Data Structures

Red Black Trees

Ahmed Bahgat Elsherif	18010078
Abdallah Yasser Ibrahim	18015026
Youssef Hesham Akl	18012153
Moaz Fathy Eldferway	18011823
Mohamed Metwalli Noureldin	18011587

```
public interface INode<T extends Comparable<T>, V> {

    void setParent(INode<T, V> parent); → O(1)

    INode<T, V> getParent();    → O(1)

    void setLeftChild(INode<T, V> leftChild); → O(1)

    INode<T, V> getLeftChild(); → O(1)

    void setRightChild(INode<T, V> rightChild); → O(1)

    INode<T, V> getRightChild(); → O(1)

    T getKey(); → O(1)

    void setKey(T key); → O(1)

    V getValue(); → O(1)

    void setValue(V value); → O(1)

    boolean getColor(); → O(1)

    void setColor(boolean color); → O(1)

    boolean isNull(); → O(1)

}
public interface IRedBlackTree<T extends Comparable<T>, V> {
     public INode<T, V> getRoot(); --> O(1)

      public boolean isEmpty(); --> O(1)

      public void clear(); --> O(1) // garbage collector handle the
removal in O(n)

      public V search(T key) throws RuntimeErrorException;-->O(log(n))

     public boolean contains(T key); -->O(log(n))

      public void insert(T key, V value); -->O(log(n))

      public boolean delete(T key); -->O(log(n))

}
```

```
public interface ITreeMap<T extends Comparable<T>, V> {

  //  O(Log N)
  public Map.Entry<T, V> ceilingEntry(T key);

  //  O(Log N)
  public T ceilingKey(T key);

  //  O(1)     // Garbage Collector handles removal in O(n)
  public void clear();

  //  O(Log N)
  public boolean containsKey(T key);

  //  O(N)
  public boolean containsValue(V value);

  //  O(N)
  public Set<Map.Entry<T,V>> entrySet();
```

```
//  O(Log N)
public Map.Entry<T, V> firstEntry();

//  O(Log N)
public T firstKey();

//  O(Log N)
public Map.Entry<T, V> floorEntry(T key);

//  O(Log N)
public T floorKey(T key);

//  O(Log N)
public V get(T key);

//  O(N)
public ArrayList<Map.Entry<T, V>> headMap(T toKey);

//  O(N)
public ArrayList<Map.Entry<T, V>> headMap(T toKey, boolean inclusive);



//  O(N)
public Set<T> keySet();

 //  O( Log N)
public Map.Entry<T, V> lastEntry();

 //  O( Log N)
public T lastKey();

 //  O( Log N)
public Map.Entry<T, V> pollFirstEntry();

 //  O( Log N)
public Map.Entry<T, V> pollLastEntry();

 //  O( Log N)
public void put(T key, V value);

 //  O( Log N * k) where k = # of map elements
public void putAll(Map<T, V> map);
```

```
   //  O( Log N)
  public boolean remove(T key);

   //  O(1)
  public int size();


  //O(N)
   public Collection<V> values();

}
```