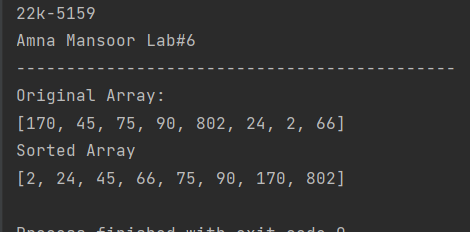
**Lab # 6 (DS)**

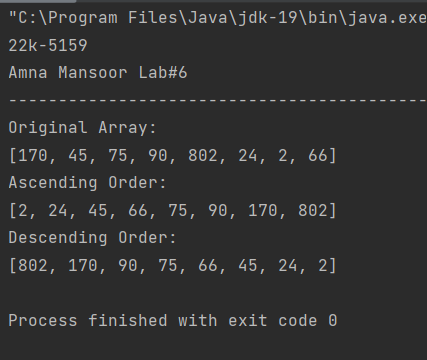
***TASK 1:***  
  
import java.util.Arrays;  
  
public class Task1 {  
 public static void main(String[] args) {  
 System.*out*.println("22k-5159");  
 System.*out*.println("Amna Mansoor Lab#6");  
 System.*out*.println("--------------------------------------------");  
 System.*out*.println("Original Array:");  
 int[] array = {170, 45, 75, 90, 802, 24, 2, 66};  
 System.*out*.println(Arrays.*toString*(array));  
 int n = array.length;  
 *radix*(array, n);   
 *Print*(array);   
 }  
  
 static int maxElement(int[] array, int n) {  
 int max = array[0];  
 for (int i = 1; i < n; i++) {  
 if (array[i] > max) {  
 max = array[i];  
 }  
 }  
 return max;  
 }  
  
 static void countSort(int[] array, int n, int pos) {  
 int[] output = new int[n];  
 int[] count = new int[10];

for (int i = 0; i < n; i++) {  
 count[(array[i] / pos) % 10]++;  
 }  
  
   
 for (int i = 1; i < 10; i++) {  
 count[i] += count[i - 1];  
 }  
  
 for (int i = n - 1; i >= 0; i--) {  
 output[count[(array[i] / pos) % 10] - 1] = array[i];  
 count[(array[i] / pos) % 10]--;  
 }  
  
 for (int i = 0; i < n; i++) {  
 array[i] = output[i];  
 }  
 }  
  
 // Radix Sort function  
 static void radix(int[] array, int n) {  
 int max = *maxElement*(array, n);  
  
   
 for (int pos = 1; max / pos > 0; pos \*= 10) {  
 *countSort*(array, n, pos);  
 }  
 }  
  
 static void Print(int[] array) {  
 System.*out*.println("Sorted Array");  
 System.*out*.println(Arrays.*toString*(array));  
 }  
}

**

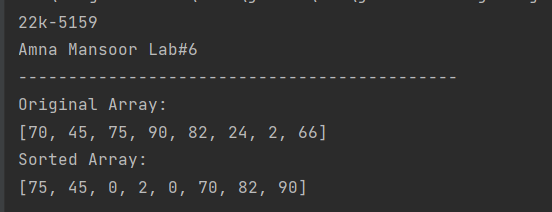
***Task # 2:***

import java.util.Arrays;  
  
public class Task2 {  
 public static void main(String[] args) {  
 System.*out*.println("22k-5159");  
 System.*out*.println("Amna Mansoor Lab#6");  
 System.*out*.println("--------------------------------------------");  
 System.*out*.println("Original Array:");  
 int[] array = {170, 45, 75, 90, 802, 24, 2, 66};  
 System.*out*.println(Arrays.*toString*(array));  
 int n = array.length;  
  
 *radixAscending*(array, n);  
 System.*out*.println("Ascending Order:");  
 *Print*(array);  
  
 *radixDescending*(array, n);  
 System.*out*.println("Descending Order:");  
 *Print*(array);  
 }  
 static int maxElement(int[] array, int n) {  
 int max = array[0];  
 for (int i = 1; i < n; i++) {  
 if (array[i] > max) {  
 max = array[i];  
 }  
 }  
 return max;  
 }  
 static void countSort(int[] array, int n, int pos) {  
 int[] output = new int[n + 1];  
 int[] count = new int[10];  
  
 for (int i = 0; i < n; i++) {  
 count[(array[i] / pos) % 10]++;  
 }  
 for (int i = 1; i < 10; i++) {  
 count[i] += count[i - 1];  
 }  
 for (int i = n - 1; i >= 0; i--) {  
 output[count[(array[i] / pos) % 10] - 1] = array[i];  
 count[(array[i] / pos) % 10]--;  
 }  
 for (int i = 0; i < n; i++) {  
 array[i] = output[i];  
 }  
 }  
 static void radixAscending(int[] array, int n) {  
 int max = *maxElement*(array, n);  
 for (int pos = 1; max / pos > 0; pos \*= 10) {  
 *countSort*(array, n, pos);  
 }  
 }  
  
 static void radixDescending(int[] array, int n) {  
 int max = *maxElement*(array, n);  
 for (int pos = 1; max / pos > 0; pos \*= 10) {  
 *countSort*(array, n, pos);  
 }  
 for (int i = 0; i < n / 2; i++) {  
 int temp = array[i];  
 array[i] = array[n - i - 1];  
 array[n - i - 1] = temp;  
 }  
 }  
 static void Print(int[] array) {  
 System.*out*.println(Arrays.*toString*(array));  
 }  
}

**

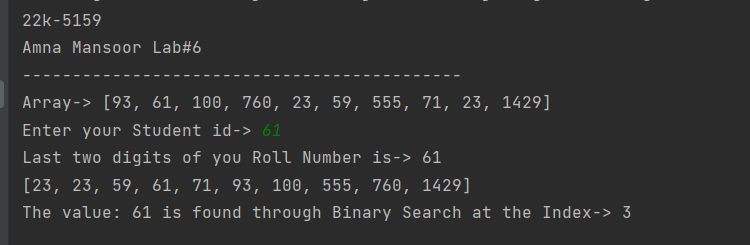
***Task 3:***

import java.util.\*;  
  
public class Task3 {  
 public static void main(String[] args) {  
 System.*out*.println("22k-5159");  
 System.*out*.println("Amna Mansoor Lab#6");  
 System.*out*.println("--------------------------------------------");  
 System.*out*.println("Original Array:");  
 int[] array = {70, 45, 75, 90, 82, 24, 2, 66};  
 System.*out*.println(Arrays.*toString*(array));  
 int n = array.length;  
 *twoWayRadix*(array, n);  
 *print*(array);  
 }  
  
 static int maxElement(int[] array, int n) {  
 int max = array[0];  
 for (int i = 1; i < n; i++) {  
 if (array[i] > max) {  
 max = array[i];  
 }  
 }  
 return max;  
 }  
  
 static void twoWayRadix(int[] array, int n) {  
 int[] odd = new int[n];  
 int[] even = new int[n];  
 int oddCount = 0;  
 int evenCount = 0;  
  
 for (int i = 0; i < n; i++) {  
 if (array[i] % 2 == 1) {  
 odd[oddCount++] = array[i];  
 } else {  
 even[evenCount++] = array[i];  
 }  
 }  
 *radix*(odd, oddCount, true);  
 *radix*(even, evenCount, false);  
  
 int i = 0;  
 for (int j = 0; j < oddCount; j++) {  
 array[i++] = odd[j];  
 }  
 for (int j = 0; j < evenCount; j++) {  
 array[i++] = even[j];  
 }  
 }  
  
 static void countSort(int[] array, int n, int pos, boolean descending) {  
 int[] output = new int[n];  
 int[] count = new int[10];  
  
 for (int i = 0; i < n; i++) {  
 count[(array[i] / pos) % 10]++;  
 }  
  
 if (descending) {  
 for (int i = 8; i >= 0; i--) {  
 count[i] += count[i + 1];  
 }  
 } else {  
 for (int i = 1; i < 10; i++) {  
 count[i] += count[i - 1];  
 }  
 }  
  
 for (int i = n - 1; i >= 0; i--) {  
 int index = count[(array[i] / pos) % 10] - 1;  
 if (index >= 0) {  
 output[index] = array[i];  
 if (descending) {  
 count[(array[i] / pos) % 10]--;  
 } else {  
 count[(array[i] / pos) % 10]++;  
 }  
 }  
 }  
 for (int i = 0; i < n; i++) {  
 array[i] = output[i];  
 }  
 }  
  
 static void radix(int[] array, int n, boolean descending) {  
 int max = *maxElement*(array, n);  
 for (int pos = 1; max / pos > 0; pos \*= 10) {  
 *countSort*(array, n, pos, descending);  
 }  
 }  
  
 static void print(int[] array) {  
 System.*out*.println("Sorted Array:");  
 System.*out*.println(Arrays.*toString*(array));  
 }  
}

**

***Task 4:***

import java.util.Arrays;  
import java.util.Scanner;  
  
public class Task4 {  
 public static void main(String[] args) {  
 System.*out*.println("22k-5159");  
 System.*out*.println("Amna Mansoor Lab#6");  
 System.*out*.println("--------------------------------------------");  
 int[] array = {93, 61, 100, 760, 23, 59, 555, 71, 23, 1429};  
 System.*out*.println("Array-> " + Arrays.*toString*(array));  
 System.*out*.print("Enter your Student id-> ");  
 Scanner sc = new Scanner(System.*in*);  
 int roll = sc.nextInt();  
 int target = roll % 100;  
 System.*out*.println("Last two digits of you Roll Number is-> " + target);  
 int result = *binary*(array, target);  
 if (result == -1) {  
 System.*out*.println("Digits not present");  
 } else {  
 System.*out*.println("The value: " + target + " is found through Binary Search at the Index-> " + result);  
 }  
 }  
 static void sort(int[] array) {  
 int n = array.length;  
 for (int i = 1; i < n; i++) {  
 for (int j = i; j > 0 && array[j] < array[j - 1]; j--) {  
 int temp = array[j];  
 array[j] = array[j - 1];  
 array[j - 1] = temp;  
 }  
 }  
 }  
 static int binary(int[] array, int target) {  
 *sort*(array);  
 System.*out*.println(Arrays.*toString*(array));  
 int start = 0, end = array.length - 1;  
 while (start <= end) {  
 int mid = start + (end - start) / 2;  
 if (array[mid] == target) {  
 return mid;  
 }  
 if (array[mid] < target) {  
 start = mid + 1;  
 }  
 else {  
 end = mid - 1;  
 }  
 }  
 return -1;  
 }  
}

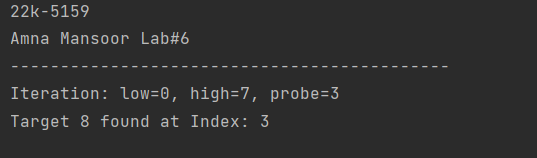
**

***Task 5:***

public class Task5 {  
 public static void main(String[] args) {

System.*out*.println("22k-5159");  
System.*out*.println("Amna Mansoor Lab#6");  
System.*out*.println("--------------------------------------------");

int[] array={2,4,6,8,10,12,14,16};  
 int index=*Interpolation*(array, 8);  
 if(index!=-1){  
 System.*out*.println("Target 8 found at Index: "+index);  
 }  
 else {  
 System.*out*.println("Target 8 not found");  
 }  
 }  
  
 private static int Interpolation(int[] array, int target) {  
 int high=array.length-1;  
 int low=0;  
 while (target>=array[low] && target<=array[high] && low<=high){  
 int probe=low+(((high-low)\*(target-array[low]))  
 /(array[high]-array[low]));  
// System.out.print("Probe-> "+probe);  
 System.*out*.println("Iteration: low=" + low + ", high=" + high + ", probe=" + probe);  
 if(array[probe]==target){  
 return probe;  
 } else if (array[probe]<target) {  
 low=probe+1;  
 }  
 else{  
 high=probe-1;  
 }  
 }  
 return -1;  
 }  
}

**