DIVIDE AND CONQUER – LAB 02

EC 4070

DATA STRUCTURES AND ALGORITHMS

NAME	**: WIJAYAWARDHANA W.A.H.A.**
REGISTRATION NO.	**: 2019/E/166**
SEMESTER	**: SEMESTER 04**
DATE ASSIGNED	**: 03 MARCH 2022**

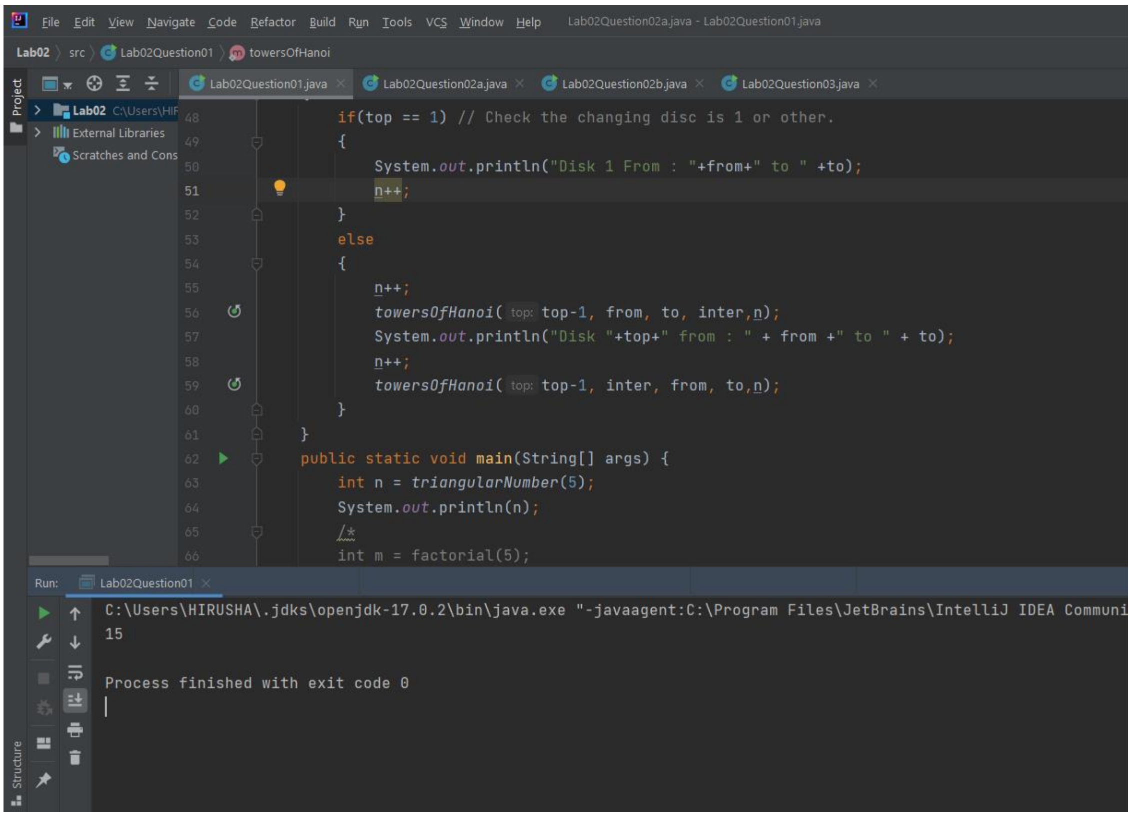
01.

Code:-

```
public class Lab02Question01 {
   // Triangular number
   public static int triangularNumber(int number)  // number variable is to provide the required triangular number.
   {
      if(number == 1) // Factorial of 1 is 1.
      {
         return 1;
      }
      else   // If input is higher than one the recursive will call for summation of number.
      {
         return (number+triangularNumber(number-1));  // Recursively call triangularNumber function.
      }
   }

   // Factorial.
   public static int factorial(int number) // Factorial method is here.
   {
      if(number > 1)  // Until 1 the recursive will run.
      {
         return (number * factorial(number -1));  // Recursively call the function.
      }
      else
      {
         return 1; // When it reaches 1 it will return 1.
      }
   }


   // Anagrams
   public static int anagramsMethod(String word , int n)  // Insert the word and calling the method for anagrams.
   {
      if(word.length()==1)  // If the word only have 1 letter can not do any change.
      {
         return 1;
      }
      for(int i =0; i<word.length(); i++)
      {
         n++;
         return (anagramsMethod(word,n));
      }

      return 0;
   }

   // Towers of Hanoi
   public static void towersOfHanoi(int top,char from,char inter, char to , int n) // Calling the method.
   {
      if(top == 1) // Check the changing disc is 1 or other.
      {
         System.out.println("Disk 1 From : "+from+" to " +to);
         n++;
      }
```

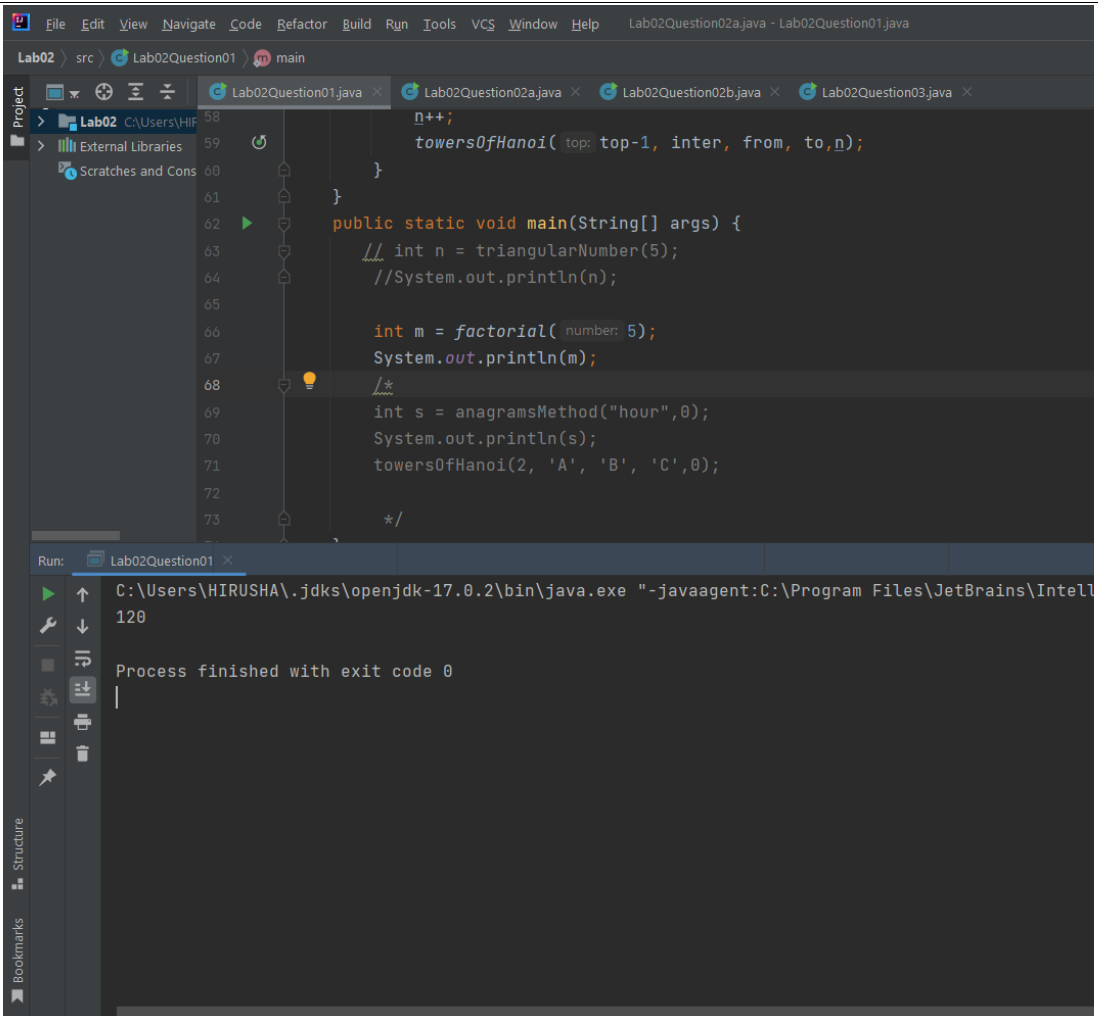
```
        else
        {
            n++;
            towersOfHanoi(top-1, from, to, inter,n);
            System.out.println("Disk "+top+" from : " + from +" to " + to);
            n++;
            towersOfHanoi(top-1, inter, from, to,n);
        }
    }
    public static void main(String[] args) {
        int n = triangularNumber(5);
        System.out.println(n);
        int m = factorial(5);
        System.out.println(m);
        int s = anagramsMethod("hour",0);
        System.out.println(s);
        towersOfHanoi(2, 'A', 'B', 'C',0);
    }
}
```

Outputs:-



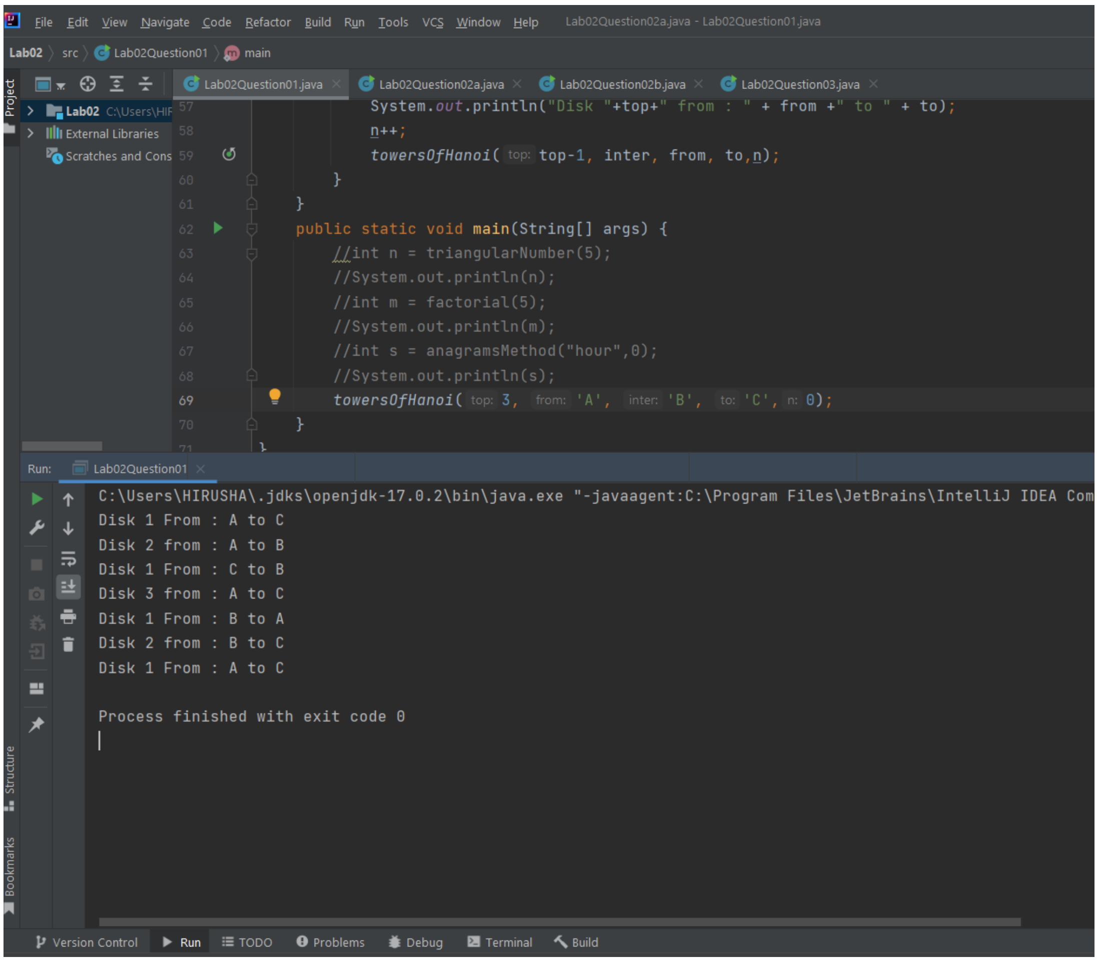
```
        if(top == 1) // Check the changing disc is 1 or other.
        {
            System.out.println("Disk 1 From : "+from+" to " +to);
            n++;
        }
        else
        {
            n++;
            towersOfHanoi( top: top-1, from, to, inter,n);
            System.out.println("Disk "+top+" from : " + from +" to " + to);
            n++;
            towersOfHanoi( top: top-1, inter, from, to,n);
        }
    }
    public static void main(String[] args) {
        int n = triangularNumber(5);
        System.out.println(n);
        /*
        int m = factorial(5);
```

```
C:\Users\HIRUSHA\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Communi
15

Process finished with exit code 0
```



```java
            n++;
            towersOfHanoi( top: top-1, inter, from, to,n);
        }
    }
    public static void main(String[] args) {
       // int n = triangularNumber(5);
        //System.out.println(n);

        int m = factorial( number: 5);
        System.out.println(m);
        /*
        int s = anagramsMethod("hour",0);
        System.out.println(s);
        towersOfHanoi(2, 'A', 'B', 'C',0);

         */
```

```
C:\Users\HIRUSHA\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\Intell
120

Process finished with exit code 0
```



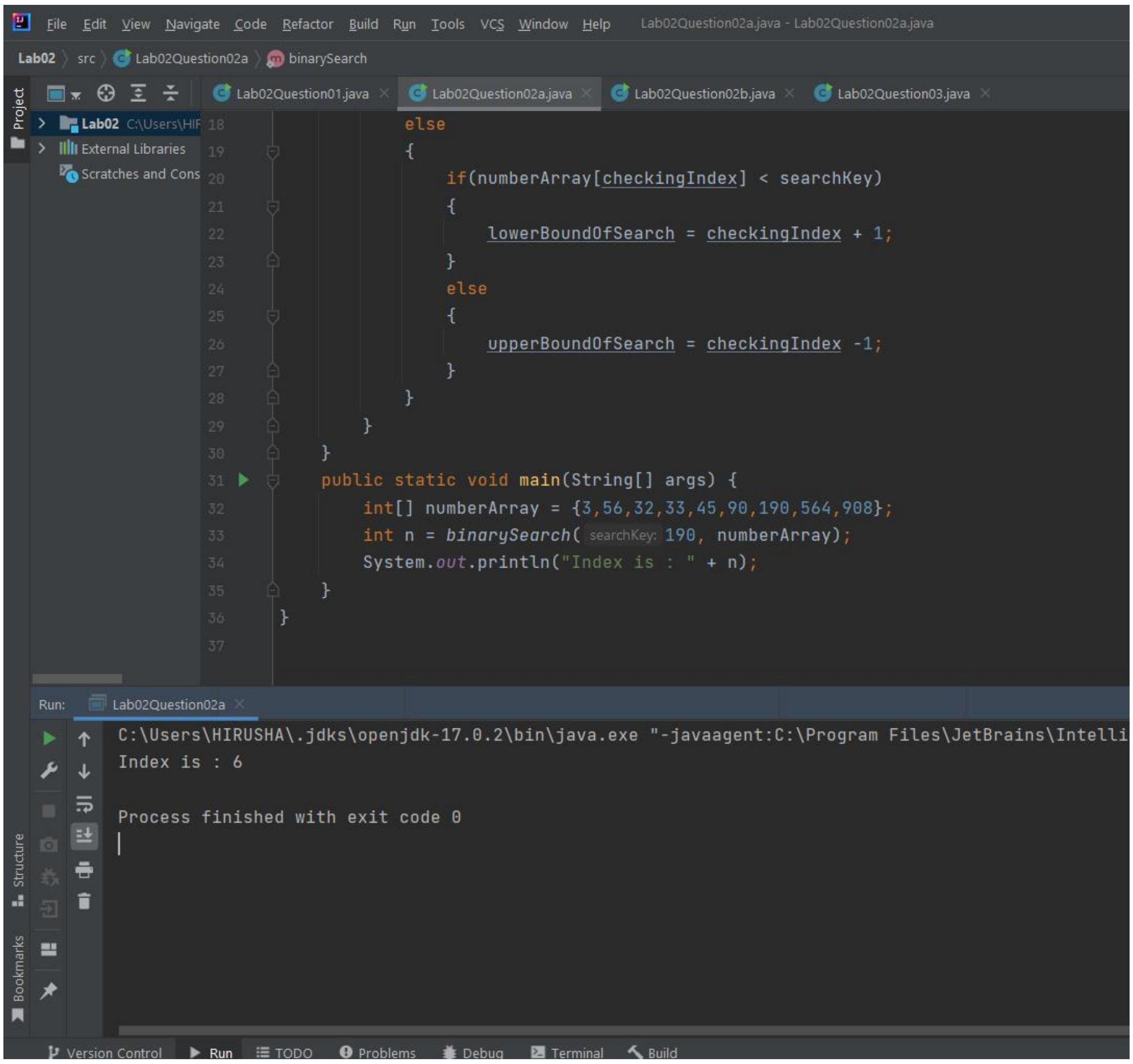
```java
            System.out.println("Disk "+top+" from : " + from +" to " + to);
            n++;
            towersOfHanoi( top: top-1, inter, from, to,n);
        }
    }
    public static void main(String[] args) {
        //int n = triangularNumber(5);
        //System.out.println(n);
        //int m = factorial(5);
        //System.out.println(m);
        //int s = anagramsMethod("hour",0);
        //System.out.println(s);
        towersOfHanoi( top: 3,  from: 'A',  inter: 'B',  to: 'C', n: 0);
    }
```

```
C:\Users\HIRUSHA\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Com
Disk 1 From : A to C
Disk 2 from : A to B
Disk 1 From : C to B
Disk 3 from : A to C
Disk 1 From : B to A
Disk 2 from : B to C
Disk 1 From : A to C

Process finished with exit code 0
```

02.
a.
Code:

```
public class Lab02Question02a {
   public static int binarySearch(long searchKey,int[] numberArray)
   {
      int lowerBoundOfSearch = 0;
      int upperBoundOfSearch = numberArray.length-1;
      int checkingIndex = 0;
      while(true)
      {
         checkingIndex = (lowerBoundOfSearch+upperBoundOfSearch)/2;
         if(numberArray[checkingIndex] == searchKey)
         {
            return  checkingIndex;
         }
         else if(lowerBoundOfSearch > upperBoundOfSearch)
         {
            return numberArray.length-1;
         }
         else
         {
            if(numberArray[checkingIndex] < searchKey)
            {
               lowerBoundOfSearch = checkingIndex + 1;
            }
            else
            {
               upperBoundOfSearch = checkingIndex -1;
            }
         }
      }
   }
   public static void main(String[] args) {
      int[] numberArray = {3,56,32,33,45,90,190,564,908};
      int n = binarySearch(190, numberArray);
      System.out.println("Index is : " + n);
   }
}
```

Output:-



```
            else
            {
                if(numberArray[checkingIndex] < searchKey)
                {
                    lowerBoundOfSearch = checkingIndex + 1;
                }
                else
                {
                    upperBoundOfSearch = checkingIndex -1;
                }
            }
        }
    }
    public static void main(String[] args) {
        int[] numberArray = {3,56,32,33,45,90,190,564,908};
        int n = binarySearch( searchKey: 190, numberArray);
        System.out.println("Index is : " + n);
    }
}
```

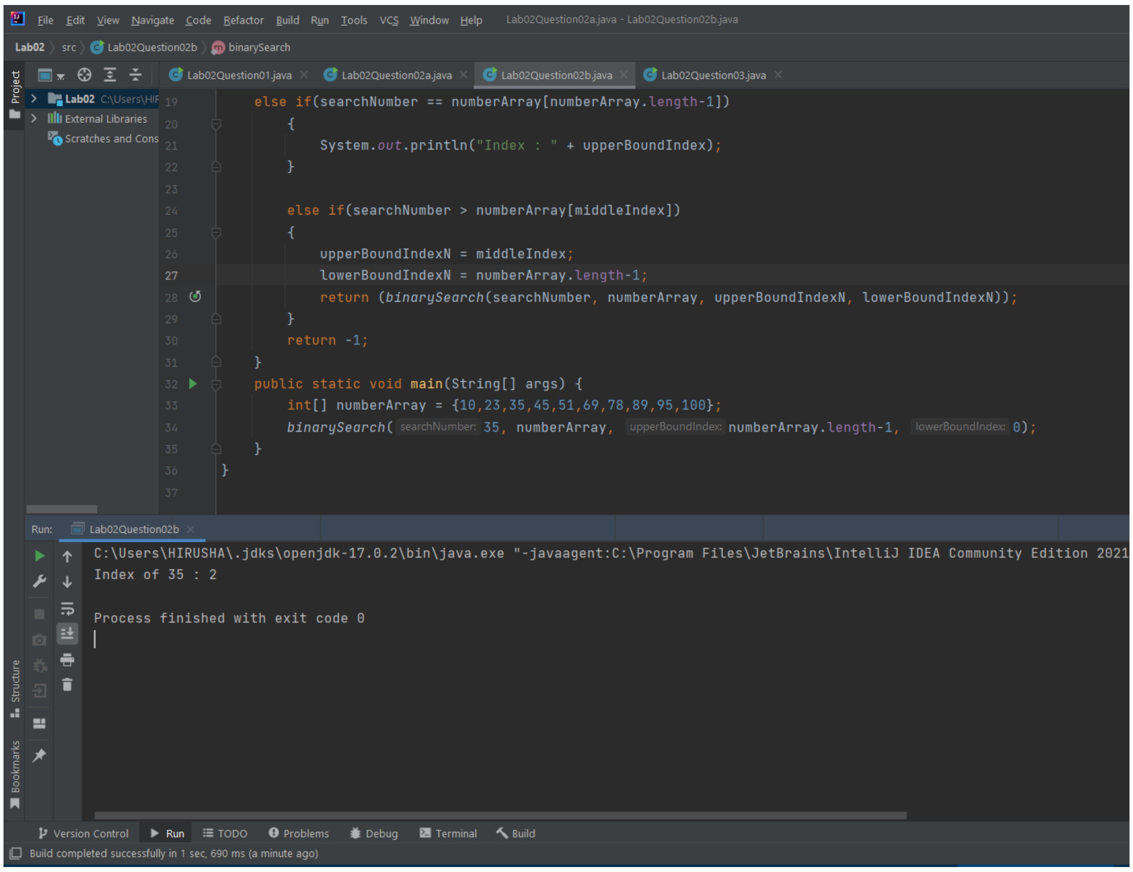
```
C:\Users\HIRUSHA\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\Intelli
Index is : 6

Process finished with exit code 0
```

b.

Code:

```
public class Lab02Question02b {
    public static int binarySearch(int searchNumber , int[] numberArray , int upperBoundIndex , int lowerBoundIndex)
    {
        int middleIndex = (upperBoundIndex + lowerBoundIndex)/2;
        int upperBoundIndexN;
        int lowerBoundIndexN;
        if(searchNumber == numberArray[middleIndex])
        {
            System.out.println("Index of " + searchNumber + " : " + middleIndex);
            return middleIndex;
        }
        else if(searchNumber < numberArray[middleIndex])
        {
            upperBoundIndexN = 0;
            lowerBoundIndexN = middleIndex;
            return (binarySearch(searchNumber, numberArray, upperBoundIndexN,
lowerBoundIndexN));
        }

         else if(searchNumber == numberArray[numberArray.length-1])
        {
            System.out.println("Index : " + upperBoundIndex);
        }

        else if(searchNumber > numberArray[middleIndex])
        {
            upperBoundIndexN = middleIndex;
            lowerBoundIndexN = numberArray.length-1;
            return (binarySearch(searchNumber, numberArray, upperBoundIndexN,
lowerBoundIndexN));
        }
        return -1;
    }
    public static void main(String[] args) {
        int[] numberArray = {10,23,35,45,51,69,78,89,95,100};
        binarySearch(35, numberArray, numberArray.length-1, 0);
    }
}
```

Output:



```
        else if(searchNumber == numberArray[numberArray.length-1])
        {
            System.out.println("Index : " + upperBoundIndex);
        }

        else if(searchNumber > numberArray[middleIndex])
        {
            upperBoundIndexN = middleIndex;
            lowerBoundIndexN = numberArray.length-1;
            return (binarySearch(searchNumber, numberArray, upperBoundIndexN, lowerBoundIndexN));
        }
        return -1;
    }
    public static void main(String[] args) {
        int[] numberArray = {10,23,35,45,51,69,78,89,95,100};
        binarySearch( searchNumber: 35, numberArray,  upperBoundIndex: numberArray.length-1,  lowerBoundIndex: 0);
    }
}
```

```
C:\Users\HIRUSHA\.jdks\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021
Index of 35 : 2

Process finished with exit code 0
```

Build completed successfully in 1 sec, 690 ms (a minute ago)

03.

Code:

```
public class Lab02Question03b {
   public static int partition(int[] sortingArray, int lowerBoundIndex, int upperBoundIndex)
   {
      int pivotElement = sortingArray[lowerBoundIndex];
      int startingIndex = lowerBoundIndex;
      int endingIndex = upperBoundIndex;
      while(startingIndex < endingIndex)
      {
         while(sortingArray[startingIndex] <= pivotElement)
         {
            startingIndex++;
         }
         while (sortingArray[endingIndex] > pivotElement)
         {
            endingIndex--;
         }
         if(startingIndex < endingIndex)
         {
            swapElements(sortingArray,startingIndex,endingIndex);
         }
      }
      swapElements(sortingArray,startingIndex,endingIndex );
      return endingIndex;
   }
   public static void swapElements(int[] sortingArray , int swapIndex01, int swapIndex02)
   {
      int tempValue = sortingArray[swapIndex01];
      sortingArray[swapIndex01] = sortingArray[swapIndex02];
      sortingArray[swapIndex02] = tempValue;
   }

   public static void quickSort(int[] sortingArray, int lowerBoundIndex, int upperBoundIndex)
   {
      if(lowerBoundIndex < upperBoundIndex)
      {
         int tempValue = partition(sortingArray,lowerBoundIndex,upperBoundIndex);
         quickSort(sortingArray,lowerBoundIndex,lowerBoundIndex-1);
         quickSort(sortingArray,lowerBoundIndex+1,upperBoundIndex);
      }
   }

   public static void main(String[] args) {
      int[] sortingArray = new int[8];
```

```
            sortingArray[0] = 12;
            sortingArray[1] = 69;
            sortingArray[2] = 3;
            sortingArray[3] = 14;
            sortingArray[4] = 8;
            sortingArray[5] = 18;
            sortingArray[6] = 89;
            sortingArray[7] = 65;
            partition(sortingArray,0,7);
            quickSort(sortingArray,0,7);
            for(int i =0; i < sortingArray.length; i++)
            {
              System.out.println(sortingArray[i]);
            }
        }
    }
```