DS-Lab 6  
22K-4818

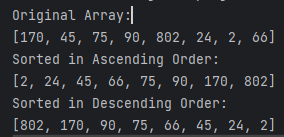
Q1.

import java.util.\*;  
  
class Task1 {  
 static void print(int arr[], int n)  
 {  
 for (int i = 0; i < n; i++)  
 System.*out*.print(arr[i] + " ");  
 }  
 static void radixsort(int arr[], int n) {  
 int m = *getMax*(arr, n);  
 for (int exp = 1; m / exp > 0; exp \*= 10)  
 *countSort*(arr, n, exp);  
 }  
 static int getMax(int arr[], int n)  
 {  
 int mx = arr[0];  
 for (int i = 1; i < n; i++)  
 if (arr[i] > mx)  
 mx = arr[i];  
 return mx;  
 }  
 static void countSort(int arr[], int n, int exp)  
 {  
 int output[] = new int[n];  
 int i;  
 int count[] = new int[10];  
 Arrays.*fill*(count, 0);  
 for (i = 0; i < n; i++)  
 count[(arr[i] / exp) % 10]++;  
 for (i = 1; i < 10; i++)  
 count[i] += count[i - 1];  
 for (i = n - 1; i >= 0; i--) {  
 output[count[(arr[i] / exp) % 10] - 1] = arr[i];  
 count[(arr[i] / exp) % 10]--;  
 }  
 for (i = 0; i < n; i++)  
 arr[i] = output[i];  
 }  
 public static void main(String[] args)  
 {  
 int arr[] = { 170, 45, 75, 90, 802, 24, 2, 66 };  
 int n = arr.length;  
 *radixsort*(arr, n);  
 *print*(arr, n);  
 }  
}



Q2.

import java.util.Arrays;  
  
public class Task2 {  
 public static void main(String[] args) {  
 int arr[] = { 170, 45, 75, 90, 802, 24, 2, 66 };  
 int n = arr.length;  
  
 System.*out*.println("Original Array:");  
 System.*out*.println(Arrays.*toString*(arr));  
  
 *radixSortAsc*(arr,n);  
 System.*out*.println("Sorted in Ascending Order:");  
 System.*out*.println(Arrays.*toString*(arr));  
  
 *radixSortDesc*(arr,n);  
 System.*out*.println("Sorted in Descending Order:");  
 System.*out*.println(Arrays.*toString*(arr));  
 }  
 static int getMax(int arr[], int n)  
 {  
 int mx = arr[0];  
 for (int i = 1; i < n; i++)  
 if (arr[i] > mx)  
 mx = arr[i];  
 return mx;  
 }  
 public static void radixSortAsc(int arr[],int n) {  
 int max = *getMax*(arr,n);  
 for (int exp = 1; max / exp > 0; exp \*= 10) {  
 *countSortAsc*(arr, exp);  
 }  
 }  
 public static void radixSortDesc(int arr[],int n) {  
 int max = *getMax*(arr,n);  
 for (int exp = 1; max / exp > 0; exp \*= 10) {  
 *countSortDesc*(arr, exp);  
 }  
 }  
 public static void countSortAsc(int arr[], int exp) {  
 int n = arr.length;  
 int output[] = new int[n];  
 int count[] = new int[10];  
 Arrays.*fill*(count, 0);  
  
 for (int i = 0; i < n; i++) {  
 count[(arr[i] / exp) % 10]++;  
 }  
  
 for (int i = 1; i < 10; i++) {  
 count[i] += count[i - 1];  
 }  
  
 for (int i = n - 1; i >= 0; i--) {  
 output[count[(arr[i] / exp) % 10] - 1] = arr[i];  
 count[(arr[i] / exp) % 10]--;  
 }  
  
 for (int i = 0; i < n; i++) {  
 arr[i] = output[i];  
 }  
 }  
 public static void countSortDesc(int arr[], int exp) {  
 int n = arr.length;  
 int output[] = new int[n];  
 int count[] = new int[10];  
 Arrays.*fill*(count, 0);  
  
 for (int i = 0; i < n; i++) {  
 count[9 - (arr[i] / exp) % 10]++;  
 }  
  
 for (int i = 1; i < 10; i++) {  
 count[i] += count[i - 1];  
 }  
  
 for (int i = n - 1; i >= 0; i--) {  
 output[count[9 - (arr[i] / exp) % 10] - 1] = arr[i];  
 count[9 - (arr[i] / exp) % 10]--;  
 }  
  
 for (int i = 0; i < n; i++) {  
 arr[i] = output[i];  
 }  
 }  
}



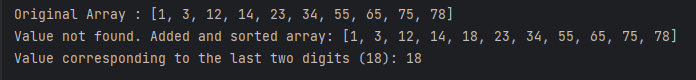
Q3,

import java.util.Arrays;  
  
public class Task3 {  
 public static void main(String[] args) {  
 int arr[] = { 10, 1, 5, 6, 12, 2, 5, 3, 13, 17, 14 };  
  
 System.*out*.println("Before Sorting: " + Arrays.*toString*(arr));  
 *radixSort*(arr);  
 System.*out*.println("After Sorting: " + Arrays.*toString*(arr));  
 }  
 static int getMax(int arr[], int n)  
 {  
 int mx = arr[0];  
 for (int i = 1; i < n; i++)  
 if (arr[i] > mx)  
 mx = arr[i];  
 return mx;  
 }  
  
 public static void radixSort(int arr[]) {  
 int n = arr.length;  
 int oddCount = 0;  
 int evenCount = 0;  
 for (int i = 0; i < n; i++) {  
 if (arr[i] % 2 == 0) {  
 evenCount++;  
 } else {  
 oddCount++;  
 }  
 }  
  
 int[] oddNumbers = new int[oddCount];  
 int[] evenNumbers = new int[evenCount];  
  
 oddCount = 0;  
 evenCount = 0;  
  
 for (int i = 0; i < n; i++) {  
 if (arr[i] % 2 == 0) {  
 evenNumbers[evenCount++] = arr[i];  
 } else {  
 oddNumbers[oddCount++] = arr[i];  
 }  
 }  
 *radixSortDesc*(oddNumbers);  
 *radixSortAsc*(evenNumbers);  
  
 int i = 0;  
 int j = 0;  
 int k = 0;  
  
 while (j < oddCount) {  
 arr[i++] = oddNumbers[j++];  
 }  
  
 while (k < evenCount) {  
 arr[i++] = evenNumbers[k++];  
 }  
 }  
 public static void radixSortAsc(int arr[]) {  
 int n = arr.length;  
 int max = *getMax*(arr,n);  
 for (int exp = 1; max / exp > 0; exp \*= 10) {  
 *countSort*(arr, exp);  
 }  
 }  
  
 public static void radixSortDesc(int arr[]) {  
 int n = arr.length;  
 int max = *getMax*(arr,n);  
 for (int exp = 1; max / exp > 0; exp \*= 10) {  
 *countSortDesc*(arr, exp);  
 }  
 }  
 public static void countSort(int arr[], int exp) {  
 int n = arr.length;  
 int output[] = new int[n];  
 int count[] = new int[10];  
 Arrays.*fill*(count, 0);  
  
 for (int i = 0; i < n; i++) {  
 count[(arr[i] / exp) % 10]++;  
 }  
  
 for (int i = 1; i < 10; i++) {  
 count[i] += count[i - 1];  
 }  
  
 for (int i = n - 1; i >= 0; i--) {  
 output[count[(arr[i] / exp) % 10] - 1] = arr[i];  
 count[(arr[i] / exp) % 10]--;  
 }  
  
 for (int i = 0; i < n; i++) {  
 arr[i] = output[i];  
 }  
 }  
 public static void countSortDesc(int arr[], int exp) {  
 int n = arr.length;  
 int output[] = new int[n];  
 int count[] = new int[10];  
 Arrays.*fill*(count, 0);  
  
 for (int i = 0; i < n; i++) {  
 count[9 - (arr[i] / exp) % 10]++;  
 }  
  
 for (int i = 1; i < 10; i++) {  
 count[i] += count[i - 1];  
 }  
  
 for (int i = n - 1; i >= 0; i--) {  
 output[count[9 - (arr[i] / exp) % 10] - 1] = arr[i];  
 count[9 - (arr[i] / exp) % 10]--;  
 }  
  
 for (int i = 0; i < n; i++) {  
 arr[i] = output[i];  
 }  
 }  
}



Q4.

import java.util.Arrays;  
  
public class Task4 {  
 public static int binarySearch(int[] arr, int target) {  
 int left = 0;  
 int right = arr.length - 1;  
  
 while (left <= right) {  
 int mid = left + (right - left) / 2;  
  
 if (arr[mid] == target) {  
 return mid;  
 } else if (arr[mid] < target) {  
 left = mid + 1;  
 } else {  
 right = mid - 1;  
 }  
 }  
 return -1;  
 }  
 public static int[] addAndSort(int[] arr, int value) {  
 int[] newArray = Arrays.*copyOf*(arr, arr.length + 1);  
 newArray[newArray.length - 1] = value;  
 Arrays.*sort*(newArray);  
 return newArray;  
 }  
 public static void main(String[] args) {  
 int[] arr = {1, 3, 12, 14, 23, 34, 55, 65, 75, 78};  
 int rollNumber = 4818;  
 int lastTwoDigits = rollNumber % 100;  
  
 Arrays.*sort*(arr);  
 System.*out*.println("Original Array : " + Arrays.*toString*(arr));  
  
 int index = *binarySearch*(arr, lastTwoDigits);  
  
 if (index == -1) {  
 arr = *addAndSort*(arr, lastTwoDigits);  
 System.*out*.println("Value not found. Added and sorted array: " + Arrays.*toString*(arr));  
 index = *binarySearch*(arr, lastTwoDigits);  
 }  
  
 if (index != -1) {  
 int result = arr[index];  
 System.*out*.println("Value corresponding to the last two digits (18): " + result);  
 } else {  
 System.*out*.println("Value corresponding to the last two digits not found.");  
 }  
 }  
}



Q5.

import java.util.Scanner;  
  
public class Task5 {  
 public static int InterpolationSearch(int[] array , int lo , int hi, int x){  
 int pos;  
 if (lo <= hi && x >= array[lo] && x <= array[hi]){  
 pos = lo + ((hi - lo) / (array[hi]-array[lo])) \* (x - array[lo]);  
 if (array[pos] == x){  
 return array[pos];  
 }  
 if (array[pos] < x){  
 return *InterpolationSearch*(array, pos+1 ,hi,x);  
 }  
 if (array[pos] > x){  
 return *InterpolationSearch*(array, lo ,pos-1,x);  
 }  
 }  
 return -1;  
 }  
  
 public static void main(String[] args) {  
 int array[] = {1,2,3,4,5,6,7,8,9};  
 int lo = 0;  
 int hi = array.length-1;  
 System.*out*.println("Enter the Element You Want To Search : ");  
 Scanner inp = new Scanner(System.*in*);  
 int x = inp.nextInt();  
 System.*out*.println("Interpolation Search: ");  
 int y = *InterpolationSearch*(array,lo,hi,x);  
 if (y == -1)  
 System.*out*.println("Element Does not Exist");  
 else System.*out*.println("element found : " + y);  
 }  
}

