PART B:

In this code a balanced Binary tree is being created and for parallelization 2 and 4 multithreads are being used at a time.

In Main:

Array contains [10,1000,100000,1000000]

For I in Array:

for 2 threads:

code…

Without Thread:

Code….

For I in Array:

For 4 Thread:

Code…

Without thread:

Code…

Conclusion:

we observe as number of nodes increases multithreading provides us better result in less time.

For Example:

Let for N= 1000000

For 2 threads:

time taken: 679ms

for without Thread:

time taken: 1138ms

But for N = 10

For 2 threads:

Time taken: 555590ns

Without thread:

Time taken: 34567ns

MultiThreading:

For 2 threads:

```
AVLTree lefttree = new AVLTree(tree.root,givenarray[jk]/2,array1);
AVLTree righttree=new AVLTree(lefttree.root,givenarray[jk]/2, array2);
long starttime=System.nanoTime();
Thread t1=new Thread(lefttree);
Thread t_1= new Thread(righttree);
t1.start();t_1.start();
t1.join();t_1.join();
```

For 4 threads:

```
AVLTree left2tree = new AVLTree(tree.root,givenarray[jk]/4,array1);
AVLTree right2tree=new AVLTree(left2tree.root,givenarray[jk]/4, array2);
AVLTree leftrighttree=new AVLTree(right2tree.root,givenarray[jk]/4, array3);
AVLTree rightlefttree=new AVLTree(leftrighttree.root,givenarray[jk]/4, array4);
long starttime=System.nanoTime();
Thread t21=new Thread(left2tree);
Thread t_21= new Thread(right2tree);
Thread t22=new Thread(leftrighttree);
Thread t_22= new Thread(rightlefttree);
t21.start();t_21.start();t22.start();t_22.start();
t21.join();t_21.join();t22.join();t_22.join();
```

Generic:

```
class Array<E>{
    private final Object[] objectarray;
    public final int length;

    public Array(int length){
        objectarray= new Object [length];
        this.length=length;
    }

    public void set(int i, String string) {
        objectarray[i]=string;
    }

    void set(int i,long l){
        objectarray[i]=l;
    }

    void set2(int i,String l){
        objectarray[i]=l;
    }

    @Override
    public String toString() {
        return Arrays.toString(objectarray);
    }
}
```

I/O:

```
10
1000
100000
1000000
```

```
BufferedReader reader;

try {
    reader = new BufferedReader(new
    FileReader(fileName: "sample.
    txt"));
    String line = reader.readLine();

    while (line != null) {
        new_arr.add(Integer.parseInt
        (line));
        line = reader.readLine();
    }

    reader.close();
} catch (IOException e) {
    e.printStackTrace();
}
```

Data is being read from sample.txt and stored in an array.

RUNNING PROCESS:

On running the code balanced binary tree is made and for finding the node, first element of array is being found in the tree.

Code provides output for 2 threads and 4 threads respectively.