

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 RecursiveFunctions {
    // 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));
        }
    }
}
```

```

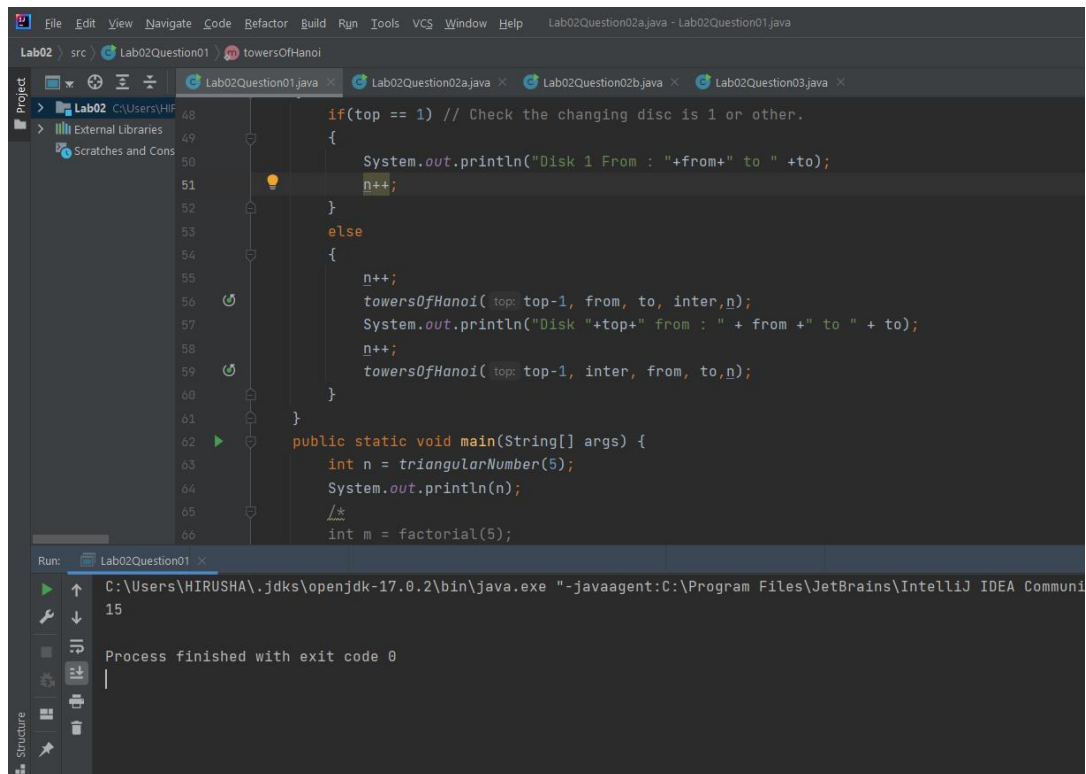
        return 0;
    }

    // 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(3, 'A', 'B', 'C',0);
}
}

```

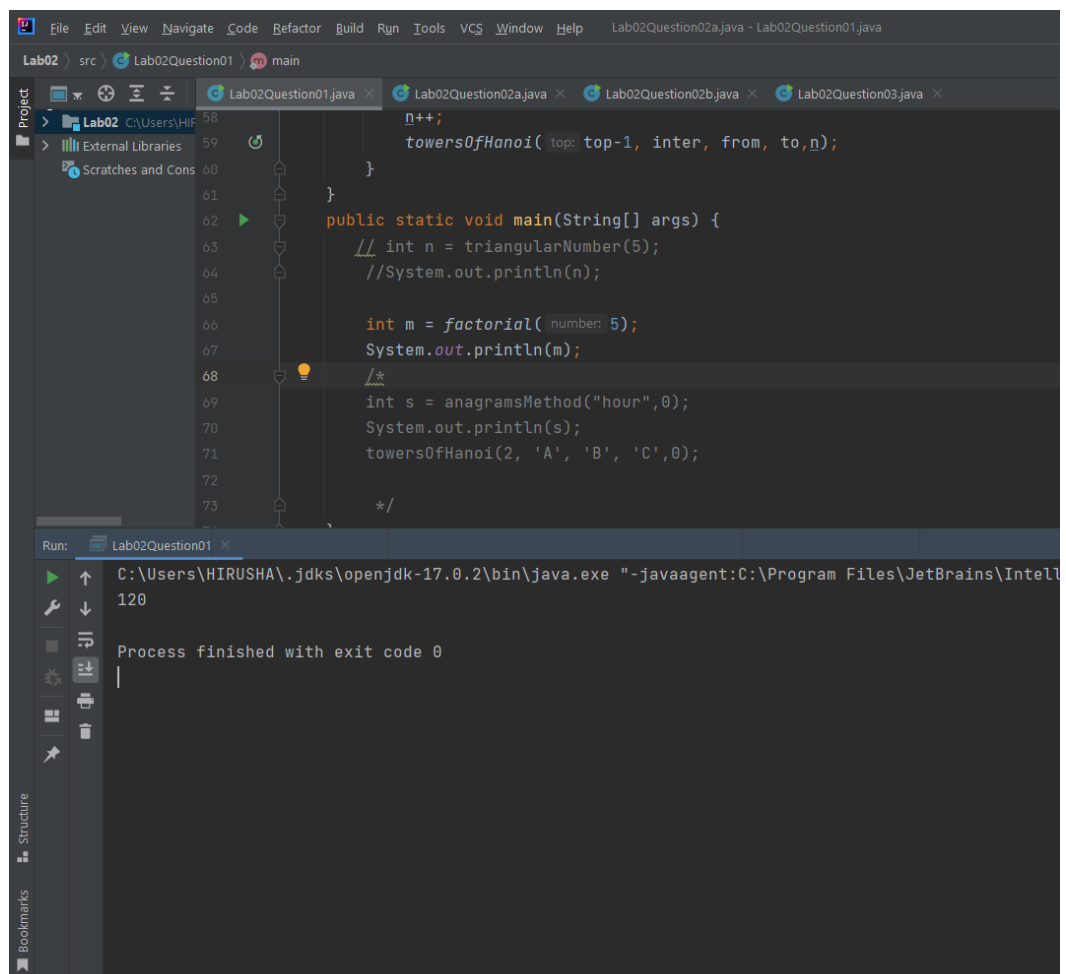
Outputs:-



The screenshot shows the IntelliJ IDEA IDE with a Java project named 'Lab02'. The main editor displays the 'towersOfHanoi' class. The code includes a recursive method 'towersOfHanoi' and a 'main' method. The 'main' method calls 'triangularNumber(5)' and 'factorial(5)'. The output window at the bottom shows the execution of 'Lab02Question01' with the output '15' and 'Process finished with exit code 0'.

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Lab02Question02a.java - Lab02Question01.java
Lab02 src Lab02Question01 towersOfHanoi
Project
  Lab02 C:\Users\HIR...
  External Libraries
  Scratches and Cons
Lab02Question01.java
  48
  49
  50
  51
  52
  53
  54
  55
  56
  57
  58
  59
  60
  61
  62
  63
  64
  65
  66
  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);
  Run: Lab02Question01
  C:\Users\HIRUSHA\jdk\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Commun
  15
  Process finished with exit code 0
```

FIGURE 01 – TRIANGULAR NUMBER OUTPUT



The screenshot shows the IntelliJ IDEA IDE with a Java project named 'Lab02'. The main editor displays the 'main' class. The code includes a 'main' method that calls 'triangularNumber(5)', 'factorial(5)', and 'anagramsMethod("hour", 0)'. The output window at the bottom shows the execution of 'Lab02Question01' with the output '120' and 'Process finished with exit code 0'.

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Lab02Question02a.java - Lab02Question01.java
Lab02 src Lab02Question01 main
Project
  Lab02 C:\Users\HIR...
  External Libraries
  Scratches and Cons
Lab02Question01.java
  58
  59
  60
  61
  62
  63
  64
  65
  66
  67
  68
  69
  70
  71
  72
  73
  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);
  */
}
Run: Lab02Question01
C:\Users\HIRUSHA\jdk\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\Intelli
120
Process finished with exit code 0
```

FIGURE 02 – FACTORIAL OUTPUT

The screenshot displays the IntelliJ IDEA IDE with a Java project named 'Lab02'. The main editor shows the 'main' method of 'Lab02Question01.java'. The code implements the Towers of Hanoi algorithm for 3 disks, moving them from peg 'A' to 'C' using 'B' as an intermediate. The output window shows the sequence of moves: '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', and 'Disk 1 From : A to C'. The process finished with exit code 0.

```
File Edit View Navigate Code Refactor Build Run Tools VCS Window Help Lab02Question02a.java - Lab02Question01.java
Lab02 src Lab02Question01 Lab02Question01 main
Project Lab02 C:\Users\HIR...
  External Libraries
  Scratches and Cons
Lab02Question01.java Lab02Question02a.java Lab02Question02b.java Lab02Question03.java
57 System.out.println("Disk "+top+" from : " + from + " to " + to);
58 n++;
59 towersOfHanoi( top-1, inter, from, to,n);
60 }
61 }
62 public static void main(String[] args) {
63     //int n = triangularNumber(5);
64     //System.out.println(n);
65     //int m = factorial(5);
66     //System.out.println(m);
67     //int s = anagramsMethod("hour",0);
68     //System.out.println(s);
69     towersOfHanoi( top: 3, from: 'A', inter: 'B', to: 'C', n: 0);
70 }
71 }

Run: Lab02Question01
C:\Users\HIRUSHA\.jdk\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
|

Version Control Run TODO Problems Debug Terminal Build
```

FIGURE 03 – TOWERS OF HANOI OUTPUT

02.

a.

Code:

```
public class BinarySearchUsingLoop {
    public static int binarySearch(long searchKey,int[] numberArray) // Binary search method.
    {
        int lowerBoundOfSearch = 0; // Lower bound define and assign.
        int upperBoundOfSearch = numberArray.length-1; // Upper bound define and assign.
        int checkingIndex = 0; // Checking index define and assign as 0.
        while(true) // While condition runs true.
        {
            checkingIndex = (lowerBoundOfSearch+upperBoundOfSearch)/2; // Setting middle index
            for searching element.
            if(numberArray[checkingIndex] == searchKey) // Check the middle element equal to
            search element.
            {
                return checkingIndex; // Return the index after equal.
            }
            else if(lowerBoundOfSearch > upperBoundOfSearch)
            {
                return numberArray.length-1;
            }
            else // If upper conditions are not true.
            {
                if(numberArray[checkingIndex] < searchKey) // Check search element less than the
                arrays checkingIndex element.
                {
                    lowerBoundOfSearch = checkingIndex + 1; // If it true search element at right side so
                    move to right side.
                }
                else
                {
                    upperBoundOfSearch = checkingIndex -1; // If not search element is less so element
                    is at left side.
                }
            }
        }
    }
    public static void main(String[] args) {
        int[] numberArray = {3,56,32,33,45,90,190,564,908}; // Define sorted array for searching
        element.
        int n = binarySearch(190, numberArray); // Calling the binarySearch method.
        System.out.println("Index is : " + n); // Print the index of searched element.
    }
}
```

Output:-

```

File Edit View Nave Code Refactor Build Run Tools VCS Window Help Lab02Question02a.java - Lab02Question02a.java
Lab02 src Lab02Question02a binarySearch
Project
  Lab02 C:\Users\HIR...
    External Libraries
    Scratches and Cons
Lab02Question01.java Lab02Question02a.java Lab02Question02b.java Lab02Question03.java
48 else
19 {
20   if(numberArray[checkingIndex] < searchKey)
21   {
22     lowerBoundOfSearch = checkingIndex + 1;
23   }
24   else
25   {
26     upperBoundOfSearch = checkingIndex - 1;
27   }
28 }
29 }
30 }
31 public static void main(String[] args) {
32   int[] numberArray = {3,56,32,33,45,90,190,564,908};
33   int n = binarySearch( searchKey: 190, numberArray);
34   System.out.println("Index is : " + n);
35 }
36 }
37
Run: Lab02Question02a
C:\Users\HIRUSHA\.jdk\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ
Index is : 6
Process finished with exit code 0
Structure
Bookmarks
Version Control Run TODO Problems Debug Terminal Build

```

FIGURE 04 – BINARY SEARCH USING LOOP OUTPUT

b.

Code:

```
public class BinarySearchRecursive {
    // Binary searching method.
    public static int binarySearch(int searchNumber , int[] numberArray , int upperBoundIndex , int
lowerBoundIndex)
    {
        int middleIndex = (upperBoundIndex + lowerBoundIndex)/2; // Define middleIndex
according to passed values.
        int upperBoundIndexN;    // Define new variable.
        int lowerBoundIndexN;
        if(searchNumber == numberArray[middleIndex]) // Check the search element equal to
middle index's element.
        {
            System.out.println("Index of " + searchNumber + " : " + middleIndex); // If true that will
print and exit the code.
            return middleIndex;
        }
        else if(searchNumber < numberArray[middleIndex]) // If searchNumber less than
middleIndex's element that element should at left side of array.
        {
            upperBoundIndexN = 0;          // Define variable according to the identification of
element value.
            lowerBoundIndexN = middleIndex;
            return (binarySearch(searchNumber, numberArray, upperBoundIndexN,
lowerBoundIndexN)); // Recall the method.
        }

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

        else if(searchNumber > numberArray[middleIndex]) // If searchNumber higher than
middleIndex's element that element should at right side of array.
        {
            upperBoundIndexN = middleIndex; // Define variable according to the identification of
element value.
            lowerBoundIndexN = numberArray.length-1;
            return (binarySearch(searchNumber, numberArray, upperBoundIndexN,
lowerBoundIndexN)); // Recall the method.
        }
        return -1;
    }
    public static void main(String[] args) {
```



```

int[] numberArray = {10,23,35,45,51,69,78,89,95,100};    // Define the sorted array for
searching.
    binarySearch(35, numberArray, numberArray.length-1, 0); // Calling binarySearch method.
}
}

```

Output:

The screenshot displays the IntelliJ IDEA IDE with a Java project named 'Lab02'. The main editor shows the implementation of a recursive binary search method. The code defines a sorted array 'numberArray' and calls the 'binarySearch' method with the target value 35. The method uses recursive calls to find the index of the target value. The output window shows the result: 'Index of 35 : 2'.

```

19      else if(searchNumber == numberArray[numberArray.length-1])
20      {
21          System.out.println("Index : " + upperBoundIndex);
22      }
23
24      else if(searchNumber > numberArray[middleIndex])
25      {
26          upperBoundIndexN = middleIndex;
27          lowerBoundIndexN = numberArray.length-1;
28          return (binarySearch(searchNumber, numberArray, upperBoundIndexN, lowerBoundIndexN));
29      }
30      return -1;
31  }
32  public static void main(String[] args) {
33      int[] numberArray = {10,23,35,45,51,69,78,89,95,100};
34      binarySearch( searchNumber: 35, numberArray, upperBoundIndex: numberArray.length-1, lowerBoundIndex: 0);
35  }
36  }
37

```

Run: Lab02Question02b

```

C:\Users\HIRUSHA\.jdk\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)

FIGURE 05 – BINARY SEARCH RECURSIVE METHOD OUTPUT

03.

Merge sort

Code:

```
public class MergeSort {
    // Define required variables.
    int arraySize;
    int[] array = new int[arraySize];
    int lowerIndex;
    int upperIndex;
    int middleIndex;
    public void setElement(int[] array,int arraySize) // Setting the array and array size.
    {
        this.arraySize = arraySize;
        this.array = array;
    }
    // Merge sort method for separating the element.
    public void mergeSort(int lowerIndex, int upperIndex)
    {
        this.lowerIndex = lowerIndex; // Assign values into class variables.
        this.upperIndex = upperIndex; // Assign values into class variables.
        if(lowerIndex < upperIndex) // Check the condition is true until one element come to array
        this will true.
        {
            int middle = (lowerIndex+upperIndex)/2; // Define and assign middle index.
            mergeSort(lowerIndex,middle); // Calling mergeSort method for left side of array.
            mergeSort(middle+1,upperIndex); // mergeSort will call for right side of array.
            merge(lowerIndex,middle,upperIndex); // After finishing dividing array to one element
            the merge will call for combining the array.
        }
    }
    // Merge method for getting the sorted array.
    public void merge(int lowerIndex, int middleIndex, int upperIndex)
    {
        this.lowerIndex = lowerIndex; // Assign values into class variables.
        this.middleIndex = middleIndex; // Assign values into class variables.
        this.upperIndex = upperIndex; // Assign values into class variables.
        int tempArray[] = new int[upperIndex-lowerIndex+1]; // Define temporary array to store
        sorted element temporary.
        int i = lowerIndex; // Setting values.
        int j = middleIndex+1; // Setting values.
        for(int k = 0; (i <=middleIndex) || (j<=upperIndex); k++) // Setting values for temporary
        array from unsorted array.
        {
            if(i > middleIndex) // According to the value this will put values in sorted array.
            {
                tempArray[k] = array[j++];
```

```

    }
    else if(j > upperIndex)
    {
        tempArray[k] = array[i++];
    }
    else if(array[i] <= array[j])
    {
        tempArray[k] = array[i++];
    }
    else
    {
        tempArray[k] = array[j++];
    }
}
for(int index = lowerIndex; index < tempArray.length; index++)
{
    array[index] = tempArray[index]; // Put the sorted elements into first array.
}
}

public static void main(String[] args) {
    MergeSort newObject = new MergeSort(); // Define an object of the class.
    int[] arrayN = new int[]{12,89,65,34,1,66,78,99}; // Define an array for sorting.
    newObject.setElement(arrayN,arrayN.length); // Calling setElement method for setting
an array for sorting.
    newObject.mergeSort(0,arrayN.length-1); // Call mergeSort method for sorting.
    for(int i =0; i< arrayN.length; i++) // Print the sorted array.
    {
        System.out.print(arrayN[i]+" ");
    }
}
}

```

Output:-

The screenshot shows an IDE with a Java file named 'Lab02.iml'. The code is the same as shown in the previous block. The output window at the bottom shows the execution of the program, displaying the sorted array: 1 12 65 66 67 89 99. The process finished with exit code 0.

FIGURE 06 – MERGE SORT OUTPUT

Quick sort

Code:-

```
public class QuickSort {
    int arraySize; // Define arraySize.
    int[] arrayElement = new int[arraySize]; // Define array for element.
    public int partition(int[] arrayElement, int lowerIndex, int higherIndex)
    {
        this.arrayElement = arrayElement; // Class object assign for method calling array.
        int pivotElement = arrayElement[higherIndex]; // Assign pivot element as last element.
        int i = lowerIndex - 1;
        for(int j = lowerIndex; j <= higherIndex-1; j++)
        {
            if(arrayElement[j] < pivotElement) // Check pivot element less than element.
            {
                i++;
                swapElement(i,j); // If condition true swap will do.
            }
        }
        swapElement(i+1,higherIndex); // Otherwise that is the highest value then it will swap.
        return(i+1);
    }

    public void swapElement(int swapElement01 , int swapElement02) // Swap elements method.
    {
        int temporaryElement = arrayElement[swapElement01]; // Store in temporary value
        before swap.
        arrayElement[swapElement01] = arrayElement[swapElement02]; // Swap elements.
        arrayElement[swapElement02] = temporaryElement; // Swap elements.
    }

    // Quick sort method.
    public void quickSort(int[] arrayElement , int lowerElement , int higherElement)
    {
        this.arrayElement = arrayElement; // Assign the array in class.
        if(lowerElement < higherElement) // Condition check high or low.
        {
            int partitionElement = partition(arrayElement,lowerElement,higherElement); // Call
            partitionElement method.
            quickSort(arrayElement,lowerElement,partitionElement-1); // Call quick sort method
            for left part.
            quickSort(arrayElement,partitionElement+1,higherElement); // Call quick sort method
            for right part.
        }
    }
}
```

```

    }
    public void printArray(int array[])
    {
        for(int i =0;i<array.length;i++)
        {
            System.out.print(array[i]+" "); // Print array element.
        }
    }
}

public static void main(String[] args) {
    int newArray[] = {23,45,12,78,56,79,90};
    QuickSort newObject = new QuickSort(); // Create object of QuickSort.
    newObject.quickSort(newArray,0,newArray.length-1); // Calling quickSort method.
    newObject.printArray(newArray); // Calling printArray method.
}
}

```

Output:-

The screenshot shows an IDE with a Java file named 'QuickSort'. The code in the editor is as follows:

```

46     }
47 }
48 public static void main(String[] args) {
49     int newArray[] = {23,45,12,78,56,79,90,34,1}; // Define and assign an array.
50     QuickSort newObject = new QuickSort(); // Create object of QuickSort.
51     newObject.quickSort(newArray, lowerElement: 0, higherElement: newArray.length-1); // Calling quickSort method.
52     newObject.printArray(newArray); // Calling printArray method.
53 }
54 }
55

```

The Run window at the bottom shows the command executed and the output:

```

Run: QuickSort
C:\Users\HIRUSHAN\jdk\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.2.2\lib\idea
1 12 23 34 45 56 78 79 90
Process finished with exit code 0

```

FIGURE 07 – QUICK SORT OUTPUT