GUJARAT TECHNOLOGICAL UNIVERSITY

Government Engineering College, Bhavnagar

Subject: **Analysis and Design of Algorithm**

B.E. C.E. Semester-5th

(Computer Branch)

Submitted By:

Name: Vankani Arjun BakulBhai

Enrollment: 180210107060

Prof. Ashish Nimavat
Prof. Sahihtabanu Macchar
(Faculty Guide)

Prof. KARSHAN KANDORIYA

(Head of the Department)

Academic Year (2020-21)

Index

Analysis and Design of Algorithm

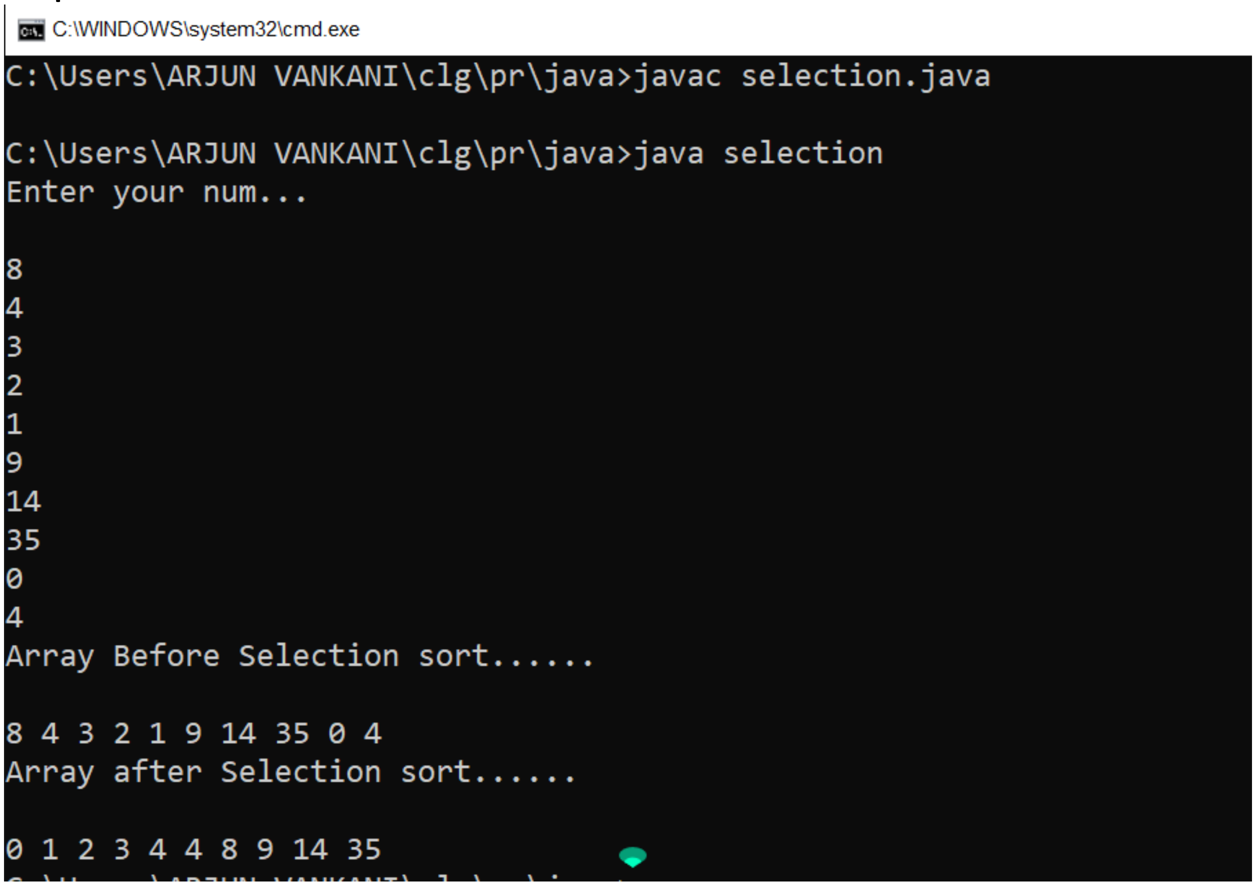
- ✓ **Q-1) Implementation of Selection sort**

Code:

```
import java.util.Scanner;

public class selection
{
          static void selectionsort(int[] arr){

          int n = arr.length;
          int temp = 0;

          for (int i = 0; i < n-1; i++)
      {

         int key= i;
         for (int j = i+1; j < n; j++)
            if (arr[j] < arr[key])
              {
                              key = j;
                             }
         temp = arr[key];
         arr[key] = arr[i];
         arr[i] = temp;
      }

          }


public static void main(String args[]){

                              Scanner n = new Scanner(System.in);
                              int arr[]=new int[10];
                              System.out.println("Enter your num...\n");
                              for(int i=0;i<arr.length;i++)
                                        {
                                                  arr[i]= n.nextInt();
                                        }
                              System.out.println("Array Before Selection sort......\n");
                              for(int i=0;i<arr.length;i++)
```

```
                        {
                                System.out.print(arr[i]+" ");
                        }
                System.out.println();
                selectionsort(arr);
                System.out.println("Array after Selection sort......\n");
                for(int i=0;i<arr.length;i++)
                        {
                        System.out.print(arr[i]+" ");
                        }
        }
}
```

Output:



```
C:\WINDOWS\system32\cmd.exe
C:\Users\ARJUN VANKANI\clg\pr\java>javac selection.java

C:\Users\ARJUN VANKANI\clg\pr\java>java selection
Enter your num...

8
4
3
2
1
9
14
35
0
4
Array Before Selection sort......

8 4 3 2 1 9 14 35 0 4
Array after Selection sort......

0 1 2 3 4 4 8 9 14 35
```

- The selection sort algorithm sorts an array by repeatedly finding the minimum element Here we used as key element.
- **Complexity**

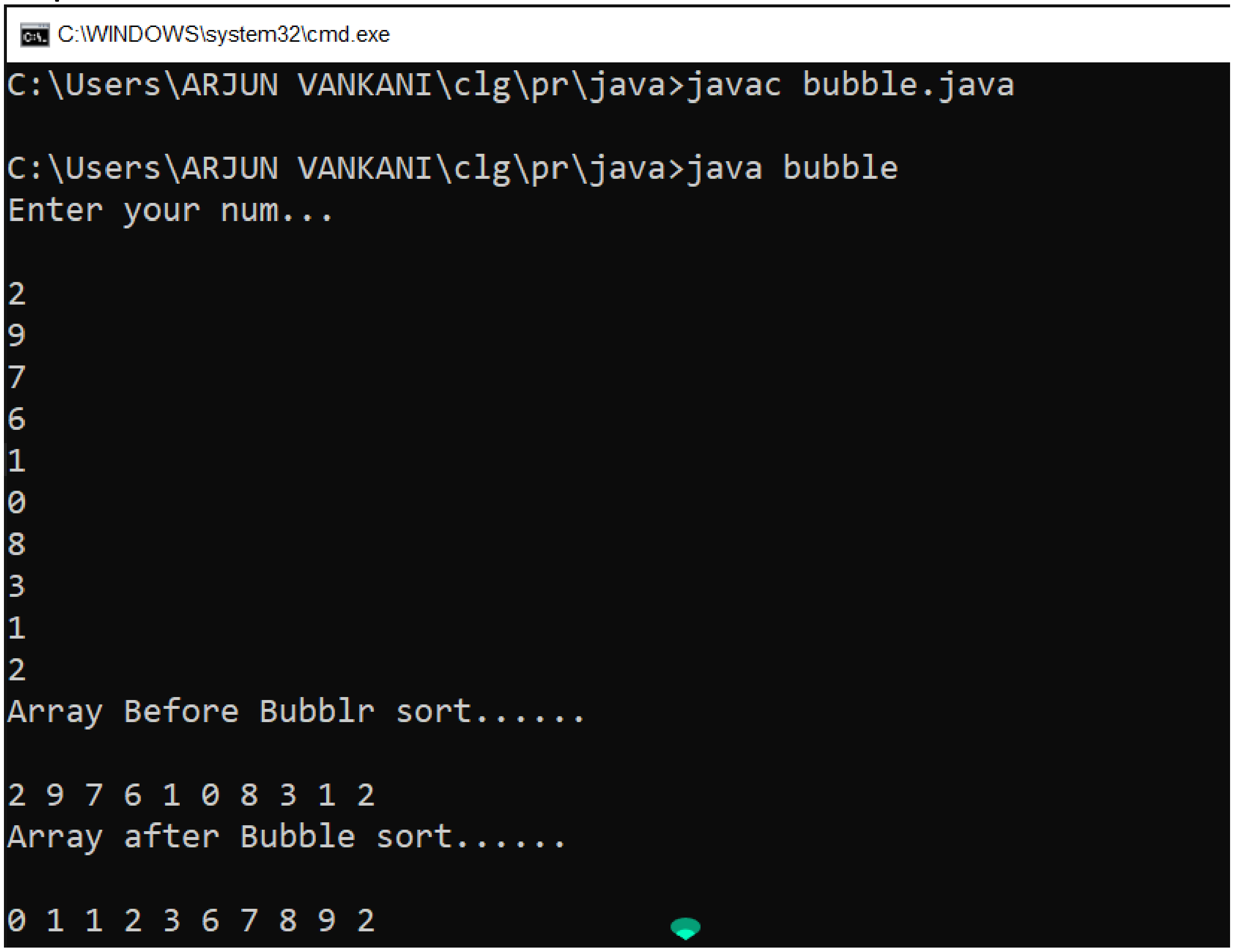
  Best Case: O(n^2)
  Worst Case: O(n^2)

✓ **Q-2) Implementation of Bubble sort**

Code:

```
import java.util.Scanner;

public class bubble
{
        static void bubblesort(int[] arr)
        {
        int n = arr.length;
        int temp= 0;
                for(int i=0;i<n;i++)
                {
                        for(int j=1;j<n-1;j++)
                        {
                                if(arr[j-1]>arr[j])
                                {
                                        temp=arr[j-1];
                                        arr[j-1]=arr[j];
                                        arr[j]=temp;
                                }
                        }
                }
        }
public static void main(String args[]){

                        Scanner n = new Scanner(System.in);
                        int arr[]=new int[10];
                        System.out.println("Enter your num...\n");
                        for(int i=0;i<arr.length;i++)
                        {
                                arr[i]= n.nextInt();
                        }
        System.out.println("Array Before Bubblr sort......\n");
                for(int i=0;i<arr.length;i++)
                {
                        System.out.print(arr[i]+" ");
                }
                System.out.println();
                bubblesort(arr);
                System.out.println("Array after Bubble sort......\n");
                for(int i=0;i<arr.length;i++)
                {
                System.out.print(arr[i]+" ");
                }
        }
}
```

Output:



```
C:\WINDOWS\system32\cmd.exe
C:\Users\ARJUN VANKANI\clg\pr\java>javac bubble.java

C:\Users\ARJUN VANKANI\clg\pr\java>java bubble
Enter your num...

2
9
7
6
1
0
8
3
1
2
Array Before Bubblr sort......

2 9 7 6 1 0 8 3 1 2
Array after Bubble sort......

0 1 1 2 3 6 7 8 9 2
```

- Bubble sort starts with very first two elements, comparing them to check which one is greater
- **Complexity**

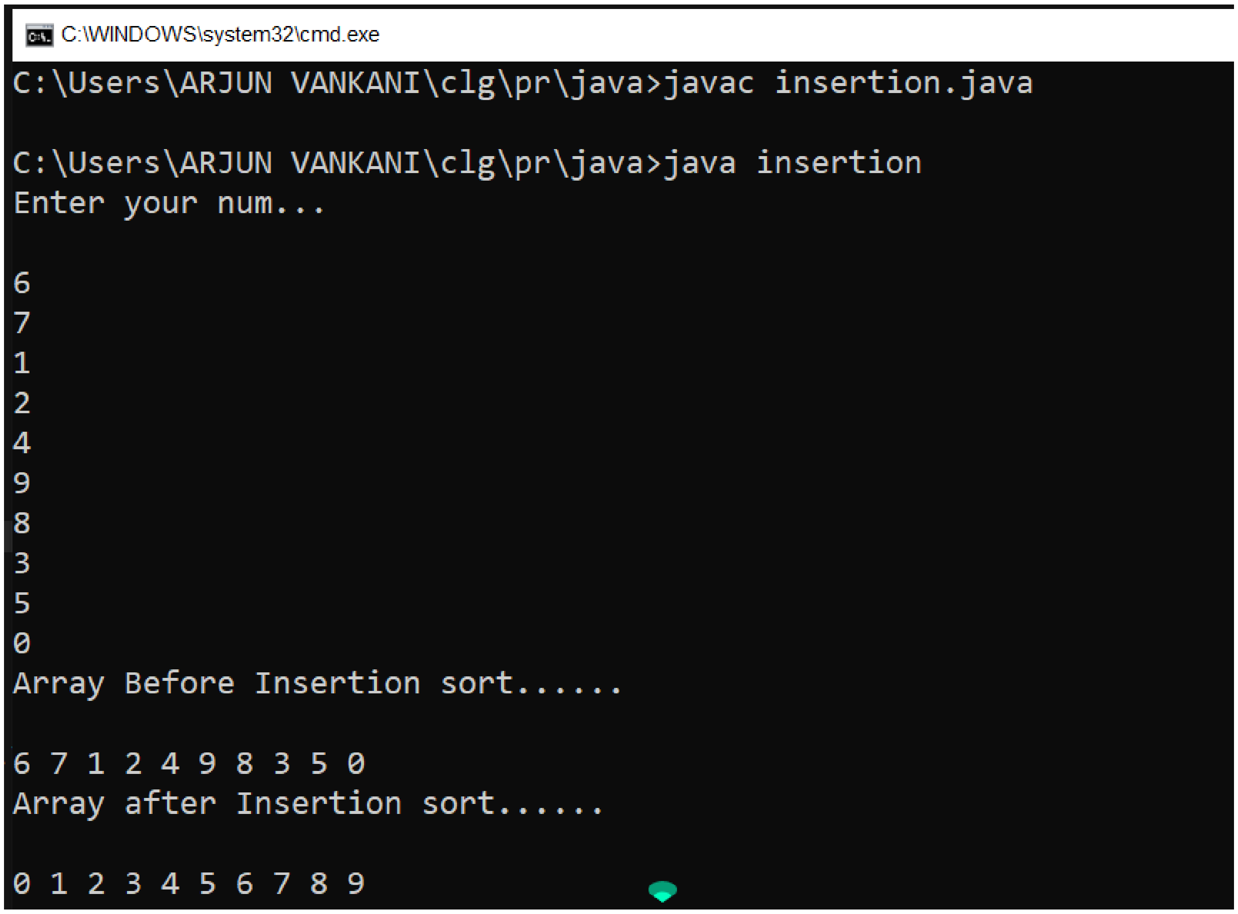
  Best Case: O(n)
  Worst Case: O(n^2)

- ✓ **Q-3) Implementation of Insertion sort**

Code:

```
import java.util.Scanner;

public class insertion

{

        static void insertionsort(int[] arr)

        {

                int n = arr.length;

                for (int i = 1; i < n; i++) {

        int key = arr[i];

        int j = i-1;

        while ( (j > -1) && ( arr [j] > key ) ) {

           arr [j+1] = arr [j];

           j--;

        }

        arr[j+1] = key;

      }

        }

public static void main(String args[]){

                        Scanner n = new Scanner(System.in);

                        int arr[]=new int[10];

                        System.out.println("Enter your num...\n");

                        for(int i=0;i<arr.length;i++)

                        {

                                arr[i]= n.nextInt();
                }
            System.out.println("Array Before Insertion sort......\n");
            for(int i=0;i<arr.length;i++)
            {
                System.out.print(arr[i]+" ");
            }
            System.out.println();
            insertionsort(arr);
            System.out.println("Array after Insertion sort......\n");
            for(int i=0;i<arr.length;i++)
            {
            System.out.print(arr[i]+" ");
            }
        }
    }
```

Output:



C:\WINDOWS\system32\cmd.exe

```
C:\Users\ARJUN VANKANI\clg\pr\java>javac insertion.java

C:\Users\ARJUN VANKANI\clg\pr\java>java insertion
Enter your num...

6
7
1
2
4
9
8
3
5
0
Array Before Insertion sort......

6 7 1 2 4 9 8 3 5 0
Array after Insertion sort......

0 1 2 3 4 5 6 7 8 9
```

- Insertion sort is a simple sorting algorithm that works similar to the way you sort playing cards in your hands
- Values from the unsorted part are picked and placed at the correct position in the sorted part.
- **Complexity**

  Best Case: O(n)
  Worst Case: O(n^2)

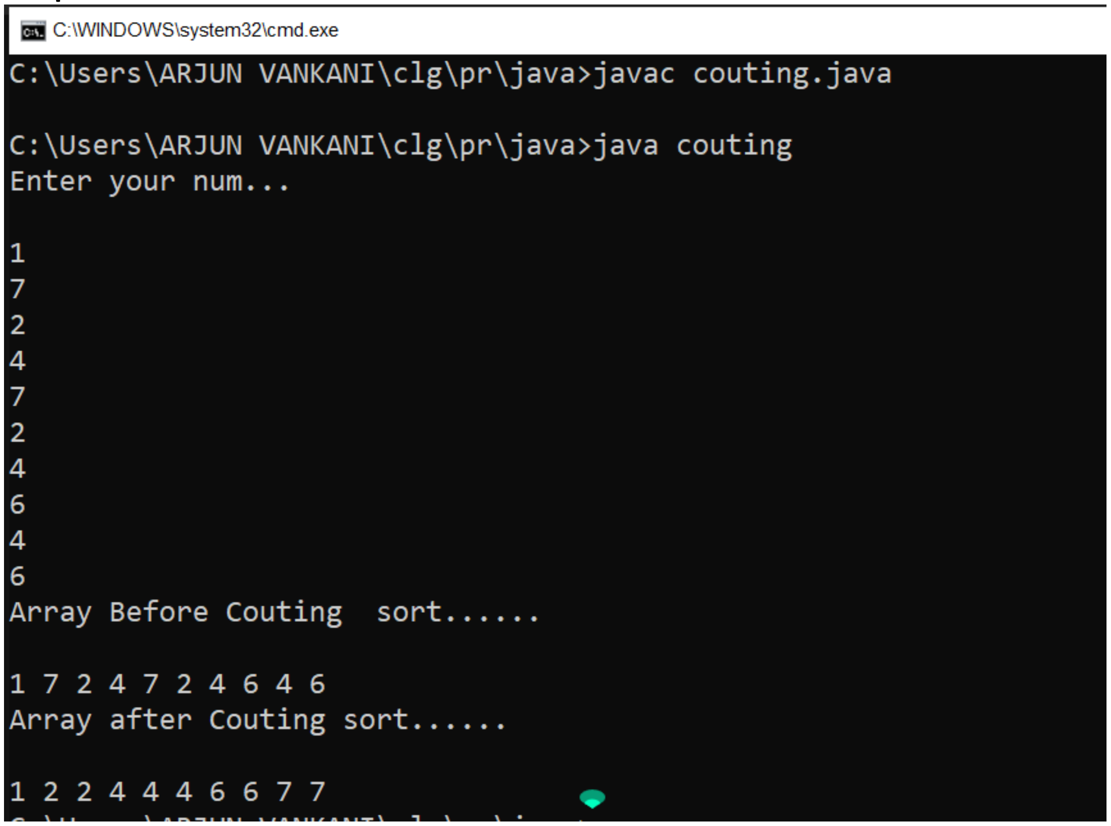
✓ **Q-4) Implementation of Counting sort**

Code:

```
import java.util.Scanner;

public class couting
{
	static void coutingsort(int[] arr)
	{
	int n = arr.length;
	int temp[] = new int[50];

	int count[] = new int[50];

	for(int i = 0; i<10; ++i)
		count[i] = 0;

	for(int i=0; i<n; ++i)
		++count[arr[i]];

	for(int i=1; i<50; ++i)
		count[i] += count[i-1];

	for(int i = n-1; i>=0; i--)
	{
		temp[count[arr[i]]-1] = arr[i];
		--count[arr[i]];
	}

	for(int i=0; i<n; ++i)
		arr[i] = temp[i];

	}
public static void main(String args[]){

				Scanner n = new Scanner(System.in);
				int arr[]=new int[10];
				System.out.println("Enter your num...\n");
				for(int i=0;i<arr.length;i++)
				{
					arr[i]= n.nextInt();
				}
				System.out.println("Array Before Couting  sort......\n");
				for(int i=0;i<arr.length;i++)
				{
```

```
                        System.out.print(arr[i]+" ");
                }
                System.out.println();
                coutingsort(arr);
                System.out.println("Array after Couting sort......\n");
                for(int i=0;i<arr.length;i++)
                {
                System.out.print(arr[i]+" ");
                }
        }
}
```

Output:



```
C:\WINDOWS\system32\cmd.exe
C:\Users\ARJUN VANKANI\clg\pr\java>javac couting.java

C:\Users\ARJUN VANKANI\clg\pr\java>java couting
Enter your num...

1
7
2
4
7
2
4
6
4
6
Array Before Couting  sort......

1 7 2 4 7 2 4 6 4 6
Array after Couting sort......

1 2 2 4 4 4 6 6 7 7
```

- This is a sorting technique based on keys between a specific range. It works by counting the number of objects having distinct key values and kind of hashing.
- **Complexity**

  Best Case: O(n + k)
  Worst Case: O(n + k)

- ✓ **Q-5) Implementation of Merge sort**

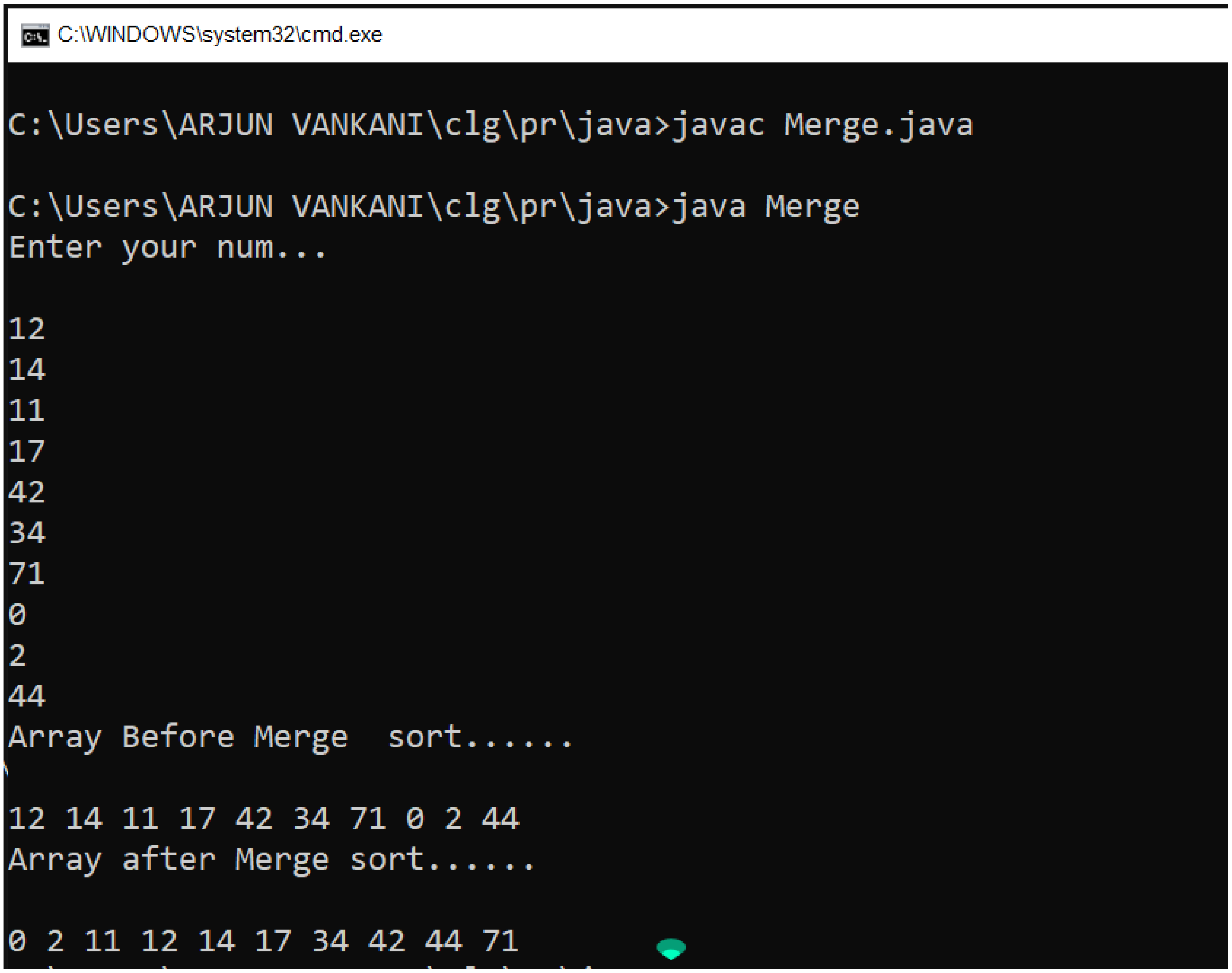
Code:

```
import java.util.Scanner;


public class Merge

{

        static void mergesort(int[] a,int b,int e)

        {

                if(b<e)

                {

                        int m = (b+e)/2;

                        mergesort(a,b,m);

                        mergesort(a,m+1,e);

                        merge(a,b,e,m);

                }


        }

        static void merge(int[] a,int b,int e,int m)

        {

                int i=b,j=m+1;

                int t[] = new int[50];

                int k=0;


                while(i<=m && j<=e)

                {
```

```
                if(a[i] <= a[j])
                {
                        t[k++] = a[i++];
                }
                else
                {
                        t[k++] = a[j++];
                }

        }
        while(i<=m)
        {
                t[k++] = a[i++];
        }
        while(j<=e)
        {
                t[k++] = a[j++];
        }
        for(i=b;i<=e;i++)
        {
                a[i] = t[i-b];
        }

}
```

```
public static void main(String args[]){

                        Scanner n = new Scanner(System.in);
                        int arr[]=new int[10];
                        System.out.println("Enter your num...\n");
                        for(int i=0;i<arr.length;i++)
                        {
                                arr[i]= n.nextInt();
                        }
                        System.out.println("Array Before Merge  sort......\n");
                        for(int i=0;i<arr.length;i++)
                        {
                                System.out.print(arr[i]+" ");
                        }
                        System.out.println();
                        mergesort(arr,0,arr.length-1);
                        System.out.println("Array after Merge sort......\n");
                        for(int i=0;i<arr.length;i++)
                        {
                        System.out.print(arr[i]+" ");
                        }
        }
}
```

Output:



C:\WINDOWS\system32\cmd.exe

```
C:\Users\ARJUN VANKANI\clg\pr\java>javac Merge.java

C:\Users\ARJUN VANKANI\clg\pr\java>java Merge
Enter your num...

12
14
11
17
42
34
71
0
2
44
Array Before Merge  sort......

12 14 11 17 42 34 71 0 2 44
Array after Merge sort......

0 2 11 12 14 17 34 42 44 71
```

- Merge Sort is a Divide and Conquer algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves.
- Three steps for Divide & Conquer 1) Divide 2) Recursion 3) Conquer.
- **Complexity**

  Best Case: O(n log(n))
  Worst Case: O(n log(n))

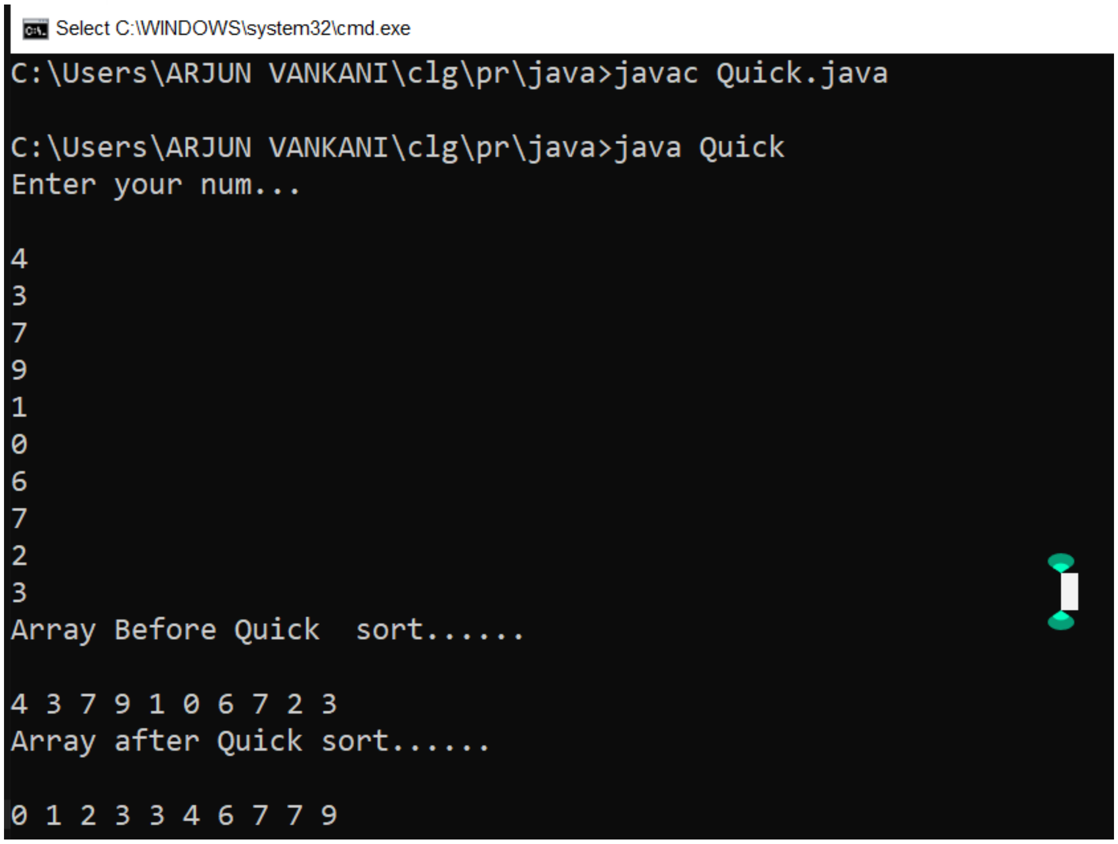
- ✓ **Q-6) Implementation of Quick sort**

Code:

```
import java.util.Scanner;
public class Quick
{
	static int part(int[] a,int b,int e)
	{
		int i=b,j=e,p=b,t;
		while(i<j)
		{
			while(a[p]>=a[i] && i<=e)
			{
				i++;
			}
			while(a[p]<a[j] && j>=b)
			{
				j--;
			}
			if(i<j)
			{
				t = a[i];
				a[i] = a[j];
				a[j] = t;

			}
			else
			{
				t = a[p];
				a[p] = a[j];
				a[j] = t;

			}

		}
		return j;

	}
	static void quicksort(int[] a,int b,int e)
	{

		if(b<e)
		{
			int p = part(a,b,e);
			quicksort(a,b,p-1);
```

```
                    quicksort(a,p+1,e);
            }

        }
    public static void main(String args[]){

                    Scanner n = new Scanner(System.in);
                    int arr[]=new int[10];
                    int num = arr.length;
                    System.out.println("Enter your num...\n");
                    for(int i=0;i<num;i++)
                    {
                        arr[i]= n.nextInt();
                    }
                    System.out.println("Array Before Quick  sort......\n");
                    for(int i=0;i<num;i++)
                    {
                        System.out.print(arr[i]+" ");
                    }
                    System.out.println();

                    quicksort(arr,0,num-1);

                    System.out.println("Array after Quick sort......\n");
                    for(int i=0;i<num;i++)
                    {
                    System.out.print(arr[i]+" ");
                    }
        }
    }
```

Output:



```
C:\Users\ARJUN VANKANI\clg\pr\java>javac Quick.java

C:\Users\ARJUN VANKANI\clg\pr\java>java Quick
Enter your num...

4
3
7
9
1
0
6
7
2
3
Array Before Quick  sort......

4 3 7 9 1 0 6 7 2 3
Array after Quick sort......

0 1 2 3 3 4 6 7 7 9
```

- Quicksort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot. There are many different versions of quicksort that pick pivot in different ways like as first element or Last element.
- **Complexity**

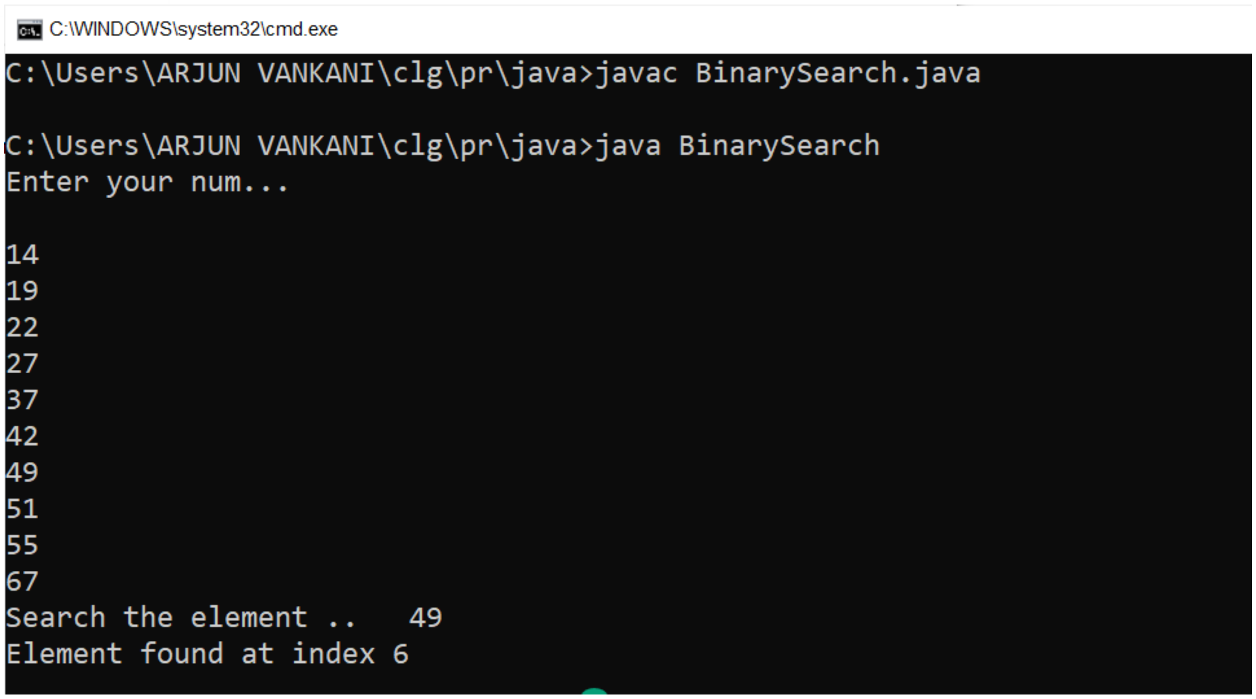
Best Case: O(n log(n))
Worst Case: O(n ^2)

- ✓ **Q-7) Implementation of Binary Search algorithm**

Code:

```
import java.util.Scanner;


public class BinarySearch {
    int binarySearch(int arr[], int l, int h, int key)
    {
        if (h >= l) {
            int mid = l + (h - l) / 2;



            if (arr[mid] == key)
                return mid;


            if (arr[mid] > key)
                return binarySearch(arr, l, mid - 1, key);


            return binarySearch(arr, mid + 1, h, key);
        }
        return -1;
    }
    public static void main(String args[])
    {


        Scanner n = new Scanner(System.in);
                    int arr[]=new int[10];
                    System.out.println("Enter your num...\n");
                    for(int i=0;i<arr.length;i++)
                            {
                                arr[i]= n.nextInt();
                        }

                Scanner n1 = new Scanner(System.in);
                System.out.print("Search the element ..\t");
                int key = n1.nextInt();

                BinarySearch bs = new BinarySearch();

        int result = bs.binarySearch(arr, 0,arr.length - 1, key);
        if (result == -1)
          System.out.println("Element not present");
        else
          System.out.println("Element found at index " + result);
    }
}
```

Output:



C:\WINDOWS\system32\cmd.exe

```
C:\Users\ARJUN VANKANI\clg\pr\java>javac BinarySearch.java

C:\Users\ARJUN VANKANI\clg\pr\java>java BinarySearch
Enter your num...

14
19
22
27
37
42
49
51
55
67
Search the element ..   49
Element found at index 6
```

- We can use binary search to reduce the number of comparisons in normal insertion sort. Binary Insertion Sort uses binary search to find the proper location to insert the selected item at each iteration.
- Some times in binary search give us the ordered list after just we want to find indexed element.
- **Complexity**

  Best Case: O(1)
  Worst Case: O(log(n))

✓ **Q-8) Implementation of Greedy Knapsack Problem**

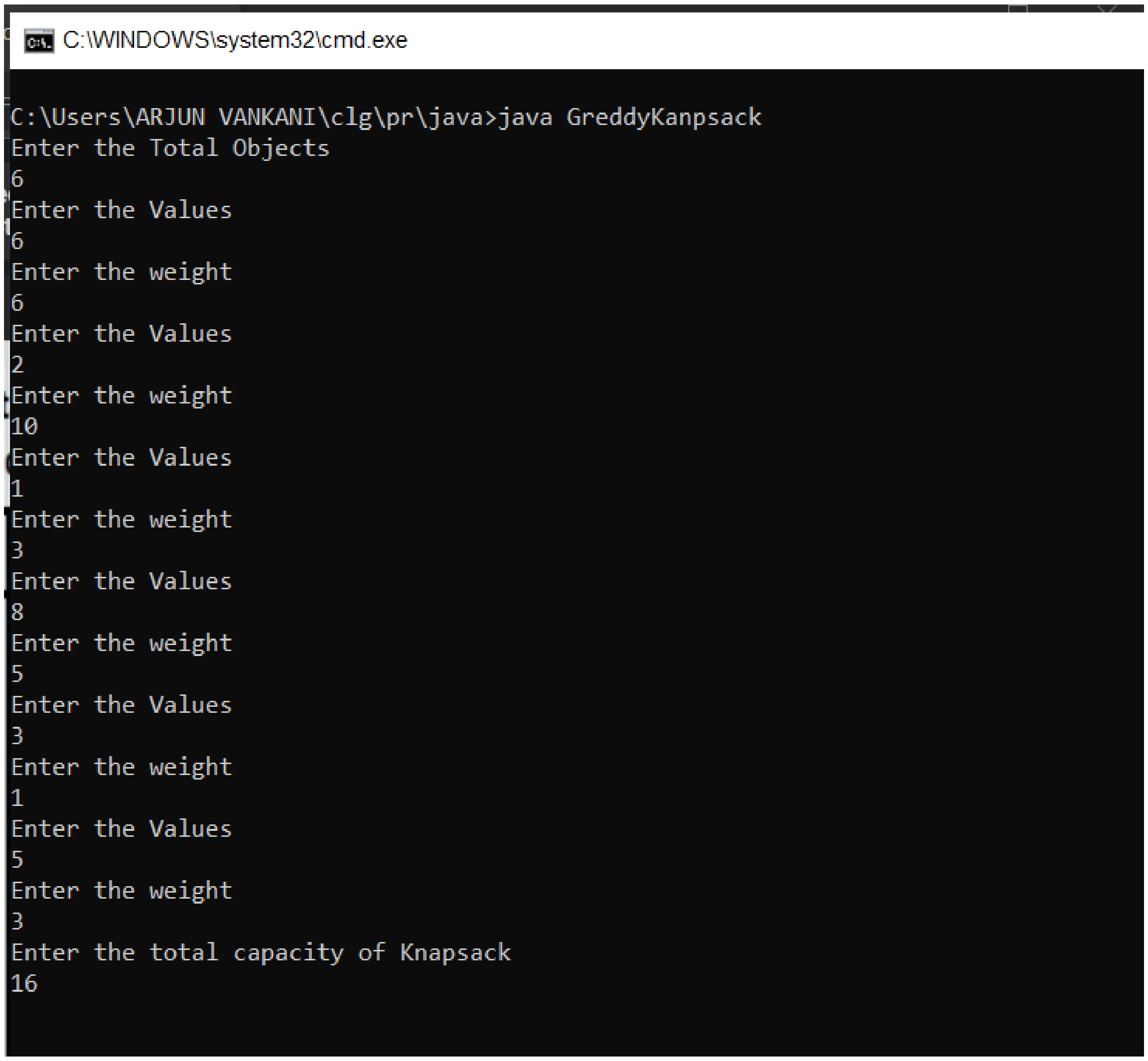
Code:

```
import java.util.Scanner;


class GreddyKanpsack
    {
            public static void main(String[] args)
            {
                    Scanner s = new Scanner(System.in);
                    int n,m;

                    System.out.println("Enter the Total Objects");
                    n = s.nextInt();

                    int weight[] = new int[n];
                    int value[] = new int[n];
                    for(int i=0;i<n;i++)
                    {
                            System.out.println("Enter the Values");
                            value[i]=s.nextInt();

                            System.out.println("Enter the weight");
                            weight[i]=s.nextInt();
                    }

                    System.out.println("Enter the total capacity of Knapsack");
                    m = s.nextInt();

                    float v[]=new float[n];    //  v  = p/w
```

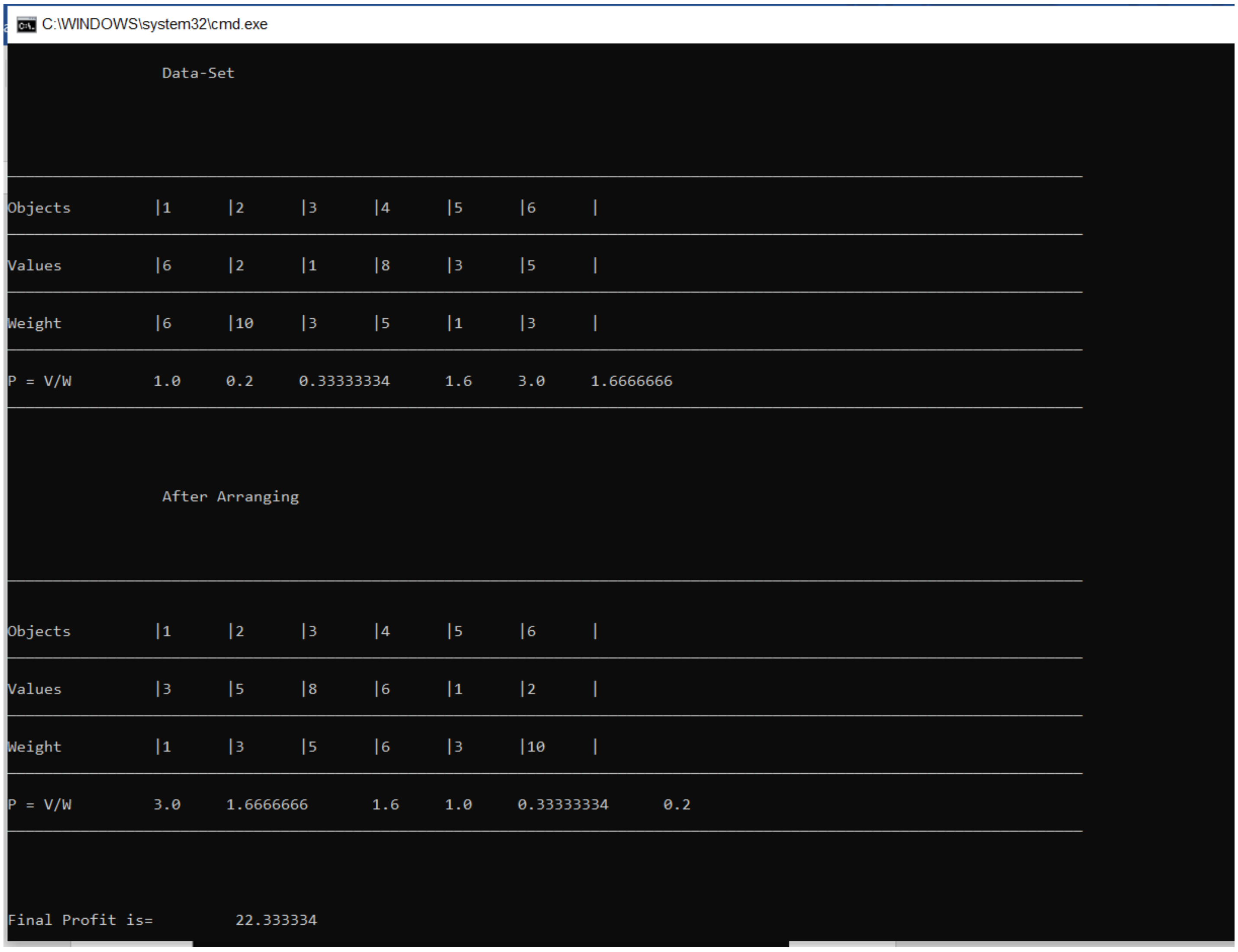
```
                for(int i=0;i<n;i++)
                {
                        v[i]=(float)value[i]/(float)weight[i];
                }

        System.out.println("\n\n");

        System.out.println("\t \t Data-Set \t \t");

        System.out.println("\n\n");

    System.out.println("\n________________________________________________________________________
______________________________________________________________________\n");
        System.out.print("Objects \t|");
        for(int i=1;i<=n;i++)
        {
                System.out.print(i+"\t|");
        }

    System.out.println("\n________________________________________________________________________
______________________________________________________________________\n");
        System.out.print("Values \t\t|");

        for(int i=0;i<n;i++)
        {
                System.out.print(value[i]+"\t|");
        }

    System.out.println("\n________________________________________________________________________
______________________________________________________________________\n");
```

```
        System.out.print("Weight \t\t|");

        for(int i=0;i<n;i++)
        {
                System.out.print(weight[i]+"\t|");
        }


    System.out.println("\n____________________________________________________________________
____________________________________________________________________\n");
        System.out.print("P = V/W \t");
        for(int i=0;i<n;i++)
        {
                System.out.print(v[i]+"\t");
        }

    System.out.println("\n____________________________________________________________________
____________________________________________________________________\n");
        for(int i=0;i<n-1;i++)
        {
                for(int j=i+1;j<n;j++)
                {
                        if(v[i]<v[j])
                        {
                        float temp=v[j];
                        v[j]=v[i];
                        v[i]=temp;

                        int temp1=value[j];
                        value[j]=value[i];
                        value[i]=temp1;


                        int temp2=weight[j];

                        weight[j]=weight[i];

                        weight[i]=temp2;

                        }

                    }

                }

    System.out.println("\n\n");

    System.out.println("\t\t After Arranging \t\t");

    System.out.print("\n\n");

    System.out.println("\n__________________________________________________________________________________________________________________________\n");

    System.out.print("\nObjects \t|");

    for(int i=1;i<=n;i++)

    {

            System.out.print(i+"\t|");

    }

    System.out.println("\n__________________________________________________________________________________________________________________________\n");

    System.out.print("Values \t\t|");


    for(int i=0;i<n;i++)

    {

            System.out.print(value[i]+"\t|");

    }

    System.out.println("\n__________________________________________________________________________________________________________________________\n");

    System.out.print("Weight \t\t|");

    for(int i=0;i<n;i++)
    {
            System.out.print(weight[i]+"\t|");
    }
    System.out.println("\n________________________________________________________________
______________________________________________________________\n");
    System.out.print("P = V/W \t");
    for(int i=0;i<n;i++)
    {
            System.out.print(v[i]+"\t");
    }
    System.out.println("\n________________________________________________________________
______________________________________________________________\n");
    int k=0;
    float sum=0;

    System.out.println();
    while(m>0)
    {
            if(weight[k]<m)
            {
                    sum+=1*value[k];
                    m=m-weight[k];
            }
            else
            {
                    float x4=m*value[k];
                    float x5=weight[k];
                    float x6=x4/x5;
                    sum=sum+x6;
                    m=0;
            }
    k++;
    }
    System.out.println("\n\nFinal Profit is= \t "+sum);
    }
}
```

Output:



```
C:\WINDOWS\system32\cmd.exe

C:\Users\ARJUN VANKANI\clg\pr\java>java GreddyKanpsack
Enter the Total Objects
6
Enter the Values
6
Enter the weight
6
Enter the Values
2
Enter the weight
10
Enter the Values
1
Enter the weight
3
Enter the Values
8
Enter the weight
5
Enter the Values
3
Enter the weight
1
Enter the Values
5
Enter the weight
3
Enter the total capacity of Knapsack
16
```



```
               Data-Set



-------------------------------------------------------------------------------------------------------
Objects       |1     |2     |3     |4     |5     |6     |
-------------------------------------------------------------------------------------------------------
Values        |6     |2     |1     |8     |3     |5     |
-------------------------------------------------------------------------------------------------------
Weight        |6     |10    |3     |5     |1     |3     |
-------------------------------------------------------------------------------------------------------
P = V/W       1.0    0.2    0.33333334    1.6    3.0    1.6666666
-------------------------------------------------------------------------------------------------------


               After Arranging



-------------------------------------------------------------------------------------------------------
Objects       |1     |2     |3     |4     |5     |6     |
-------------------------------------------------------------------------------------------------------
Values        |3     |5     |8     |6     |1     |2     |
-------------------------------------------------------------------------------------------------------
Weight        |1     |3     |5     |6     |3     |10    |
-------------------------------------------------------------------------------------------------------
P = V/W       3.0    1.6666666     1.6    1.0    0.33333334     0.2
-------------------------------------------------------------------------------------------------------


Final Profit is=        22.333334
```

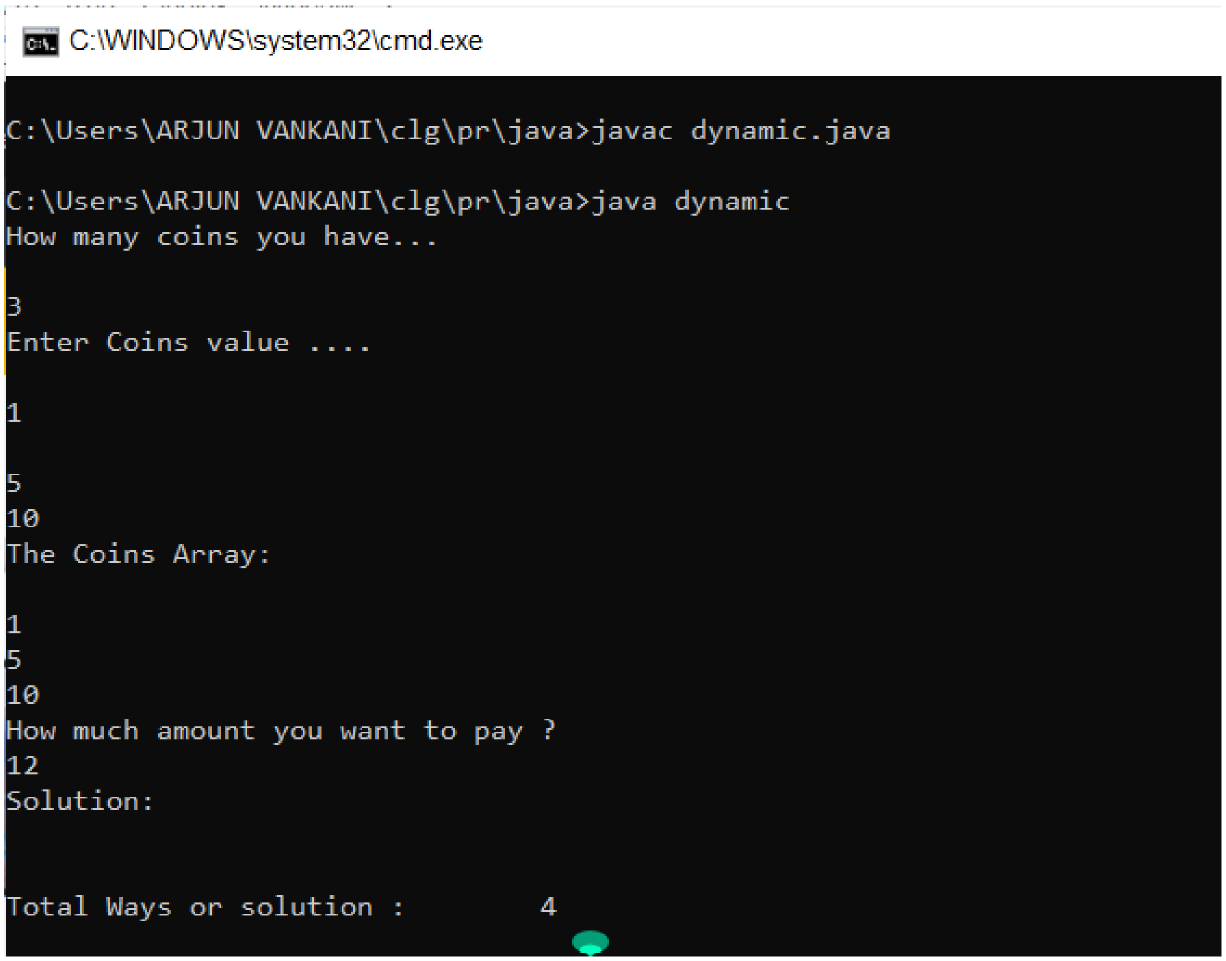
- We have some objects and every object is having some weights, we are provided with a bag that bag is known as Knapsack
  We have to fill the maximum objects in the bag according to their weights and profit so that the profit we get is maximum.
- It runs O(n*W) where w is weight & sometimes O(2^n)

- ✓ **Q-9) Implementation of Making Coin Change Problem**

Code:

```
import java.util.Scanner;

class dynamic {

        static int getNumberOfWays(int N, int[] Coins)
        {

                int[] ways = new int[(int)N + 1];


                ways[0] = 1;


                for (int i = 0; i < Coins.length; i++) {

                        // Make a comparison to each index value
                        // of ways with the coin value.
                        for (int j = 0; j < ways.length; j++) {

                                if (Coins[i] <= j) {

                                        // Update the ways array
                                        ways[j] += ways[(int)(j - Coins[i])];

                                }
                                //System.out.print("\t"+ways[j]);
                }
                //System.out.println();
        }

        // return the value at the Nth position
        // of the ways array.
        System.out.println("\nTotal Ways or solution : \t"+ways[(int)N]);
        return 1;
}


public static void main(String args[])
{
        Scanner n = new Scanner(System.in);
        int num,pay;

        System.out.println("How many coins you have...\n");
        num = n.nextInt();
        int Coins[]=new int[num];
        System.out.println("Enter Coins value ....\n");
        for(int i=0;i<num;i++)
                {
                        Coins[i]= n.nextInt();
                }

        System.out.println("The Coins Array:\n");
        for (int i : Coins)
                System.out.println(i);
                System.out.println("How much amount you want to pay ?\t");
                pay = n.nextInt();
                System.out.println("Solution:\n");
                getNumberOfWays(pay, Coins);


        }
}
```

Output:



```
C:\WINDOWS\system32\cmd.exe

C:\Users\ARJUN VANKANI\clg\pr\java>javac dynamic.java

C:\Users\ARJUN VANKANI\clg\pr\java>java dynamic
How many coins you have...

3
Enter Coins value ....

1

5
10
The Coins Array:

1
5
10
How much amount you want to pay ?
12
Solution:


Total Ways or solution :        4
```

- The change-making problem addresses the question of finding the minimum number of coins of certain denominations that add up to a given amount of money.
- It takes time O(n*W) or sometimes O(m*n)

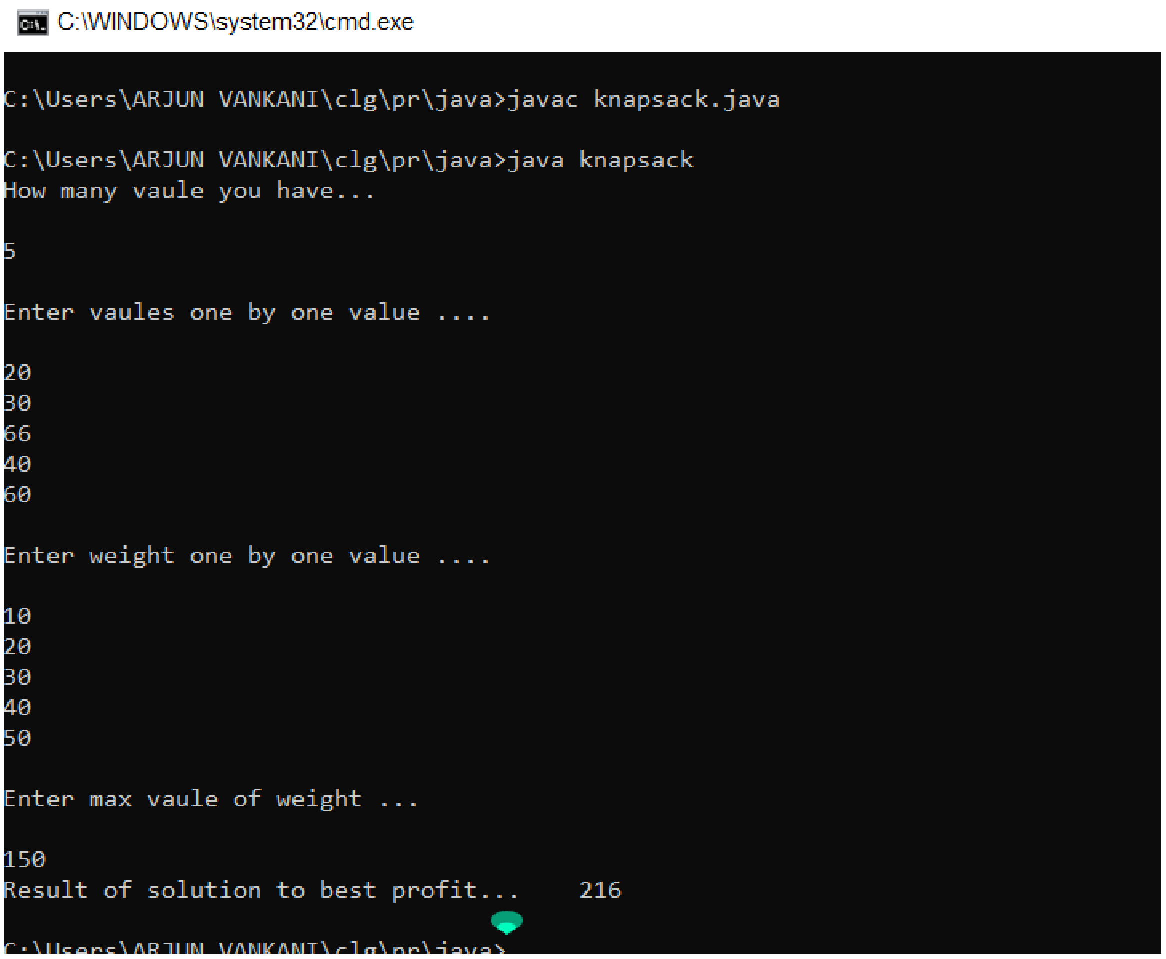
✓ **Q-10) Implementation of Dynamic Knapsack Problem**

Code:

```
import java.util.Scanner;

public class knapsack {

    static int max(int a, int b)

  {

     return (a > b) ? a : b;

  }

    static int knapSack(int W_max, int weight[],int val[], int num)

  {

     if (num == 0 || W_max == 0)

        return 0;

     if (weight[num - 1] > W_max)

        return knapSack(W_max, weight, val, num-1);

     else

        return max(val[num - 1]+knapSack(W_max - weight[num - 1], weight, val, num-1),

           knapSack(W_max, weight, val, num-1));

  }

    public static void main(String[] args) {

            Scanner n = new Scanner(System.in);

            System.out.println("How many vaule you have...\n");

            int num = n.nextInt();


            int val[]=new int[num];

            int weight[]=new int[num];

            System.out.println("\nEnter vaules one by one value ....\n");
```

```
                for(int i=0;i<num;i++)
                        {
                                val[i]= n.nextInt();
                        }
                System.out.println("\nEnter weight one by one value ....\n");
                for(int i=0;i<num;i++)
                        {
                                weight[i]= n.nextInt();
                        }
                System.out.println("\nEnter max vaule of weight ...\n ");
                int W_max = n.nextInt();
        System.out.println("Result of solution to best profit...\t"+knapSack(W_max, weight, val, num));
        }
    }
```

Output:



C:\WINDOWS\system32\cmd.exe

```
C:\Users\ARJUN VANKANI\clg\pr\java>javac knapsack.java

C:\Users\ARJUN VANKANI\clg\pr\java>java knapsack
How many vaule you have...

5

Enter vaules one by one value ....

20
30
66
40
60

Enter weight one by one value ....

10
20
30
40
50

Enter max vaule of weight ...

150
Result of solution to best profit...    216

C:\Users\ARJUN VANKANI\clg\pr\java>
```

- Given weights and values of n items, put these items in a knapsack of capacity W to get the maximum total value in the knapsack. This is different from greedy because here we can't divide the weight.
- You cannot break an item, either pick the complete item or don't pick it 0-1 property. we can't take as fraction.
- It takes O(2^n) time

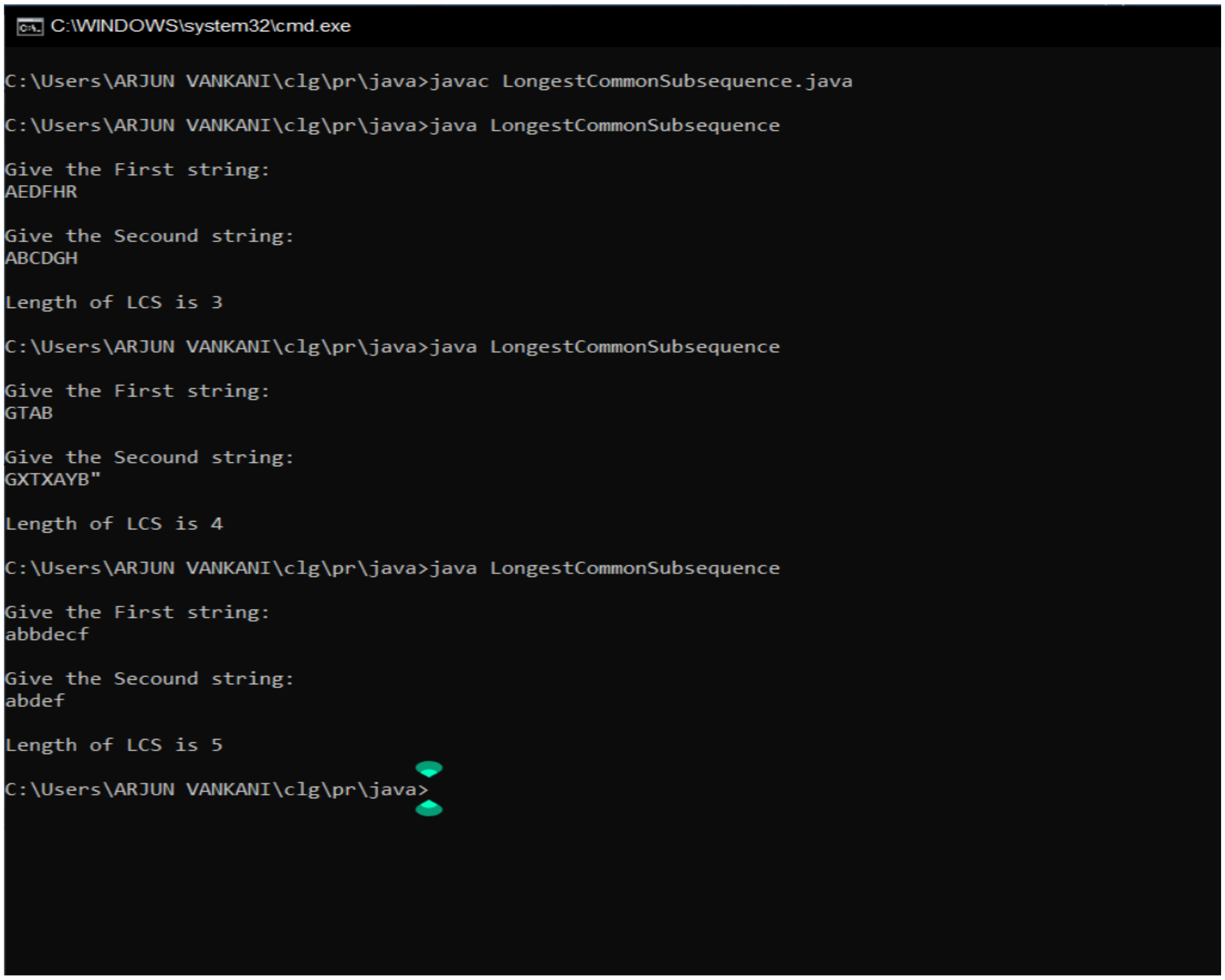
✓ **Q-11) Implementation of Longest Common Subsequence Problem**

Code:

```
import java.util.Scanner;


public class LongestCommonSubsequence
{
  int lcs( char[] X, char[] Y, int m, int n )
  {
    if (m == 0 || n == 0)
      return 0;
    if (X[m-1] == Y[n-1]) {

      return 1 + lcs(X, Y, m-1, n-1);
    }
       else{

      return max(lcs(X, Y, m, n-1), lcs(X, Y, m-1, n));
    }
  }
  int max(int a, int b)
  {
    return (a > b)? a : b;
  }


  public static void main(String[] args)
  {
    LongestCommonSubsequence Lcs = new LongestCommonSubsequence();
       Scanner num = new Scanner(System.in);
       System.out.println("\nGive the First string:\t");
```

```
        String s1 = num.nextLine();

        System.out.println("\nGive the Secound string:\t");

        String s2 = num.nextLine();

      char[] X=s1.toCharArray();

      char[] Y=s2.toCharArray();

      int m = X.length;

      int n = Y.length;

      System.out.println("\nLength of LCS is" + " "+

                              Lcs.lcs( X, Y, m, n ));

    }

  }
```

Output:



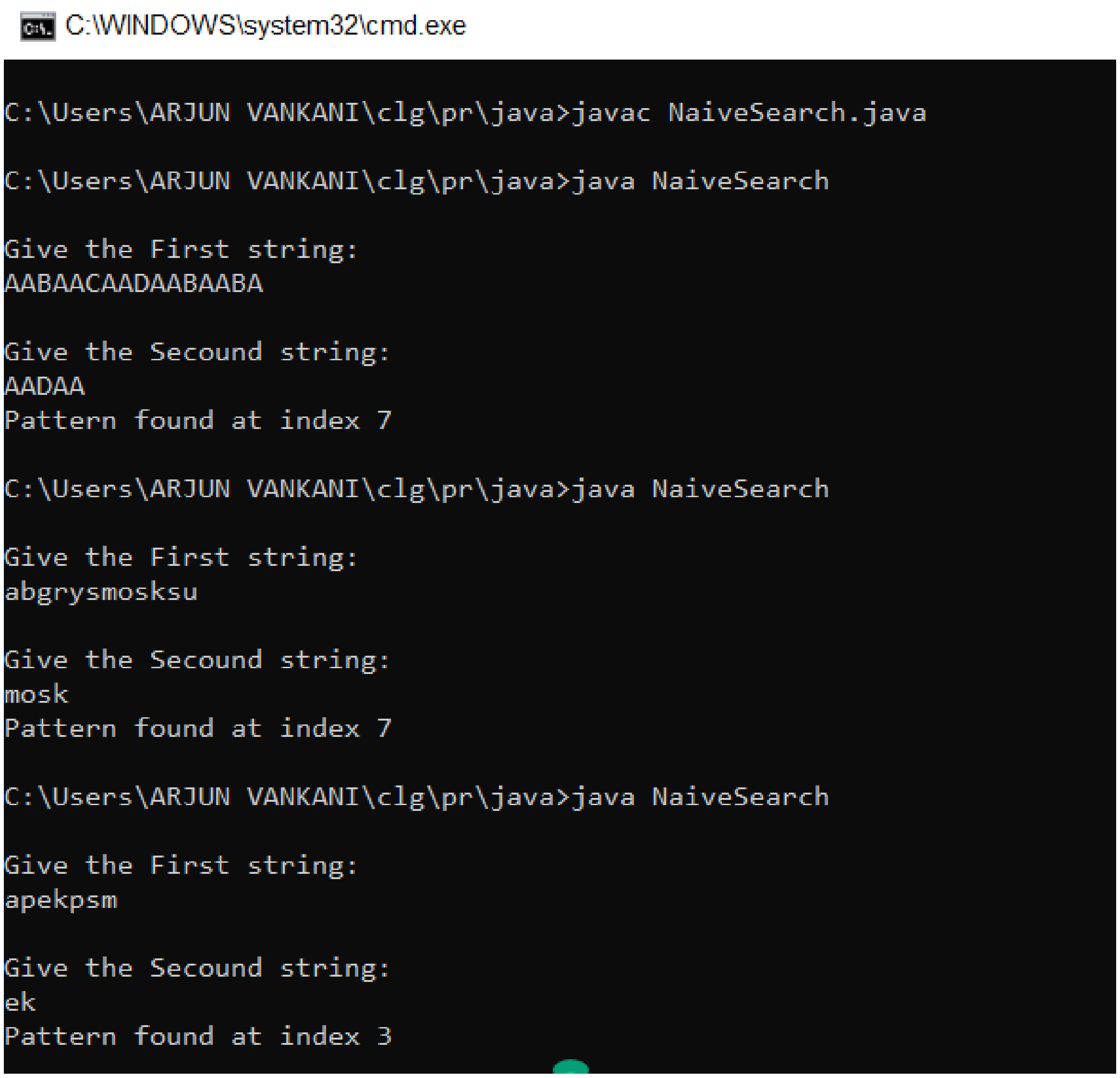
```
C:\WINDOWS\system32\cmd.exe

C:\Users\ARJUN VANKANI\clg\pr\java>javac LongestCommonSubsequence.java

C:\Users\ARJUN VANKANI\clg\pr\java>java LongestCommonSubsequence

Give the First string:
AEDFHR

Give the Secound string:
ABCDGH

Length of LCS is 3

C:\Users\ARJUN VANKANI\clg\pr\java>java LongestCommonSubsequence

Give the First string:
GTAB

Give the Secound string:
GXTXAYB"

Length of LCS is 4

C:\Users\ARJUN VANKANI\clg\pr\java>java LongestCommonSubsequence

Give the First string:
abbdecf

Give the Secound string:
abdef

Length of LCS is 5

C:\Users\ARJUN VANKANI\clg\pr\java>
```

- The longest common subsequence (LCS) problem is the problem of finding the longest subsequence common to all sequences in a set of sequences often just two sequences. It differs from the longest common substring
- It takes O(m*n) times.

✓ **Q-12) Implementation of Naïve String Search Algorithm**
Code:

```
import java.util.Scanner;

public class NaiveSearch {

        public static void search(String s1, String s2)
        {
                int M = s2.length();
                int N = s1.length();

                for (int i = 0; i <= N - M; i++) {

                        int j;
                        for (j = 0; j < M; j++)
                                if (s1.charAt(i + j) != s2.charAt(j))
                                        break;
                        if (j == M)
                                System.out.println("Pattern found at index " + (i+1));
                }
        }
        public static void main(String[] args)
        { Scanner num = new Scanner(System.in);

                System.out.println("\nGive the First string:\t");
                String s1 = num.nextLine();
                System.out.println("\nGive the Secound string:\t");
                String s2 = num.nextLine();

                search(s1, s2);
        }
}
```

Output:



C:\WINDOWS\system32\cmd.exe

```
C:\Users\ARJUN VANKANI\clg\pr\java>javac NaiveSearch.java

C:\Users\ARJUN VANKANI\clg\pr\java>java NaiveSearch

Give the First string:
AABAACAADAABAABA

Give the Secound string:
AADAA
Pattern found at index 7

C:\Users\ARJUN VANKANI\clg\pr\java>java NaiveSearch

Give the First string:
abgrysmosksu

Give the Secound string:
mosk
Pattern found at index 7

C:\Users\ARJUN VANKANI\clg\pr\java>java NaiveSearch

Give the First string:
apekpsm

Give the Secound string:
ek
Pattern found at index 3
```

- The naïve string-matching algorithm is essentially the most popular strategy to discovering the positions of stated patterns in a given textual content for numerous causes like no pre-processing requirement, no additional house for operation, and so on.
- Slide the pattern over text one by one and check for a match. If a match is found, then slides by 1 again to check for subsequent matches.
- It takes O(m + n) to O(m*n) time

All the Complexity

Sort	Best Case	Worst Case
Bubble Sort	O(n)	O(n^2)
Insertion Sort	O(n)	O(n^2)
Selection Sort	O(n^2)	O(n^2)
Merge Sort	O(n* log(n))	O(n* log(n))
Quick Sort	O(n* log(n))	O(n^2)
Heap Sort	O(n* log(n))	O(n* log(n))
Shell Sort	O(n* log(n))	O(n* log(n))^2)
Bucket Sort	O(n + k)	O(n^2)
Radix Sort	O(n*k)	O(n*k)
Counting Sort	O(n + k)	O(n + k)