**Q.1 Implementation of Array operations - Insert, Delete, Search, Update, and Display.**

*/\* PRACTICAL-1: Implementation of Array operations - Insert, Delete, Search, Update, and Display.*

*Name: Angat Shah*

*Enrollment No: 202203103510097*

*Branch: B.Tech Computer Science and Engineering \*/*

*// CODE:*

public class practical1 {

private int[] array;

private int size;

public practical1(int length) {

array = new int[length];

size = 0;

}

public void insert(int value) {

if (size < array.length) {

array[size] = value;

size++;

} else {

System.out.println("Array is full. Cannot insert " + value);

}

}

public void delete(int index) {

if (index >= 0 && index < size) {

for (int i = index; i < size - 1; i++) {

array[i] = array[i + 1];

}

size--;

} else {

System.out.println("Invalid index. Cannot delete from " + index);

}

}

public int search(int value) {

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

if (array[i] == value) {

return i;

}

}

return -1;

}

public void update(int index, int value) {

if (index >= 0 && index < size) {

array[index] = value;

} else {

System.out.println("Invalid index. Cannot update " + index);

}

}

public void display() {

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

System.out.print(array[i] + " ");

}

System.out.println();

}

public static void main(String[] args) {

practical1 array = new practical1(5);

array.insert(1);

array.insert(5);

array.insert(7);

array.insert(8);

array.insert(11);

System.out.print("--> Given Array: ");

array.display();

array.delete(2);

System.out.print("--> After Deleting '5' from the Array: ");

array.display();

array.update(3, 30);

System.out.print("--> After Updating the element at '3rd' Index to '30' in the Array: ");

array.display();

System.out.println("--> Index of '11': " + array.search(30));

}

}

**Q.2 Implementation of Array applications of Sparse Matrices.**

*/\* PRACTICAL-2: Implementation of Array applications of Sparse Matrices.*

*Name: Angat Shah*

*Enrollment No: 202203103510097*

*Branch: B.Tech Computer Science and Engineering \*/*

*// CODE:*

import java.util.ArrayList;

class SparseMatrix {

private int rows;

private int columns;

private ArrayList<SparseMatrixElement> elements;

public SparseMatrix(int rows, int columns) {

*this*.rows = rows;

*this*.columns = columns;

*this*.elements = new ArrayList<>();

}

public void setElement(int row, int col, int value) {

if (row < 0 || row >= rows || col < 0 || col >= columns) {

throw new IllegalArgumentException("Invalid matrix indices");

}

elements.removeIf(element -> element.getRow() == row && element.getColumn() == col);

if (value != 0) {

elements.add(new SparseMatrixElement(row, col, value));

}

}

public int getElement(int row, int col) {

if (row < 0 || row >= rows || col < 0 || col >= columns) {

throw new IllegalArgumentException("Invalid matrix indices");

}

for (SparseMatrixElement element : elements) {

if (element.getRow() == row && element.getColumn() == col) {

return element.getValue();

}

}

return 0;

}

public void printMatrix() {

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

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

System.out.print(getElement(i, j) + " ");

}

System.out.println();

}

}

}

class SparseMatrixElement {

private int row;

private int column;

private int value;

public SparseMatrixElement(int row, int column, int value) {

*this*.row = row;

*this*.column = column;

*this*.value = value;

}

public int getRow() {

return row;

}

public int getColumn() {

return column;

}

public int getValue() {

return value;

}

}

public class practical2 {

public static void main(String[] args) {

SparseMatrix sparseMatrix = new SparseMatrix(3, 3);

System.out.println("--> Generated Sparse Matrix on the Given Input: ");

sparseMatrix.setElement(0, 0, 1);

sparseMatrix.setElement(1, 1, 2);

sparseMatrix.setElement(2, 2, 3);

sparseMatrix.printMatrix();

}

}

**Q.3 Implement a program for stack that performs following operations using array.**

**(a) PUSH (b) POP (c) PEEP (d) CHANGE (e) DISPLAY**

*/\* PRACTICAL-3: Implement a program for stack that performs following operations using array.*

*(a) PUSH (b) POP (c) PEEP (d) CHANGE (e) DISPLAY*

*Name: Angat Shah*

*Enrollment No: 202203103510097*

*Branch: B.Tech Computer Science and Engineering \*/*

*// CODE:*

public class practical3 {

private int maxSize;

private int top;

private int[] stackArray;

public practical3(int size) {

*this*.maxSize = size;

*this*.stackArray = new int[maxSize];

*this*.top = -1;

}

public void push(int value) {

if (top < maxSize - 1) {

top++;

stackArray[top] = value;

System.out.println("--> Pushed: " + value);

} else {

System.out.println("--> Stack is full. Cannot push: " + value);

}

}

public void pop() {

if (top >= 0) {

System.out.println("--> Popped: " + stackArray[top]);

top--;

} else {

System.out.println("--> Stack is empty.");

}

}

public void peek() {

if (top >= 0) {

System.out.println("--> Top element: " + stackArray[top]);

} else {

System.out.println("--> Stack is empty.");

}

}

public void change(int value, int newValue) {

boolean found = false;

if (top >= 0) {

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

if (stackArray[i] == value) {

stackArray[i] = newValue;

System.out.println("--> Value changed from " + value + " TO " + newValue);

found = true;

break;

}

}

if (!found) {

System.out.println("--> Value not found in stack.");

}

} else {

System.out.println("--> Stack is empty.");

}

}

public void display() {

if (top >= 0) {

System.out.print("--> Stack: [");

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

System.out.print(stackArray[i] + " ");

}

System.out.println("]");

} else {

System.out.println("--> Stack is empty.");

}

}

public static void main(String[] args) {

practical3 stack = new practical3(5);

stack.push(10);

stack.push(20);

stack.push(30);

stack.push(40);

stack.push(50);

stack.display();

stack.push(60);

stack.pop();

stack.pop();

stack.display();

stack.peek();

stack.change(30, 300);

stack.display();

stack.change(100, 1000);

}

}

**Q.4 Write a program to implement Queue using arrays that performs following operations.**

**(a) INSERT (b) DELETE (c) DISPLAY**

*/\* PRACTICAL-4: Write a program to implement Queue using arrays that performs following operations.*

*(a) INSERT (b) DELETE (c) DISPLAY*

*Name: Angat Shah*

*Enrollment No: 202203103510097*

*Branch: B.Tech Computer Science and Engineering \*/*

*// CODE:*

import java.util.Scanner;

public class practical4 {

private int[] queue;

private int front;

private int rear;

private int capacity;

private int size;

public practical4(int capacity) {

*this*.capacity = capacity;

*this*.queue = new int[capacity];

*this*.front = 0;

*this*.rear = -1;

*this*.size = 0;

}

public void insert(int item) {

if (isFull()) {

System.out.println("Queue is full. Cannot insert more elements.");

return;

}

rear = (rear + 1) % capacity;

queue[rear] = item;

size++;

System.out.println("Inserted element: " + item);

}

public int delete() {

if (isEmpty()) {

System.out.println("Queue is empty. Cannot delete element.");

return -1;

}

int deletedItem = queue[front];

front = (front + 1) % capacity;

size--;

return deletedItem;

}

public void display() {

if (isEmpty()) {

System.out.println("Queue is empty.");

return;

}

System.out.print("Queue elements: ");

int count = 0;

int index = front;

while (count < size) {

System.out.print(queue[index] + " ");

index = (index + 1) % capacity;

count++;

}

System.out.println();

}

public boolean isFull() {

return size == capacity;

}

public boolean isEmpty() {

return size == 0;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the capacity of the queue: ");

int capacity = scanner.nextInt();

practical4 queue = new practical4(capacity);

int choice;

do {

System.out.println("\nQueue Operations:");

System.out.println("1. Insert");

System.out.println("2. Delete");

System.out.println("3. Display");

System.out.println("4. Exit");

System.out.print("Enter your choice: ");

choice = scanner.nextInt();

switch (choice) {

case 1:

System.out.print("Enter element to insert: ");

int element = scanner.nextInt();

queue.insert(element);

break;

case 2:

int deletedItem = queue.delete();

if (deletedItem != -1)

System.out.println("Deleted element: " + deletedItem);

break;

case 3:

queue.display();

break;

case 4:

System.out.println("Exiting...");

break;

default:

System.out.println("Invalid choice. Please enter a valid option.");

}

} while (choice != 4);

scanner.close();

}

}

**Q.5 Write a menu driven program to implement following operations on the singly linked list.**

**(a) Insert a node at the front of the linked list**

**(b) Insert a node at the end of the linked list**

**(c) Insert a node such that linked list is in ascending order**

**(d) Delete a First node of the linked list**

**(e) Delete a node before specified position**

**(f) Delete a node after specified position.**

import java.util.Scanner;

class Node {

int data;

Node next;

Node(int data) {

*this*.data = data;

*this*.next = null;

}

}

class LinkedList {

Node head;

LinkedList() {

head = null;

}

*// Insert a node at the front of the linked list*

void insertAtFront(int data) {

Node newNode = new Node(data);

newNode.next = head;

head = newNode;

}

*// Insert a node at the end of the linked list*

void insertAtEnd(int data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

return;

}

Node temp = head;

while (temp.next != null) {

temp = temp.next;

}

temp.next = newNode;

}

*// Insert a node in ascending order*

void insertInAscendingOrder(int data) {

Node newNode = new Node(data);

if (head == null || head.data >= data) {

newNode.next = head;

head = newNode;

return;

}

Node current = head;

while (current.next != null && current.next.data < data) {

current = current.next;

}

newNode.next = current.next;

current.next = newNode;

}

*// Delete the first node of the linked list*

void deleteFirstNode() {

if (head == null) {

System.out.println("-->> List is empty");

return;

}

head = head.next;

}

*// Delete a node before specified position*

void deleteNodeBeforePosition(int position) {

if (head == null || position < 1) {

System.out.println("Invalid position or list is empty");

return;

}

Node temp = head;

for (int i = 1; temp != null && i < position - 1; i++) {

temp = temp.next;

}

if (temp == null || temp.next == null) {

System.out.println("Position out of range");

return;

}

temp.next = temp.next.next;

}

*// Delete a node after specified position*

void deleteNodeAfterPosition(int position) {

if (head == null) {

System.out.println("List is empty");

return;

}

Node temp = head;

for (int i = 1; temp != null && i < position; i++) {

temp = temp.next;

}

if (temp == null || temp.next == null) {

System.out.println("Position out of range");

return;

}

temp.next = temp.next.next;

}

*// Display the linked list*

void display() {

Node temp = head;

while (temp != null) {

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

temp = temp.next;

}

System.out.println();

}

}

public class demo {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

LinkedList linkedList = new LinkedList();

int choice;

do {

System.out.println("\n--> Operation Menu:");

System.out.println("1. Insert at the front");

System.out.println("2. Insert at the end");

System.out.println("3. Insert in ascending order");

System.out.println("4. Delete first node");

System.out.println("5. Delete node before specified position");

System.out.println("6. Delete node after specified position");

System.out.println("7. Display linked list");

System.out.println("0. Exit");

System.out.print("Enter your choice: ");

choice = scanner.next().charAt(0);

switch (choice) {

case '1':

System.out.print("-->> Enter Data to Insert at the Front: ");

int frontData = scanner.nextInt();

linkedList.insertAtFront(frontData);

break;

case '2':

System.out.print("-->> Enter Data to Insert at the End: ");

int endData = scanner.nextInt();

linkedList.insertAtEnd(endData);

break;

case '3':

System.out.print("-->> Enter Data to Insert in Ascending Order: ");

int ascData = scanner.nextInt();

linkedList.insertInAscendingOrder(ascData);

break;

case '4':

linkedList.deleteFirstNode();

break;

case '5':

System.out.print("-->> Enter position before which to delete: ");

int posBefore = scanner.nextInt();

linkedList.deleteNodeBeforePosition(posBefore);

break;

case '6':

System.out.print("-->> Enter position after which to delete: ");

int posAfter = scanner.nextInt();

linkedList.deleteNodeAfterPosition(posAfter);

break;

case '7':

linkedList.display();

break;

case '0':

System.out.println("#Exiting...");

break;

default:

System.out.println("Invalid choice");

}

} while (choice != '0');

scanner.close();

}

}

**Q.6 Implementation of Binary Search Trees.**

*/\* PRACTICAL-6: Implementation of Binary Search Trees.*

*Name: Angat Shah*

*Enrollment No: 202203103510097*

*Branch: B.Tech Computer Science and Engineering\*/*

*// CODE:*

import java.util.Scanner;

class TreeNode {

int data;

TreeNode left;

TreeNode right;

public TreeNode(int data) {

*this*.data = data;

*this*.left = null;

*this*.right = null;

}

}

class BinarySearchTree {

private TreeNode root;

public BinarySearchTree() {

root = null;

}

public void insert(int data) {

root = insertRec(root, data);

}

private TreeNode insertRec(TreeNode root, int data) {

if (root == null) {

root = new TreeNode(data);

return root;

}

if (data < root.data)

root.left = insertRec(root.left, data);

else if (data > root.data)

root.right = insertRec(root.right, data);

return root;

}

public void inorder() {

inorderRec(root);

}

private void inorderRec(TreeNode root) {

if (root != null) {

inorderRec(root.left);

System.out.print(root.data + " ");

inorderRec(root.right);

}

}

public void preorder() {

preorderRec(root);

}

private void preorderRec(TreeNode root) {

if (root != null) {

System.out.print(root.data + " ");

preorderRec(root.left);

preorderRec(root.right);

}

}

public void postorder() {

postorderRec(root);

}

private void postorderRec(TreeNode root) {

if (root != null) {

postorderRec(root.left);

postorderRec(root.right);

System.out.print(root.data + " ");

}

}

}

public class practical6 {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

BinarySearchTree bst = new BinarySearchTree();

int choice;

do {

System.out.println("\nBinary Search Tree Operations:");

System.out.println("1. Insert");

System.out.println("2. Inorder Traversal");

System.out.println("3. Preorder Traversal");

System.out.println("4. Postorder Traversal");

System.out.println("5. Exit");

System.out.print("Enter your choice: ");

choice = scanner.nextInt();

switch (choice) {

case 1:

System.out.print("Enter element to insert: ");

int element = scanner.nextInt();

bst.insert(element);

break;

case 2:

System.out.print("Inorder Traversal: ");

bst.inorder();

System.out.println();

break;

case 3:

System.out.print("Preorder Traversal: ");

bst.preorder();

System.out.println();

break;

case 4:

System.out.print("Postorder Traversal: ");

bst.postorder();

System.out.println();

break;

case 5:

System.out.println("Exiting...");

break;

default:

System.out.println("Invalid choice. Please enter a valid option.");

}

} while (choice != 5);

scanner.close();

}

}

**Q.7 Implementation of Sorting techniques.**

**(a) Bubble Sort**

**(b) Selection Sort**

**(c) Merge Sort.**

*/\* PRACTICAL-7: Implementation of Sorting techniques.*

*(a) Bubble Sort*

*(b) Selection Sort*

*(c) Merge Sort.*

*Name: Angat Shah*

*Enrollment No: 202203103510097*

*Branch: B.Tech Computer Science and Engineering \*/*

*// CODE:*

import java.util.Scanner;

public class practical7 {

*// Bubble Sort*

public static void bubbleSort(int[] arr) {

int n = arr.length;

boolean swapped;

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

swapped = false;

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;

swapped = true;

}

}

if (!swapped) *// If no two elements were swapped in the inner loop, then the array is already sorted*

break;

System.out.println("Step " + (i + 1) + ": " + arrayToString(arr));

}

}

*// Selection Sort*

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[minIndex];

arr[minIndex] = arr[i];

arr[i] = temp;

System.out.println("Step " + (i + 1) + ": " + arrayToString(arr));

}

}

*// Merge Sort*

public static void mergeSort(int[] arr, int l, int r) {

if (l < r) {

int mid = (l + r) / 2;

mergeSort(arr, l, mid);

mergeSort(arr, mid + 1, r);

merge(arr, l, mid, r);

}

}

private static void merge(int[] arr, int l, int mid, int r) {

int n1 = mid - l + 1;

int n2 = r - mid;

int[] leftArr = new int[n1];

int[] rightArr = new int[n2];

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

leftArr[i] = arr[l + i];

}

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

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

}

int i = 0, j = 0, k = l;

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

if (leftArr[i] <= rightArr[j]) {

arr[k] = leftArr[i];

i++;

} else {

arr[k] = rightArr[j];

j++;

}

k++;

}

while (i < n1) {

arr[k] = leftArr[i];

i++;

k++;

}

while (j < n2) {

arr[k] = rightArr[j];

j++;

k++;

}

}

*// Utility method to print array*

private static String arrayToString(int[] arr) {

StringBuilder sb = new StringBuilder();

for (int num : arr) {

sb.append(num).append(" ");

}

return sb.toString();

}

*// Main method to test the sorting algorithms with user input*

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

*// Taking user input for array size*

System.out.print("Enter the size of the array: ");

int size = scanner.nextInt();

int[] arr = new int[size];

*// Taking user input for array elements*

System.out.println("Enter the elements of the array:");

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

arr[i] = scanner.nextInt();

}

*// Choosing sorting algorithm*

System.out.println("Choose sorting algorithm:");

System.out.println("1. Bubble Sort");

System.out.println("2. Selection Sort");

System.out.println("3. Merge Sort");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

switch (choice) {

case 1:

System.out.println("\nApplying Bubble Sort:");

bubbleSort(arr);

break;

case 2:

System.out.println("\nApplying Selection Sort:");

selectionSort(arr);

break;

case 3:

System.out.println("\nApplying Merge Sort:");

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

break;

default:

System.out.println("Invalid choice!");

}

*// Displaying sorted array*

System.out.println("\nSorted Array:");

System.out.println(arrayToString(arr));

scanner.close();

}

}

**Q.8 Implementation of Searching techniques.**

**(a) Sequential Search**

**(b) Binary Search.**

*/\* PRACTICAL-8: Implementation of Searching techniques.*

*(a) Sequential Search*

*(b) Binary Search.*

*Name: Angat Shah*

*Enrollment No: 202203103510097*

*Branch: B.Tech Computer Science and Engineering \*/*

*// CODE:*

import java.util.Scanner;

public class practical8 {

*// I. Sequential Search*

public static int sequentialSearch(int[] arr, int target) {

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

if (arr[i] == target) {

return i;

}

}

return -1;

}

*// II. Binary Search*

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;

}

private static String arrayToString(int[] arr) {

StringBuilder sb = new StringBuilder();

for (int num : arr) {

sb.append(num).append(" ");

}

return sb.toString();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the size of the array: ");

int size = scanner.nextInt();

int[] arr = new int[size];

System.out.println("Enter the elements of the array:");

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

arr[i] = scanner.nextInt();

}

System.out.println("Sorting the array...");

bubbleSort(arr);

System.out.println("Sorted Array:");

System.out.println(arrayToString(arr));

System.out.println("\nChoose searching technique:");

System.out.println("1. Sequential Search");

System.out.println("2. Binary Search");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt();

System.out.print("Enter the target element to search: ");

int target = scanner.nextInt();

int index = -1;

switch (choice) {

case 1:

System.out.println("\nApplying Sequential Search:");

index = sequentialSearch(arr, target);

break;

case 2:

System.out.println("\nApplying Binary Search:");

index = binarySearch(arr, target);

break;

default:

System.out.println("Invalid choice!");

}

if (index != -1) {

System.out.println("Element found at index: " + index);

} else {

System.out.println("Element not found in the array.");

}

scanner.close();

}

private 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;

}

}

}

}

}