LAB-7

Name: V.SAIKRISHNA

Reg.No.:19BCE7638

1Q.Binary Search.Java Code.

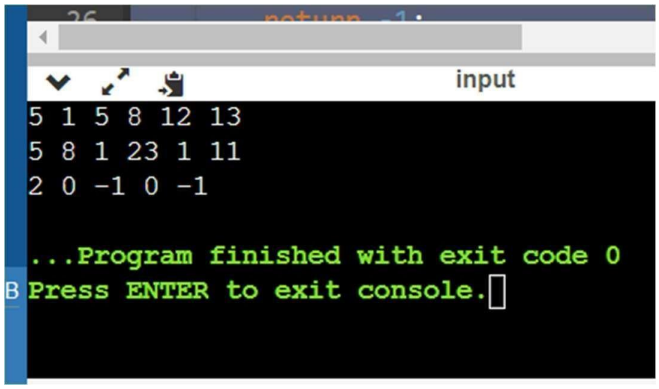
```
import java.io.*;
import java.util.*;


public class Main {
static int binarySearch(int[] a, int x)
{
int left = 0, right = a.length - 1;
    while (left <= right) {
        int mid = left + (right - left) /
        2;

        if (x == a[mid]) {
            return mid;
        } else if (x <
            a[mid]) { right =
            mid - 1;
        } else {
            left = mid + 1;
        }
    }
    return -1;
```

```
static int linearSearch(int[] a, int x)
   { for (int i = 0; i < a.length; i++) {
      if (a[i] == x)
         return i;
   }
   return -1;
}

public static void main(String[] args) {
   FastScanner scanner = new
   FastScanner(System.in); int n =
   scanner.nextInt();
   int[] a = new int[n];
   for (int i = 0; i < n; i++) {
      a[i] =
      scanner.nextInt();
   }
   int m =
   scanner.nextInt(); int[]
   b = new int[m];
   for (int i = 0; i < m; i++)
      {
    b[i] = scanner.nextInt();
   }
```

```
    for (int i = 0; i < m; i++) {

        System.out.print(binarySearch(a, b[i]) + " ");

    }
}

static class FastScanner {
    BufferedReader br;
    StringTokenizer st;

    FastScanner(InputStream
        stream) { try {
            br = new BufferedReader(new
            InputStreamReader(stream));
        } catch (Exception
            e) {
            e.printStackTrace
            ();
        }
    }
```

```
        String next() {
          while (st == null ||
            !st.hasMoreTokens()) { try {
              st = new StringTokenizer(br.readLine());
            } catch (IOException e) {
              e.printStackTrace();
            }
          }
          return st.nextToken();

}

        int nextInt() {
          return Integer.parseInt(next());
        }
      }
   }
```

Output:



2Q: Max votes.

Code:

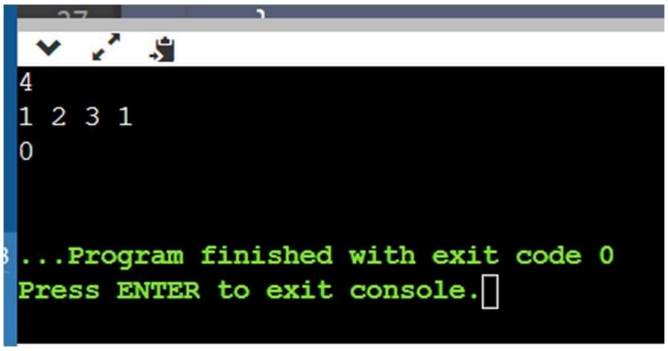
```
import
java.util.*;
import java.io.*;

public class Main {
   private static int getMaxVote(int[] a, int left, int right)
      { if (left == right) {
         return -1;
      }
```

```
    if (left + 1 == right) {
        return a[left];
    }
    int left_elem = getMaxVote(a, left, (left + right - 1) / 2 +
    1); int right_elem = getMaxVote(a, (left + right - 1) / 2 +
    1, right);

    int lcount = 0;
    for (int i = left; i < right; i++)
        { if (a[i] == left_elem)
            lcount += 1;
    }
    if (lcount > (right - left) / 2)
        return left_elem;

    int rcount = 0;
    for (int i = left; i < right; i++)
        { if (a[i] == right_elem)
            rcount += 1;
    }
    if (rcount > (right - left) / 2)
        return right_elem;
    return -1;
}
```

```
public static void main(String[] args) {
   FastScanner scanner = new
   FastScanner(System.in); int n =
   scanner.nextInt();
   int[] a = new int[n];
   for (int i = 0; i < n; i++) {
      a[i] =
      scanner.nextInt();
   }
   if (getMaxVote(a, 0, a.length) !=
      -1) { System.out.println(1);
   } else {
      System.out.println(0);



   }
}

static class FastScanner {
   BufferedReader br;
   StringTokenizer st;

   FastScanner(InputStream
      stream) { try {
         br = new BufferedReader(new
         InputStreamReader(stream));
      } catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }

    String next() {
        while (st == null ||
            !st.hasMoreTokens()) { try {
                st = new StringTokenizer(br.readLine());
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
        return st.nextToken();
    }

    int nextInt() {
        return Integer.parseInt(next());
    }
  }
}
```

Output:



2Q. Maximum Votes import java.util.*; import java.io.*;

```
public class Main {
   private static int getMaxVote(int[] a, int left, int right)
      { if (left == right) {
         return -1;

      }
      if (left + 1 == right)
         { return a[left];
      }

      int left_elem = getMaxVote(a, left, (left + right - 1) / 2 +
      1); int right_elem = getMaxVote(a, (left + right - 1) / 2 +
      1, right);

      int lcount = 0;
      for (int i = left; i < right;
         i++) { if (a[i] ==
         left_elem)
            lcount += 1;
      }

      if (lcount > (right - left) / 2)
         return left_elem;

      int rcount = 0;
```

```
for (int i = left; i < right;
   i++) { if (a[i] ==
   right_elem)
      rcount += 1;
}

if (rcount > (right - left) / 2)
```

```
        return
    right_elem; return
    -1;
  }

  public static void main(String[] args) {

    FastScanner scanner = new
    FastScanner(System.in); int n =
    scanner.nextInt();
    int[] a = new int[n];
    for (int i = 0; i < n; i++) {
      a[i] =
      scanner.nextInt();
    }

    if (getMaxVote(a, 0, a.length) != -1) {
      System.out.println(1);
    } else {

      System.out.println(0);



    }

  }


  static class
    FastScanner {
    BufferedReader br;
    StringTokenizer st;

    FastScanner(InputStream
      stream) { try {
        br = new BufferedReader(new
InputStreamReader(stream));
```

```
    } catch (Exception
       e) {
       e.printStackTrace
       ();
    }

}

String next() {
    while (st == null || !st.hasMoreTokens()) {
       try {
```

```
            st = new StringTokenizer(br.readLine());

            } catch (IOException
                        e) {
             e.printStackTrace();
            }
          }
          return st.nextToken();

      }

      int nextInt() {
         return Integer.parseInt(next());
      }

   }
}
```