BLACK |
Current: sjsujmc.com/&sa=U&ved=2ahUK 39 BLACK |
Current: twitter.com/SJSU%3Fref\_src% 41 BLACK |
Current: www.applitrack.com/sjsu/onl 44 RED | Pa
Current: www.niche.com/colleges/san- 49 BLACK |
Current: ideassjsu.org/&sa=U&ved=2ah 53 BLACK |
Current: www.youtube.com/user/sjsu&s 56 RED | Pa
Current: www.ed2go.com/sjsu/&sa=U&ve 58 BLACK |
Current: www.sjsu.edu/academics/&sa= 62 RED | Pa
Current: www.calstateonline.net/Cal- 88 RED | Pa
Current: www.myikc.com/sjsu-grad/&sa= 99 BLACK |
-----

```

It shows the parent, left, right children of each node.



If user press 1, it allow user to enter the url and rank score to insert a node. It will call the RBInsert method as well as RBInsertFixup method

```
// allow user to insert node
if (selection == 1){
    System.out.println("Enter url");
    String url = scanner.next();
    System.out.println("Enter PageRank Score");
    int pageRank = scanner.nextInt();
    RBNode insert = new RBNode(url, pageRank);
    t.RBInsert(t, insert);
    System.out.println("Inserted");
    System.out.println("-----");
    System.out.println("Here is the Red Black Tree in Inorder walk");
    t.printRBTree(t.root);
    System.out.println("-----");
}
```

In this case, I will insert [www.sjsu.edu](http://www.sjsu.edu) with a page rank 100



1

Enter url

[www.sjsu.edu](http://www.sjsu.edu)

Enter PageRank Score

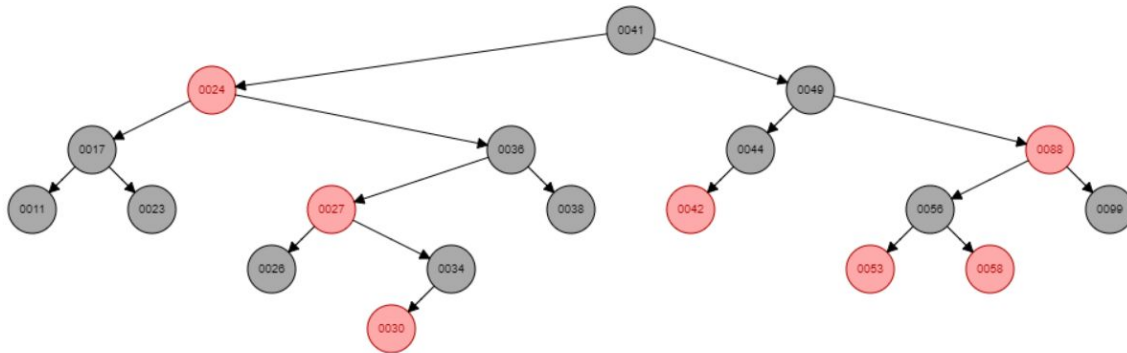
100

Inserted

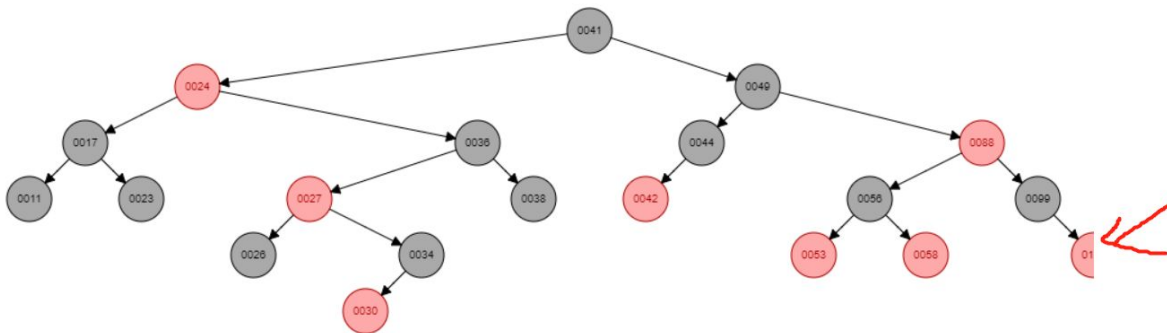
-----  
Here is the Red Black Tree in Inorder walk

Current: [www.mathworks.com/academia/](http://www.mathworks.com/academia/) 11 BLACK |  
Current: [www.cbssports.com/college-f](http://www.cbssports.com/college-f) 17 BLACK |  
Current: [sjsunews.com/&sa=U&ved=2ahU](http://sjsunews.com/&sa=U&ved=2ahU) 23 BLACK |  
Current: [en.wikipedia.org/wiki/Mary\\_](http://en.wikipedia.org/wiki/Mary_) 24 RED | P  
Current: [en.wikipedia.org/wiki/Weste](http://en.wikipedia.org/wiki/Weste) 26 BLACK |  
Current: [www.mercurynews.com/2020/11](http://www.mercurynews.com/2020/11) 27 RED | P  
Current: [www.mercurynews.com/2020/12](http://www.mercurynews.com/2020/12) 30 RED | P  
Current: [www.sjsu.edu/&sa=U&ved=2ahU](http://www.sjsu.edu/&sa=U&ved=2ahU) 34 BLACK |  
Current: [www.wscuc.org/institutions/](http://www.wscuc.org/institutions/) 36 BLACK |  
Current: [www.gradreports.com/college](http://www.gradreports.com/college) 38 BLACK |  
Current: [sjsujmc.com/&sa=U&ved=2ahUK](http://sjsujmc.com/&sa=U&ved=2ahUK) 39 BLACK |  
Current: [twitter.com/SJSU%3Fref\\_src%](https://twitter.com/SJSU%3Fref_src%) 41 BLACK |  
Current: [www.applitrack.com/sjsu/onl](http://www.applitrack.com/sjsu/onl) 44 RED | P  
Current: [www.niche.com/colleges/san-](http://www.niche.com/colleges/san-) 49 BLACK |  
Current: [ideassjsu.org/&sa=U&ved=2ah](http://ideassjsu.org/&sa=U&ved=2ah) 53 BLACK |  
Current: [www.youtube.com/user/sjsu&s](http://www.youtube.com/user/sjsu&s) 56 RED | P  
Current: [www.ed2go.com/sjsu/&sa=U&ve](http://www.ed2go.com/sjsu/&sa=U&ve) 58 BLACK |  
Current: [www.sjsu.edu/academics/&sa=](http://www.sjsu.edu/academics/&sa=) 62 RED | P  
Current: [www.calstateonline.net/Cal-](http://www.calstateonline.net/Cal-) 88 RED | P  
Current: [www.myikc.com/sjsu-grad/&sa=](http://www.myikc.com/sjsu-grad/&sa=) 99 BLACK  
Current: [www.sjsu.edu](http://www.sjsu.edu) 100 RED | Parent: [www.myi](http://www.myi)

This will be the RB tree before insertion



After inserted



After that, Users are also allowed to press 2 to delete a node.

```
// allowed user to delete node
if(selection == 2){
    System.out.println("Enter PageRank Score");
    int pageRank = scanner.nextInt();
    RBNode delete = t.iterativeTreeSearch(t.root,pageRank);
    t.RBDelete(t,delete);
    System.out.println("Deleted");
    System.out.println("-----");
    System.out.println("Here is the Red Black Tree in Inorder walk");
    t.printRBTree(t.root);
    System.out.println("-----");
}
```

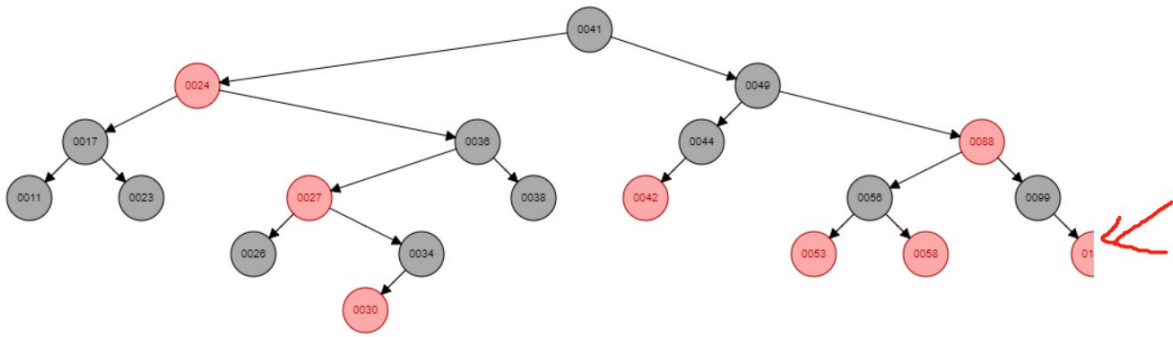
In this case I will delete node 99

```
-----
2
Enter PageRank Score
99
Deleted
-----

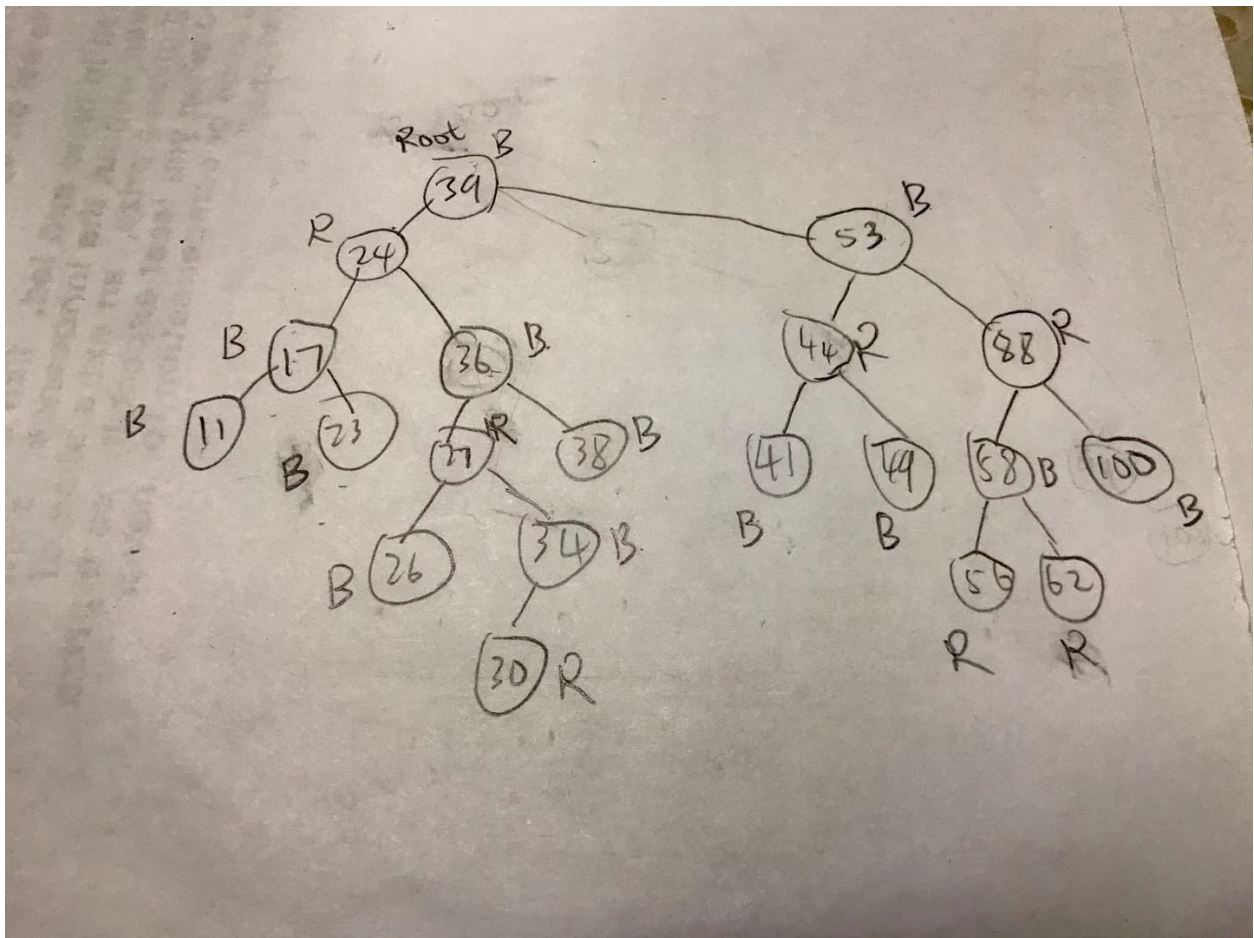
Here is the Red Black Tree in Inorder walk
Current: www.mathworks.com/academia/ 11 BLACK | Parent: ww
Current: www.cbssports.com/college-f 17 BLACK | Parent: er
Current: sjsunews.com/&sa=U&ved=2ahU 23 BLACK | Parent: ww
Current: en.wikipedia.org/wiki/Mary\_ 24 RED | Parent: sjsu
Current: en.wikipedia.org/wiki/Weste 26 BLACK | Parent: ww
Current: www.mercurynews.com/2020/11 27 RED | Parent: www.
Current: www.mercurynews.com/2020/12 30 RED | Parent: www.
Current: www.sjsu.edu/&sa=U&ved=2ahU 34 BLACK | Parent: ww
Current: www.wscuc.org/institutions/ 36 BLACK | Parent: er
Current: www.gradreports.com/college 38 BLACK | Parent: ww
Current: sjsujmc.com/&sa=U&ved=2ahUK 39 BLACK | Parent: nu
Current: twitter.com/SJSU%3Fref\_src% 41 BLACK | Parent: ww
Current: www.applitrack.com/sjsu/onl 44 RED | Parent: idea
Current: www.niche.com/colleges/san- 49 BLACK | Parent: ww
Current: ideassjsu.org/&sa=U&ved=2ah 53 BLACK | Parent: s
Current: www.youtube.com/user/sjsu&s 56 RED | Parent: www.
Current: www.ed2go.com/sjsu/&sa=U&ve 58 BLACK | Parent: ww
Current: www.sjsu.edu/academics/&sa= 62 RED | Parent: www.
Current: www.calstateonline.net/Cal- 88 RED | Parent: idea
Current: www.sjsu.edu 100 BLACK | Parent: www.calstateonli
-----
|
```

Before deleting.





After deleting



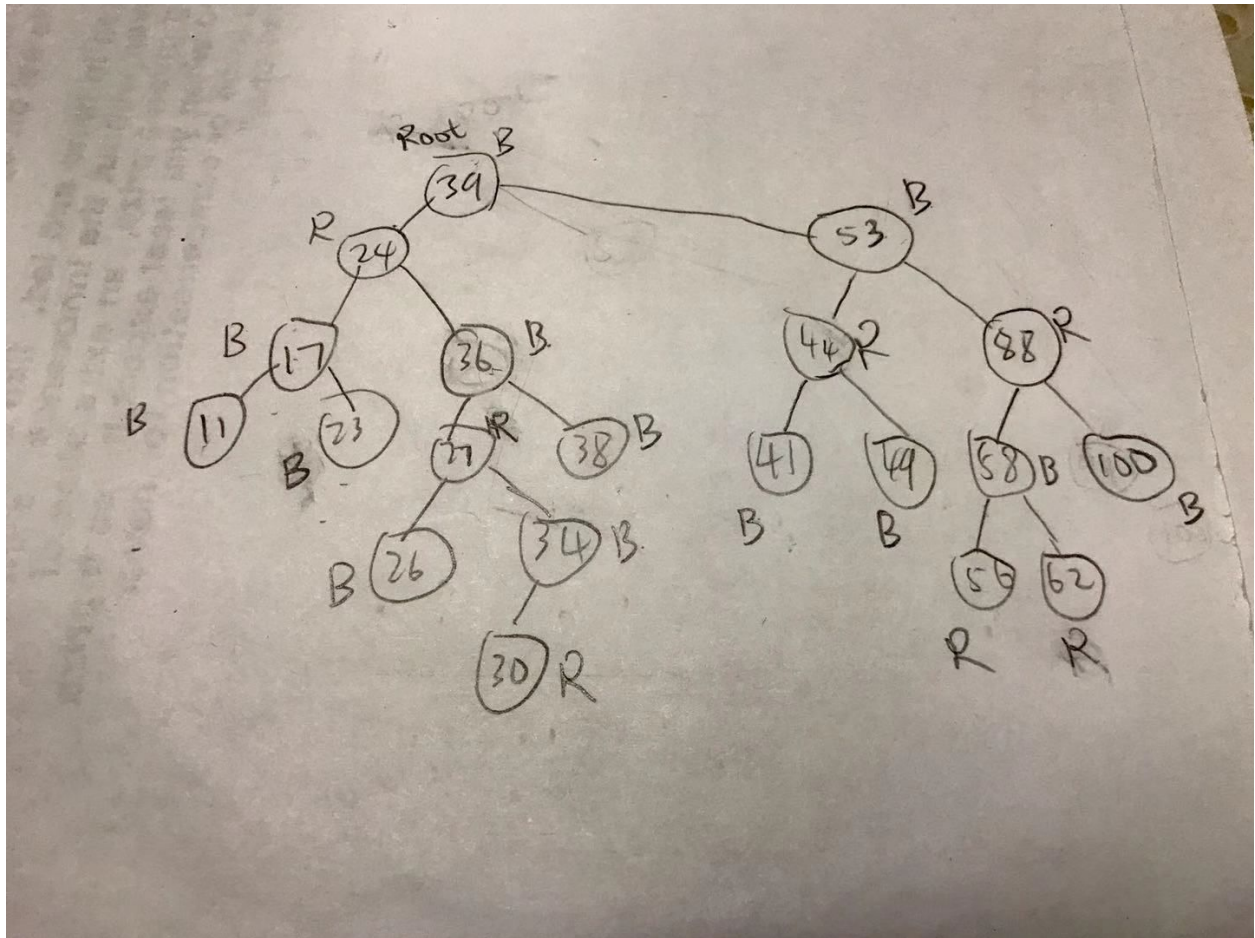
```
// perform sorting
if(selection == 3){
    System.out.println("Here is the sorted Red Black Tree ");
    t.RBSort(t.root);
    System.out.println("-----");
}
```

When user press 3, it allows the user to see the sorted url base on the Rankscore, it will call RBSort method (reverse Inorder walk).

Before sorting

Here is the Red Black Tree in Inorder walk

Current: [www.mathworks.com/academia/](http://www.mathworks.com/academia/) 11 BLACK  
 Current: [www.cbssports.com/college-f](http://www.cbssports.com/college-f) 17 BLACK  
 Current: [sjsunews.com/&sa=U&ved=2ahU](http://sjsunews.com/&sa=U&ved=2ahU) 23 BLACK  
 Current: [en.wikipedia.org/wiki/Mary\\_](http://en.wikipedia.org/wiki/Mary_) 24 RED |  
 Current: [en.wikipedia.org/wiki/Weste](http://en.wikipedia.org/wiki/Weste) 26 BLACK  
 Current: [www.mercurynews.com/2020/11](http://www.mercurynews.com/2020/11) 27 RED |  
 Current: [www.mercurynews.com/2020/12](http://www.mercurynews.com/2020/12) 30 RED |  
 Current: [www.sjsu.edu/&sa=U&ved=2ahU](http://www.sjsu.edu/&sa=U&ved=2ahU) 34 BLACK  
 Current: [www.wscuc.org/institutions/](http://www.wscuc.org/institutions/) 36 BLACK  
 Current: [www.gradreports.com/college](http://www.gradreports.com/college) 38 BLACK  
 Current: [sjsujmc.com/&sa=U&ved=2ahUK](http://sjsujmc.com/&sa=U&ved=2ahUK) 39 BLACK  
 Current: [twitter.com/SJSU%3Fref\\_src%](https://twitter.com/SJSU%3Fref_src%) 41 BLACK  
 Current: [www.applitrack.com/sjsu/onl](http://www.applitrack.com/sjsu/onl) 44 RED |  
 Current: [www.niche.com/colleges/san-](http://www.niche.com/colleges/san-) 49 BLACK  
 Current: [ideassjsu.org/&sa=U&ved=2ah](http://ideassjsu.org/&sa=U&ved=2ah) 53 BLACK  
 Current: [www.youtube.com/user/sjsu&s](http://www.youtube.com/user/sjsu&s) 56 RED |  
 Current: [www.ed2go.com/sjsu/&sa=U&ve](http://www.ed2go.com/sjsu/&sa=U&ve) 58 BLACK  
 Current: [www.sjsu.edu/academics/&sa=](http://www.sjsu.edu/academics/&sa=) 62 RED |  
 Current: [www.calstateonline.net/Cal-](http://www.calstateonline.net/Cal-) 88 RED |  
 Current: [www.sjsu.edu](http://www.sjsu.edu) 100 BLACK | Parent: [www.sjsu.edu](http://www.sjsu.edu)



After sorting



-----  
3

Here is the sorted Red Black Tree

<a href="http://www.sjsu.edu">www.sjsu.edu</a>	PageRank:100	BLACK
<a href="http://www.calstateonline.net/Cal-">www.calstateonline.net/Cal-</a>	PageRank:88	RED
<a href="http://www.sjsu.edu/academics/&amp;sa=">www.sjsu.edu/academics/&amp;sa=</a>	PageRank:62	RED
<a href="http://www.ed2go.com/sjsu/&amp;sa=U&amp;ve">www.ed2go.com/sjsu/&amp;sa=U&amp;ve</a>	PageRank:58	BLACK
<a href="http://www.youtube.com/user/sjsu&amp;s">www.youtube.com/user/sjsu&amp;s</a>	PageRank:56	RED
<a href="http://ideassjsu.org/&amp;sa=U&amp;ved=2ah">ideassjsu.org/&amp;sa=U&amp;ved=2ah</a>	PageRank:53	BLACK
<a href="http://www.niche.com/colleges/san-">www.niche.com/colleges/san-</a>	PageRank:49	BLACK
<a href="http://www.applitrack.com/sjsu/onl">www.applitrack.com/sjsu/onl</a>	PageRank:44	RED
<a href="https://twitter.com/SJSU%3Fref_src%">twitter.com/SJSU%3Fref_src%</a>	PageRank:41	BLACK
<a href="http://sjsujmc.com/&amp;sa=U&amp;ved=2ahUK">sjsujmc.com/&amp;sa=U&amp;ved=2ahUK</a>	PageRank:39	BLACK
<a href="http://www.gradreports.com/college">www.gradreports.com/college</a>	PageRank:38	BLACK
<a href="http://www.wscuc.org/institutions/">www.wscuc.org/institutions/</a>	PageRank:36	BLACK
<a href="http://www.sjsu.edu/&amp;sa=U&amp;ved=2ahU">www.sjsu.edu/&amp;sa=U&amp;ved=2ahU</a>	PageRank:34	BLACK
<a href="http://www.mercurynews.com/2020/12">www.mercurynews.com/2020/12</a>	PageRank:30	RED
<a href="http://www.mercurynews.com/2020/11">www.mercurynews.com/2020/11</a>	PageRank:27	RED
<a href="https://en.wikipedia.org/wiki/Weste">en.wikipedia.org/wiki/Weste</a>	PageRank:26	BLACK
<a href="https://en.wikipedia.org/wiki/Mary_">en.wikipedia.org/wiki/Mary_</a>	PageRank:24	RED
<a href="http://sjsunews.com/&amp;sa=U&amp;ved=2ahU">sjsunews.com/&amp;sa=U&amp;ved=2ahU</a>	PageRank:23	BLACK
<a href="http://www.cbssports.com/college-f">www.cbssports.com/college-f</a>	PageRank:17	BLACK
<a href="http://www.mathworks.com/academia/">www.mathworks.com/academia/</a>	PageRank:11	BLACK

-----

1

If the user presses another number, the application will terminate.



```

        selection = scanner.nextInt();
    }
    if( selection != 1 && selection != 2 && selection != 3){
        System.out.println("Thanks for using! Bye! ");
    }
}
}

```

Outpress:

```

-----
6
Thanks for using! Bye!
|

```

### Procedure to unzip files, install application and run code.

Window OS:

1. Unzip PA3-Wu.zip on prefer directory.
2. Run cmd (Command Prompt) on the computer.
3. Use command cd in cmd to change the directory to where RBTTree.jar, DoubleHashing.jar, HashingWithChaining.jar, and MultiplicationMethod.jar located.
4. Use command java -jar RBTTree.jar in cmd to run the program.
5. Use command java -jar DoubleHashing.jar in cmd to run the program.
6. Use command java -jar HashingWithChaining.jar in cmd to run the program.
7. Use command java -jar MultiplicationMethod.jar in cmd to run the program.

### Problems encountered during the implementation

One problem that I encountered during the implementation of the Red Black Tree and the Google Search Engine is figuring out a way to generate a random score with no duplicate. I first tried using Sets, but I didn't find out how to get the element in Set. Then, I tried to use TreeSet, but the number will be sorted in order when using it. Finally, I found out that I can use HashSet since HashSet don't allow duplicates and it will not sort in order. I also found out that I can use enhance for loop to get the elements in this set after a serverl test. I also have a hard time to print out the RB Tree since it is not like an Array or an ArrayList. Then, I figure out that I can print it like an inorder walk and show the parent, left child, and right child of every node as a visualization for the user. Other problems that have occurred when I am implementing the insertion method in Red Black Tree is NullPointerException and the RBTTree doesn't follow the

properties and I used the debug tool from the IDE to go through every step and try several possible solutions to solve the problem.

When I was implementing the multiplication method, I had a hard time on how to convert integers to binary numbers. Then, I had found out that there is a suggestion method from the IDE call toBinaryString(), which allows me to convert it to a binary string. After that I used this method and got the binary string as well as the significant bits using substring(). Another problem with using the toBinaryString() is that it can't convert numbers that are very big. Therefore, I limit the k value by just adding up the ascii code from every character of the String.

### **Lessons Learned**

From this assignment, I had learned how to handle NullPointerException using the Debug tool from the IDE. However, there are also unchecked exceptions that won't show up. These bugs are required to go through every step of the code and solve the problem. I also learned how to convert integers to binary strings and convert binary strings back to decimal. I had learn how to construct an RB tree using RB tree properties as well as perform sorting using RB Tree properties. With all the lessons I learned from this assignment, I can see that I am getting better on programming applications and being a better software engineer. There is a long way for me to be a professional software engineer. This assignment is just a part of my way to be a professional software engineer.