***Assignment***

**Bubble sort program**

**import java.util.Arrays;**

*public class BubbleSort {*

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

*int[] arr = {64, 34, 25, 12, 22, 11, 90};*

*System.out.println("Array before sorting: " + Arrays.toString(arr));*

*bubbleSort(arr);*

*System.out.println("Array after sorting: " + Arrays.toString(arr));*

*}*

*public static void bubbleSort(int[] arr) {*

*int n = arr.length;*

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

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

*if (arr[j] > arr[j + 1]) {*

*int temp = arr[j];*

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

*arr[j + 1] = temp;*

*}*

*}*

*}*

*}*

*}*

**Output**

***Array before sorting: [64, 34, 25, 12, 22, 11, 90]***

***Array after sorting: [11, 12, 22, 25, 34, 64, 90]***

**Quick sort**

*import java.util.Arrays;*

*public class QuickSort {*

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

*int[] arr = {64, 34, 25, 12, 22, 11, 90};*

*System.out.println("Array before sorting: " + Arrays.toString(arr));*

*quickSort(arr, 0, arr.length - 1);*

*System.out.println("Array after sorting: " + Arrays.toString(arr));*

*}*

*public static void quickSort(int[] arr, int low, int high) {*

*if (low < high) {*

*int pi = partition(arr, low, high);*

*quickSort(arr, low, pi - 1);*

*quickSort(arr, pi + 1, high);*

*}*

*}*

*public static int partition(int[] arr, int low, int high) {*

*int pivot = arr[high];*

*int i = low - 1;*

*for (int j = low; j < high; j++) {*

*if (arr[j] < pivot) {*

*i++;*

*int temp = arr[i];*

*arr[i] = arr[j];*

*arr[j] = temp;*

*}*

*}*

*int temp = arr[i + 1];*

*arr[i + 1] = arr[high];*

*arr[high] = temp;*

*return i + 1;*

*}*

*}*

Output :

Array before sorting: [64, 34, 25, 12, 22, 11, 90]

Array after sorting: [11, 12, 22, 25, 34, 64, 90]

**Selection sort:**

*import java.util.Arrays;*

*public class SelectionSort {*

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

*int[] arr = {64, 34, 25, 12, 22, 11, 90};*

*System.out.println("Array before sorting: " + Arrays.toString(arr));*

*selectionSort(arr);*

*System.out.println("Array after sorting: " + Arrays.toString(arr));*

*}*

*public static void selectionSort(int[] arr) {*

*int n = arr.length;*

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

*int minIndex = i;*

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

*if (arr[j] < arr[minIndex]) {*

*minIndex = j;*

*}*

*}*

*int temp = arr[i];*

*arr[i] = arr[minIndex];*

*arr[minIndex] = temp;*

*}*

*}*

*}*

Output

Array before sorting: [64, 34, 25, 12, 22, 11, 90]

Array after sorting: [11, 12, 22, 25, 34, 64, 90]

Insertion sort

*import java.util.Arrays;*

*public class InsertionSort {*

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

*int[] arr = {64, 34, 25, 12, 22, 11, 90};*

*System.out.println("Array before sorting: " + Arrays.toString(arr));*

*insertionSort(arr);*

*System.out.println("Array after sorting: " + Arrays.toString(arr));*

*}*

*public 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 >= 0 && arr[j] > key) {*

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

*j--;*

*}*

*arr[j + 1] = key;*

*}*

*}*

*}*

Output:

Array before sorting: [64, 34, 25, 12, 22, 11, 90]

Array after sorting: [11, 12, 22, 25, 34, 64, 90]

**Merge sort**

import java.util.Arrays;

public class MergeSort {

public static void main(String[] args) {

int[] arr = {64, 34, 25, 12, 22, 11, 90};

System.out.println("Array before sorting: " + Arrays.toString(arr));

mergeSort(arr, 0, arr.length - 1);

System.out.println("Array after sorting: " + Arrays.toString(arr));

}

public static void mergeSort(int[] arr, int left, int right) {

if (left < right) {

int mid = left + (right - left) / 2;

mergeSort(arr, left, mid);

mergeSort(arr, mid + 1, right);

merge(arr, left, mid, right);

}

}

public static void merge(int[] arr, int left, int mid, int right) {

int n1 = mid - left + 1;

int n2 = right - mid;

int[] L = new int[n1];

int[] R = new int[n2];

for (int i = 0; i < n1; i++) {

L[i] = arr[left + i];

}

for (int j = 0; j < n2; j++) {

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

}

int i = 0, j = 0;

int k = left;

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

} else {

arr[k] = R[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

}

Output

Array before sorting: [64, 34, 25, 12, 22, 11, 90]

Array after sorting: [11, 12, 22, 25, 34, 64, 90]

**Quick sort**

*import java.util.Arrays;*

*public class QuickSort {*

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

*int[] arr = {64, 34, 25, 12, 22, 11, 90};*

*System.out.println("Array before sorting: " + Arrays.toString(arr));*

*quickSort(arr, 0, arr.length - 1);*

*System.out.println("Array after sorting: " + Arrays.toString(arr));*

*}*

*public static void quickSort(int[] arr, int low, int high) {*

*if (low < high) {*

*int pi = partition(arr, low, high);*

*quickSort(arr, low, pi - 1);*

*quickSort(arr, pi + 1, high);*

*}*

*}*

*public static int partition(int[] arr, int low, int high) {*

*int pivot = arr[high];*

*int i = low - 1;*

*for (int j = low; j < high; j++) {*

*if (arr[j] < pivot) {*

*i++;*

*int temp = arr[i];*

*arr[i] = arr[j];*

*arr[j] = temp;*

*}*

*}*

*int temp = arr[i + 1];*

*arr[i + 1] = arr[high];*

*arr[high] = temp;*

*return i + 1;*

*}*

*}*

Output

Array before sorting: [64, 34, 25, 12, 22, 11, 90]

Array after sorting: [11, 12, 22, 25, 34, 64, 90]

**Sorting strings using bubble sort**

import java.util.Arrays;

public class BubbleSortStrings {

public static void main(String[] args) {

String[] arr = {"zebra", "lion", "elephant", "tiger", "giraffe"};

System.out.println("Array before sorting: " + Arrays.toString(arr));

bubbleSort(arr);

System.out.println("Array after sorting: " + Arrays.toString(arr));

}

public static void bubbleSort(String[] arr) {

int n = arr.length;

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

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

if (arr[j].compareTo(arr[j + 1]) > 0) {

// Swap arr[j] and arr[j+1]

String temp = arr[j];

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

arr[j + 1] = temp;

}

}

}

}

}

Output

Array before sorting: [zebra, lion, elephant, tiger, giraffe]

Array after sorting: [elephant, giraffe, lion, tiger, zebra]

**Bubble sort for linked list by swapping nodes**

*class Node {*

*int data;*

*Node next;*

*Node(int d) {*

*data = d;*

*next = null;*

*}*

*}*

*public class BubbleSortLinkedList {*

*Node head;*

*void bubbleSort() {*

*if (head == null || head.next == null)*

*return;*

*boolean swapped;*

*do {*

*swapped = false;*

*Node prev = null;*

*Node curr = head;*

*Node next = head.next;*

*while (next != null) {*

*if (curr.data > next.data) {*

*if (prev != null)*

*prev.next = next;*

*else*

*head = next;*

*curr.next = next.next;*

*next.next = curr;*

*prev = next;*

*next = curr.next;*

*swapped = true;*

*} else {*

*prev = curr;*

*curr = next;*

*next = next.next;*

*}*

*}*

*} while (swapped);*

*}*

*void push(int new\_data) {*

*Node new\_node = new Node(new\_data);*

*new\_node.next = head;*

*head = new\_node;*

*}*

*void printList() {*

*Node temp = head;*

*while (temp != null) {*

*System.out.print(temp.data + " ");*

*temp = temp.next;*

*}*

*System.out.println();*

*}*

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

*BubbleSortLinkedList list = new BubbleSortLinkedList();*

*list.push(64);*

*list.push(34);*

*list.push(25);*

*list.push(12);*

*list.push(22);*

*list.push(11);*

*list.push(90);*

*System.out.println("Linked List before sorting:");*

*list.printList();*

*list.bubbleSort();*

*System.out.println("Linked List after sorting:");*

*list.printList();*

*}*

*}*

Output:

Linked List before sorting:

90 11 22 12 25 34 64

Linked List after sorting:

11 12 22 25 34 64 90

**Bubble sort on Doubly Linked List**

*class Node {*

*int data;*

*Node prev, next;*

*Node(int d) {*

*data = d;*

*prev = next = null;*

*}*

*}*

*public class BubbleSortDoublyLinkedList {*

*Node head; void bubbleSort() {*

*if (head == null || head.next == null)*

*return;*

*boolean swapped;*

*Node last = null;*

*do {*

*swapped = false;*

*Node current = head;*

*while (current.next != last) {*

*if (current.data > current.next.data) {*

*swap(current, current.next);*

*swapped = true;*

*}*

*current = current.next;*

*}*

*last = current;*

*} while (swapped);*

*}*

*void swap(Node n1, Node n2) {*

*int temp = n1.data;*

*n1.data = n2.data;*

*n2.data = temp;*

*}*

*void insert(int new\_data) {*

*Node new\_node = new Node(new\_data);*

*new\_node.next = head;*

*new\_node.prev = null;*

*if (head != null)*

*head.prev = new\_node;*

*head = new\_node;*

*}*

*void printList(Node node) {*

*while (node != null) {*

*System.out.print(node.data + " ");*

*node = node.next;*

*}*

*System.out.println();*

*}*

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

*BubbleSortDoublyLinkedList list = new BubbleSortDoublyLinkedList();*

*list.insert(64);*

*list.insert(34);*

*list.insert(25);*

*list.insert(12);*

*list.insert(22);*

*list.insert(11);*

*list.insert(90);*

*System.out.println("Doubly Linked List before sorting:");*

*list.printList(list.head);*

*list.bubbleSort();*

*System.out.println("Doubly Linked List after sorting:");*

*list.printList(list.head);*

*}*

*}*

Output:

Doubly Linked List before sorting:

90 11 22 12 25 34 64

Doubly Linked List after sorting:

11 12 22 25 34 64 90