PRACTICAL 1

In a class, some students like math, some like science, and some both the subjects. Find the students

who like (i) Math only i.e. (M) - (MnS), (ii) Science only i.e. (S) - (MnS), (iii) Either Math or Science

(MuS).

import java.util.HashSet;

import java.util.Set;

class StudentSubjects {

public static void main(String[] args) {

// Create 3 sets of mathlovers , sciencelovers , bothlovers

// using Set class

Set<String> mathLovers = new HashSet<>();

Set<String> scienceLovers = new HashSet<>();

Set<String> bothLovers = new HashSet<>();

// Populate the sets with student names and their preferences

mathLovers.add("S1");

mathLovers.add("S2");

bothLovers.add("S3");

scienceLovers.add("S4");

bothLovers.add("S5");

mathLovers.add("S6");

// Find students who like Math only (M - MnS)

Set<String> mathOnly = new HashSet<>(mathLovers);

mathOnly.removeAll(bothLovers);

// Find students who like Science only (S - MnS)

Set<String> scienceOnly = new HashSet<>(scienceLovers);

scienceOnly.removeAll(bothLovers);

// Find students who like either Math or Science (MuS)

Set<String> eitherMathOrScience = new HashSet<>();

eitherMathOrScience.addAll(mathLovers);

eitherMathOrScience.addAll(scienceLovers);

System.out.println("Students who like Math only: " + mathOnly + "\n");

System.out.println("Students who like Science only: " + scienceOnly + "\n");

System.out.println("Students who like either Math or Science: " + eitherMathOrScience);

}

}

PRACTICAL 2

Perform matrix operations with pointers (C++)

#include <iostream>

#include <string.h>

using namespace std;

class mat

{

public:

int \*matrix;

int x;

int y;

mat(int x, int y)

{

this->x = x;

this->y = y;

this->matrix = new int[x \* y]();

}

void print()

{

cout << "\n";

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

{

cout << "[ ";

for (int j = 0; j < y; j++)

{

cout << \*(matrix + j + i \* y) << " "; // Corrected indexing

}

cout << "]\n";

}

}

void set()

{

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

{

for (int j = 0; j < y; j++)

{

cout << "Enter the Element " << i + 1 << "x" << j + 1 << " : ";

cin >> \*(matrix + j + i \* y);

}

}

}

mat transpose()

{

mat m(y, x);

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

{

for (int j = 0; j < x; j++)

{

\*(m.matrix + j + i \* y) = \*(matrix + j \* x + i);

}

}

return m;

}

};

mat add(mat m1, mat m2)

{

int x = m1.x, y = m1.y;

mat m(x, y);

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

{

for (int j = 0; j < y; j++)

{

\*(m.matrix + j + i \* y) = \*(m1.matrix + j + i \* y) + \*(m2.matrix + j + i \* y);

}

}

return m;

}

mat subtract(mat m1, mat m2)

{

int x = m1.x, y = m1.y;

mat m(x, y);

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

{

for (int j = 0; j < y; j++)

{

\*(m.matrix + j + i \* y) = \*(m1.matrix + j + i \* y) - \*(m2.matrix + j + i \* y);

}

}

return m;

}

mat takeDimensions(string str)

{

int x, y;

cout << "Enter the no. of Rows for " << str << " : ";

cin >> x;

cout << "Enter the no. of columns for " << str << " : ";

cin >> y;

mat m(x, y);

cout << "For " << str << " : \n";

m.set();

return m;

}

mat multiply(mat m1, mat m2)

{

int x1 = m1.x, y1 = m1.y;

int x2 = m2.x, y2 = m2.y;

mat m(x1, y2);

if (y1 != x2)

{

return m;

}

for (int i = 0; i < x1; i++)

{

for (int j = 0; j < y2; j++)

{

for (int k = 0; k < y1; k++)

{

\*(m.matrix + j + i \* y2) += \*(m1.matrix + k + i \* y1) \* \*(m2.matrix + j + k \* y2);

}

}

}

return m;

}

int main()

{

mat m1 = takeDimensions("mat1");

mat m2 = takeDimensions("mat2");

bool end = false;

int inp, mat;

while (!end)

{

cout << "Enter the Operation\n1.print\n2.transpose\n3.add\n4.subtract\n5.multiply\n6.Exit\n";

cin >> inp;

switch (inp)

{

case 1:

cout << "1.mat1 / 2.mat2 ? : ";

cin >> mat;

if (mat == 1)

{

m1.print();

break;

}

m2.print();

break;

case 2:

cout << "1.mat1 / 2.mat2 ? : ";

cin >> mat;

if (mat == 1)

{

m1.transpose().print();

break;

}

m2.transpose().print();

break;

case 3:

add(m1, m2).print();

break;

case 4:

subtract(m1, m2).print();

break;

case 5:

multiply(m1, m2).print();

break;

case 6:

end = true;

break;

default:

cout << "Try Valid Input\n";

break;

}

}

return 0;

}

PRACTICAL 3

Create a database for inventory system of a shop using array of structures and perform following

operations on it: i. Add record ii. Display Database iii. Search record (Use linear search algorithm)

iv. Delete record v. Update records vi. Sort records using bubble sort.

import java.util.Scanner;

class InventoryItem {

int itemCode;

double itemPrice;

public InventoryItem(int itemCode, double itemPrice) {

this.itemCode = itemCode;

this.itemPrice = itemPrice;

}

}

public class InventorySystem {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

InventoryItem[] inventory = new InventoryItem[100];

int size = 0;

int choice;

do {

displayMenu();

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

choice = scanner.nextInt();

switch (choice) {

case 1:

addRecord(inventory, scanner, size++);

break;

case 2:

displayDatabase(inventory, size);

break;

case 3:

searchRecord(inventory, size, scanner.nextInt());

break;

case 4:

deleteRecord(inventory, size, scanner.nextInt());

size--;

break;

case 5:

sortRecordsByPrice(inventory, size);

System.out.println("Records sorted successfully.");

break;

case 0:

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

break;

default:

System.out.println("Invalid choice. Please try again.");

}

} while (choice != 0);

scanner.close();

}

private static void displayMenu() {

System.out.println("\nInventory System Menu");

System.out.println("1. Add Record");

System.out.println("2. Display Database");

System.out.println("3. Search Record");

System.out.println("4. Delete Record");

System.out.println("5. Sort Records by Price");

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

}

private static void addRecord(InventoryItem[] inventory, Scanner scanner, int size) {

System.out.print("Enter item code: ");

int itemCode = scanner.nextInt();

System.out.print("Enter item price: ");

double itemPrice = scanner.nextDouble();

inventory[size] = new InventoryItem(itemCode, itemPrice);

System.out.println("Record added successfully.");

}

private static void displayDatabase(InventoryItem[] inventory, int size) {

System.out.println("\nInventory Database:");

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

System.out.println("Item Code: " + inventory[i].itemCode +

", Item Price: $" + inventory[i].itemPrice);

}

}

private static void searchRecord(InventoryItem[] inventory, int size, int searchCode) {

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

if (inventory[i].itemCode == searchCode) {

System.out.println("Record found: " + inventory[i]);

return;

}

}

System.out.println("Record not found for item code " + searchCode);

}

private static void deleteRecord(InventoryItem[] inventory, int size, int deleteCode) {

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

if (inventory[i].itemCode == deleteCode) {

System.arraycopy(inventory, i + 1, inventory, i, size - i - 1);

System.out.println("Record deleted successfully.");

return;

}

}

System.out.println("Record not found for item code " + deleteCode);

}

private static void sortRecordsByPrice(InventoryItem[] inventory, int size) {

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

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

if (inventory[j].itemPrice > inventory[j + 1].itemPrice) {

InventoryItem temp = inventory[j];

inventory[j] = inventory[j + 1];

inventory[j + 1] = temp;

}

}

}

}

}

PRACTICAL 4 (create file data.txt insame folder)

Create database for employees of an organization using sequential file and perform following

operations: i. Add records ii. Display records iii. Search record iv. Modify record v. Delete record

import java.util.Scanner;

import java.io.File;

import java.io.FileWriter;

import java.io.FileNotFoundException;

import java.io.IOException;

class Employee {

int id;

String name;

String position;

double salary;

Employee(int id, String name, String position, double salary) {

this.id = id;

this.name = name;

this.position = position;

this.salary = salary;

}

Employee() {

this.id = 0;

this.name = "0";

this.position = "null";

this.salary = 0;

}

public void display() {

System.out.println("");

System.out.println("ID : " + id);

System.out.println("Name : " + name);

System.out.println("Position : " + position);

System.out.println("Salary : " + salary);

}

}

class Company {

Employee[] E;

int capacity;

int size = 0;

Company(int capacity) {

this.capacity = capacity;

E = new Employee[capacity];

}

public void getData() {

try {

File file = new File("data.txt");

Scanner rd = new Scanner(file);

int id;

String name;

String position;

double salary;

while (rd.hasNextLine()) {

id = Integer.parseInt(rd.nextLine());

name = rd.nextLine();

position = rd.nextLine();

salary = Double.parseDouble(rd.nextLine());

rd.nextLine();

E[size] = new Employee(id, name, position, salary);

size++;

}

rd.close();

} catch (FileNotFoundException e) {

System.out.println("File Not Found...");

}

}

public void setData(){

try {

FileWriter writer = new FileWriter("data.txt");

String data="";

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

data+=""+E[i].id+"\n";

data+=""+E[i].name+"\n";

data+=""+E[i].position+"\n";

data+=""+E[i].salary+"\n";

data+="##############\n";

}

writer.write(data);

writer.close();

} catch (IOException e) {

System.out.println("File Not Found....");

}

}

public void add(Employee e) {

E[size] = e;

size++;

}

public void display() {

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

E[i].display();

}

}

public int search(int id) {

int i = 0;

while (i < size) {

if (E[i].id == id) {

System.out.println("Found : ");

return (i);

}

i++;

}

System.out.println("Emplyee Not Found...");

;

return -1;

}

public void modify() {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the Id to be Modified : ");

int id = sc.nextInt();

int index = this.search(id);

if (index == -1) {

return;

}

System.out.println("Enter what u want to modify (1.id 2.name 3.position 4.salary) : ");

int mode = sc.nextInt();

sc.nextLine();

switch (mode) {

case 1:

System.out.print("Enter the Value : ");

int nId = sc.nextInt();

E[index].id = nId;

break;

case 2:

System.out.print("Enter the Value : ");

String nName = sc.nextLine();

E[index].name = nName;

break;

case 3:

System.out.print("Enter the Value : ");

String nPosition = sc.nextLine();

// sc.nextLine();

E[index].position = nPosition;

break;

case 4:

System.out.print("Enter the Value : ");

double nSalary = sc.nextDouble();

E[index].salary = nSalary;

break;

default:

System.out.println("Try Again");

break;

}

}

public void delete() {

Scanner sc = new Scanner(System.in);

System.out.print("Enter the Id to be Deleted : ");

int id = sc.nextInt();

int index = this.search(id);

if (index == -1) {

return;

}

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

E[i - 1] = E[i];

if (i + 1 == size) {

E[i] = null;

}

}

this.size--;

}

}

public class P4 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

Company com = new Company(100);

boolean end = false;

com.getData();

while (!end) {

System.out.println("");

System.out.println("Enter the Command : \n1.add\n2.display\n3.search\n4.modify\n5.delete\n6.Exit\n");

int inp = sc.nextInt();

sc.nextLine();

int id;

switch (inp) {

case 1:

System.out.print("Enter Employee ID: ");

id = sc.nextInt();

sc.nextLine();

System.out.print("Enter Employee Name: ");

String name = sc.nextLine();

System.out.print("Enter Employee Position: ");

String position = sc.nextLine();

System.out.print("Enter Employee Salary: ");

double salary = sc.nextDouble();

sc.nextLine(); // Consume the newline character

Employee e = new Employee(id, name, position, salary);

com.add(e);

break;

case 2:

com.display();

break;

case 3:

System.out.print("Enter Employee ID to Search: ");

id = sc.nextInt();

int index = com.search(id);

if (index != -1)

com.E[index].display();

break;

case 4:

com.modify();

break;

case 5:

com.delete();

break;

case 6:

end = true;

com.setData();

break;

default:

System.out.println("Wrong Input Try Again......");

break;

}

}

}

}

PRACTICAL 5

Sort the set of strings in ascending order using Insertion sort and descending order by using Selection

sort (Display pass by pass output) . Search a particular string using binary search with and without

recursion

import java.util.Scanner;

public class P5 {

static void print(String[] list) {

System.out.print("[ ");

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

if (i == list.length - 1) {

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

break;

}

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

}

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

}

static String[] getStrings() {

System.out.print("Enter the length of the List : ");

size = sc.nextInt();

sc.nextLine();

String[] list = new String[size];

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

System.out.print("Enter the string no." + (i + 1) + " : ");

list[i] = sc.nextLine();

}

return list;

}

static String[] asendingInsertion(String[] list) {

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

String key = list[i];

int j = i - 1;

int it = 0;

while (j >= 0 && list[j].compareTo(key) > 0) {

System.out.print("Iteration " + (it) + " : ");

print(list);

list[j + 1] = list[j];

j--;

it++;

}

list[j + 1] = key;

}

return list;

}

static String[] insertion(String[] list) {

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

String key = list[i];

int j = i - 1;

while (j >= 0 && list[j].compareTo(key) > 0) {

list[j + 1] = list[j];

j--;

}

list[j + 1] = key;

}

return list;

}

static String[] decendingSelection(String[] list) {

String temp;

int biggest;

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

biggest = i;

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

System.out.print("Iteration " + (i + j) + " : ");

print(list);

if (list[biggest].compareTo(list[j]) < 0) {

biggest = j;

}

}

temp = list[biggest];

list[biggest] = list[i];

list[i] = temp;

}

return list;

}

static int binary(String[] list, String str) {

list = insertion(list);

int left = 0;

int right = size - 1;

int mid;

while (left <= right) {

System.out.print("Iteration : ");

print(list);

mid = (left + right) / 2;

if (list[mid].equals(str)) {

return mid;

} else if (list[mid].compareTo(str) > 0) {

right = mid - 1;

} else if (list[mid].compareTo(str) < 0) {

left = mid + 1;

}

}

return -1;

}

static int binaryRecur(String[] list, String str, int left, int right) {

int mid = (left + right) / 2;

System.out.print("Iteration : ");

print(list);

if (right < left) {

return -1;

}

if (str.equals(list[mid])) {

return mid;

}

else if (list[mid].compareTo(str) > 0) {

return binaryRecur(list, str, left, mid - 1);

} else {

return binaryRecur(list, str, mid + 1, right);

}

}

static Scanner sc = new Scanner(System.in);

static int size;

public static void main(String[] args) {

String list[] = getStrings();

boolean end = false;

while(!end){

System.out.println("Enter the Operation \n1.Insertion Sort \n2.Selection Sort \n3.Binary Search \n4.Binary Search Recursion\n5.Exit");

int mode = sc.nextInt();

sc.nextLine();

String str;

switch (mode) {

case 1:

print(asendingInsertion(list));

break;

case 2:

print(decendingSelection(list));

break;

case 3:

System.out.print("Enter the String to be Searched : ");

str = sc.nextLine();

System.out.println("Index for "+str+" : "+binary(list,str));

break;

case 4:

System.out.print("Enter the String to be Searched : ");

str = sc.nextLine();

System.out.println("Index for "+str+" : "+binaryRecur(insertion(list),str,0,size-1));

break;

case 5:

end = true;

break;

default:

break;

}

}

}

}

PRACTICAL 6

a) Implement Quick Sort and Merge Sortto sort the given list of numbers. Display corresponding list

in each pass. (with recursion)

import java.util.Scanner;

public class P6 {

public static void print(int[] list) {

System.out.print("[ ");

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

if (i == list.length - 1) {

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

break;

}

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

}

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

}

public static int[] getList() {

System.out.print("Enter the length of the List : ");

size = sc.nextInt();

sc.nextLine();

int[] list = new int[size];

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

System.out.print("Enter the number at " + (i + 1) + " : ");

list[i] = sc.nextInt();

}

return list;

}

public static int partition(int[] list, int start, int end) {

int pivot = list[end];

int i = start - 1;

int j = start;

while (j < end) {

if (list[j] < pivot) {

i++;

int temp = list[j];

list[j] = list[i];

list[i] = temp;

}

j++;

}

i++;

list[end] = list[i];

list[i] = pivot;

System.out.print("Iterations : ");

print(list);

return i;

}

public static void quickSort(int[] list, int start, int end) {

if (start < end) {

int pivot = partition(list, start, end);

quickSort(list, start, pivot - 1);

quickSort(list, pivot + 1, end);

}

}

public static void mergeSort(int[] list, int start, int end) {

if (start >= end) {

return;

}

int mid = (start + end) / 2;

mergeSort(list, start, mid);

mergeSort(list, mid + 1, end);

merge(list, start, mid, end);

}

static void merge(int list[], int start, int mid, int end) {

int newList[] = new int[end - start + 1];

int idx = start;

int idy = mid + 1;

int i = 0;

while (idx <= mid && idy <= end) {

if (list[idx] <= list[idy]) {

newList[i] = list[idx];

idx++;

} else {

newList[i] = list[idy];

idy++;

}

i++;

}

while (idx <= mid) {

newList[i] = list[idx];

idx++;

i++;

}

while (idy <= end) {

newList[i] = list[idy];

idy++;

i++;

}

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

list[start + i] = newList[i];

}

System.out.print("Iterations : ");

print(list);

}

static Scanner sc = new Scanner(System.in);

static int size;

public static void main(String[] args) {

int[] list = getList();

int[] mList = list;

int[] qList = list;

boolean end = false;

while (!end) {

System.out.println("Enter the Operation \n1.Quick Sort \n2.Merge Sort\n3.Exit");

int mode = sc.nextInt();

sc.nextLine();

switch (mode) {

case 1:

quickSort(qList, 0, size - 1);

System.out.print("Sorted List : ");

print(qList);

break;

case 2:

mergeSort(mList, 0, size - 1);

System.out.print("Sorted List : ");

print(mList);

break;

case 3:

end = true;

break;

default:

break;

}

}

}

}

PRACTICAL 7

Accept conventional matrix and convert it into sparse matrix using structure and Perform addition of

two sparse matrices. Implement simple and fast transpose algorithms on sparse matrix.

import java.util.Scanner;

class SparseMat {

public int[][] mat;

public int row, column;

public int size;

SparseMat() {

}

SparseMat(int[][] m) {

size = 0;

row = m.length;

column = m[0].length;

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

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

if (m[i][j] != 0) {

size++;

}

}

}

this.size = size;

this.mat = new int[size][3];

int x = 0;

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

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

if (m[i][j] != 0) {

mat[x][0] = i;

mat[x][1] = j;

mat[x][2] = m[i][j];

x++;

}

}

}

}

public SparseMat simpleTranspose(){

SparseMat mat = new SparseMat();

mat.row = row;

mat.column = column;

mat.size = size;

mat.mat = new int[size][3];

int k = 0, counter = 0;

while (k < column) {

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

if (this.mat[i][1] == k) {

mat.mat[counter][0] = this.mat[i][1];

mat.mat[counter][1] = this.mat[i][0];

mat.mat[counter][2] = this.mat[i][2];

counter++;

}

}

k++;

}

return mat;

}

public SparseMat fastTranspose(){

SparseMat mat = new SparseMat();

mat.row = row;

mat.column = column;

mat.size = size;

mat.mat = new int[size][3];

int[] total = new int[column];

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

total[this.mat[i][1]]+=1;

}

int[] index = new int[total.length];

index[0] = 1;

for(int i=1;i<index.length;i++){

index[i] = index[i-1] + total[i-1];

}

int loc;

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

loc = index[this.mat[i][1]];

mat.mat[loc-1][0] = this.mat[i][1];

mat.mat[loc-1][1] = this.mat[i][0];

mat.mat[loc-1][2] = this.mat[i][2];

index[this.mat[i][1]] ++;

}

return mat;

}

public void print() {

System.out.println("\nR C V");

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

for (int j = 0; j < 3; j++) {

System.out.printf(mat[i][j] + " ");

}

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

}

System.out.println("");

}

}

public class P7 {

public static SparseMat add(SparseMat m1, SparseMat m2) {

SparseMat mat = new SparseMat();

mat.row = m1.row;

mat.column = m1.column;

mat.mat = new int[100][3];

int x = 0, y = 0, z = 0;

mat.size = 0;

while (x < m1.size && y < m2.size) {

if (m1.mat[x][0] == m2.mat[y][0] && m1.mat[x][1] == m2.mat[y][1]) {

mat.mat[z][0] = m1.mat[x][0];

mat.mat[z][1] = m1.mat[x][1];

mat.mat[z][2] = m1.mat[x][2] + m2.mat[y][2];

mat.size++;

x++;

y++;

z++;

} else if (m1.mat[x][0] == m2.mat[y][0]) {

if (m1.mat[x][1] < m2.mat[y][1]) {

mat.mat[z][0] = m1.mat[x][0];

mat.mat[z][1] = m1.mat[x][1];

mat.mat[z][2] = m1.mat[x][2];

mat.size++;

x++;

z++;

} else {

mat.mat[z][0] = m2.mat[y][0];

mat.mat[z][1] = m2.mat[y][1];

mat.mat[z][2] = m2.mat[y][2];

mat.size++;

y++;

z++;

}

}

else if (m1.mat[x][0] > m2.mat[y][0]) {

mat.mat[z][0] = m2.mat[y][0];

mat.mat[z][1] = m2.mat[y][1];

mat.mat[z][2] = m2.mat[y][2];

mat.size++;

y++;

z++;

} else {

mat.mat[z][0] = m1.mat[x][0];

mat.mat[z][1] = m1.mat[x][1];

mat.mat[z][2] = m1.mat[x][2];

mat.size++;

z++;

x++;

}

}

return mat;

}

public static void printMat(int[][] mat) {

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

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

System.out.print("[ ");

for (int j = 0; j < mat[0].length; j++) {

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

}

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

}

}

public static int[][] getMat() {

System.out.print("Enter the no. of Rows : ");

int row = sc.nextInt();

System.out.print("Enter the no. of Columns : ");

int column = sc.nextInt();

int mat[][] = new int[row][column];

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

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

System.out.print("Enter the For (" + (i + 1) + " x " + (j + 1) + " ) : ");

mat[i][j] = sc.nextInt();

}

}

return mat;

}

static Scanner sc = new Scanner(System.in);

public static void main(String args[]) {

System.out.println("Enter for Mat1 : ");

int[][] mat1 = getMat();

System.out.println("Enter for Mat2 : ");

int[][] mat2 = getMat();

printMat(mat1);

printMat(mat2);

SparseMat m1 = new SparseMat(mat1);

SparseMat m2 = new SparseMat(mat2);

m1.print();

m2.print();

boolean end = false;

while (!end) {

System.out.println("Enter the Operation \n1.add \n2.simple transpose\n3.fast Transpose\n4.Exit");

int mode = sc.nextInt();

sc.nextLine();

switch (mode) {

case 1:

(add(m1, m2)).print();

break;

case 2:

m1.simpleTranspose().print();

break;

case 3:

m1.fastTranspose().print();

break;

case 4:

end = true;

break;

default:

break;

}

}

}

}

PRACTICAL 8

Implement a singly linked list(SLL) as an ADT

class SLL {

Node head;

private int size;

SLL() {

size = 0;

}

public class Node {

String data;

Node next;

Node(String data) {

this.data = data;

this.next = null;

size++;

}

}

public void addFirst(String data) {

Node newNode = new Node(data);

newNode.next = head;

head = newNode;

}

public void addLast(String data) {

Node newNode = new Node(data);

if (head == null) {

head = newNode;

return;

}

Node lastNode = head;

while (lastNode.next != null) {

lastNode = lastNode.next;

}

lastNode.next = newNode;

}

public void printList() {

Node currNode = head;

while (currNode != null) {

System.out.print(currNode.data + " -> ");

currNode = currNode.next;

}

System.out.println("null");

}

public void removeFirst() {

if (head == null) {

System.out.println("Empty List, nothing to delete");

return;

}

head = this.head.next;

size--;

}

public void removeLast() {

if (head == null) {

System.out.println("Empty List, nothing to delete");

return;

}

size--;

if (head.next == null) {

head = null;

return;

}

Node secondLast = head;

Node lastNode = head.next;

while (lastNode.next != null) {

secondLast = secondLast.next;

lastNode = lastNode.next;

}

secondLast.next = null;

}

public int getSize() {

return size;

}

public static void main(String args[]) {

SLL list = new SLL();

list.addLast("is");

list.addLast("a");

list.addLast("list");

list.printList();

list.addFirst("this");

list.printList();

System.out.println(list.getSize());

list.removeFirst();

list.printList();

list.removeLast();

list.printList();

}

}

USING COLLECTION FRAMEWORK

import java.util.\*;

class SLLCF {

public static void main(String args[]) {

LinkedList<String> list = new LinkedList<String>();

list.add("is");

list.add("a");

list.addLast("list");

list.addFirst("this");

list.add(3, "linked");

System.out.println(list);

System.out.println(list.get(0));

System.out.println(list.size());

list.remove(3);

list.removeFirst();

list.removeLast();

System.out.println(list);

}

}

PRACTICAL 9

Implement any database using doubly linked list (DLL) and perform database operations.

public class DoublyLL {

public static class Node {

int val;

Node next;

Node prev;

Node(int val) {

this.val = val;

}

}

public static void display (Node head) {

Node temp = head;

while (temp != null) {

System.out.print(temp.val + " ");

temp = temp.next;

}

System.out.println();

}

public static void displayrev (Node tail) {

Node temp = tail;

while (temp != null) {

System.out.print(temp.val + " ");

temp = temp.prev;

}

System.out.println();

}

public static void main(String args[]) {

// 4 10 2 99 13

Node a = new Node(4);

Node b = new Node(10);

Node c = new Node(2);

Node d = new Node(99);

Node e = new Node(13);

a.prev = null;

a.next = b;

b.prev = a;

b.next = c;

c.prev = b;

c.next = d;

d.prev = c;

d.next = e;

e.prev = d;

e.next = null;

display(a);

displayrev(e);

}

}

PRACTICAL 10

Implement stack ADT using linked list. Write a program for infix to postfix conversion using stack.

Also, evaluate the postfix expression.

import java.util.Scanner;

import java.util.Stack;

public class InfixToPostfixEvaluator {

static boolean isOperator(char c) {

return (c == '+' || c == '-' || c == '\*' || c == '/');

}

static int getPrecedence(char operator) {

switch (operator) {

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

}

return -1;

}

static String infixToPostfix(String infixExpression) {

StringBuilder postfix = new StringBuilder();

Stack<Character> operatorStack = new Stack<>();

for (char c : infixExpression.toCharArray()) {

if (Character.isLetterOrDigit(c)) {

postfix.append(c);

} else if (c == '(') {

operatorStack.push(c);

} else if (c == ')') {

while (!operatorStack.isEmpty() && operatorStack.peek() != '(') {

postfix.append(operatorStack.pop());

}

operatorStack.pop(); // Pop the '('

} else if (isOperator(c)) {

while (!operatorStack.isEmpty() && getPrecedence(c) <= getPrecedence(operatorStack.peek())) {

postfix.append(operatorStack.pop());

}

operatorStack.push(c);

}

}

while (!operatorStack.isEmpty()) {

postfix.append(operatorStack.pop());

}

return postfix.toString();

}

static int evaluatePostfix(String postfixExpression) {

Stack<Integer> operandStack = new Stack<>();

for (char c : postfixExpression.toCharArray()) {

if (Character.isDigit(c)) {

operandStack.push(Character.getNumericValue(c));

} else if (isOperator(c)) {

int operand2 = operandStack.pop();

int operand1 = operandStack.pop();

switch (c) {

case '+':

operandStack.push(operand1 + operand2);

break;

case '-':

operandStack.push(operand1 - operand2);

break;

case '\*':

operandStack.push(operand1 \* operand2);

break;

case '/':

operandStack.push(operand1 / operand2);

break;

}

}

}

return operandStack.pop();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter infix expression: ");

String infixExpression = scanner.nextLine();

scanner.close();

String postfixExpression = infixToPostfix(infixExpression);

System.out.println("Infix Expression: " + infixExpression);

System.out.println("Postfix Expression: " + postfixExpression);

int result = evaluatePostfix(postfixExpression);

System.out.println("Result after evaluation: " + result);

}

}

PRACTICAL 11

Implement the circular queue.

import java.util.Scanner;

public class CircularQueue {

private int[] queue;

private int front;

private int rear;

private int size;

// constructor to create a CQ by passing the size as the parameter

public CircularQueue(int capacity) {

size = capacity + 1; // One extra space to distinguish between full and empty states

queue = new int[size];

front = rear = 0;

}

//if front = rear then returns true

public boolean isEmpty() {

return front == rear;

}

//if (rear+1)%size = front then returns true

// modulo operator checks if the queue wraps around the array

public boolean isFull() {

return (rear + 1) % size == front;

}

//ENQUEUE

public void enqueue(int element) {

//if

//queue is full then print Queue Full

if (isFull()) {

System.out.println("Queue is full. Cannot enqueue.");

}

//else

//add the element at rear which is queue[rear]

else {

queue[rear] = element;

rear = (rear + 1) % size; //update the rear by rear = (rear+1)%size

System.out.println(element + " enqueued successfully.");

}

}

//DEQUEUE

public void dequeue() {

//if

//the queue is empty then print queue is empty

if (isEmpty()) {

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

}

//else

else {

int dequeuedElement = queue[front]; // take the first element of queue and put it into a varible

front = (front + 1) % size; //increase the pointer of the front

System.out.println(dequeuedElement + " dequeued successfully.");//print thr dequeued element

}

}

public void display() {

//if queue is empty then print the queue is empty

if (isEmpty()) {

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

}

else {

System.out.print("Elements in the circular queue: ");

//iterate through the elements till the front is not equal to rear

int i = front;

while (i != rear) {

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

i = (i + 1) % size; //increase the pointer we have taken this modulo op so that whenever the pointer reaches end of queue so after incrementing the pointer it will perform the modulo operation and the remainder will be 0 so the pointer will point back to the 0th index position which is 0.

}

System.out.println();

}

}

public static void main(String[] args) {

//create the object of scnner class

Scanner sc = new Scanner(System.in);

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

//take the size in capacity vaiable by scanning

int capacity = sc.nextInt();

//create obj of circular queue by passing capacity in parameter

CircularQueue cq = new CircularQueue(capacity);

while (true) {

System.out.println("Choose an option:");

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

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

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

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

//scan and store the user input in choice variable

int choice = sc.nextInt();

switch (choice) {

case 1:

System.out.println("Enter the element to enqueue:");

int enqueueElement = sc.nextInt();

cq.enqueue(enqueueElement);

break;

case 2:

cq.dequeue();

break;

case 3:

cq.display();

break;

case 4:

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

System.exit(0);

default:

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

}

}

}

}

PRACTICAL 12

Implement a priority Queue for a list of patients using linked list.

/\*\*

\* Prac12 implement a priority Queue for a list of patients using linked list.

\*/

import java.util.PriorityQueue;

import java.util.Scanner;

public class Prac12 {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

// Creating a Priority Queue

PriorityQueue<Integer> priorityQueue = new PriorityQueue<>();

while (true) {

System.out.println("Choose an option:");

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

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

System.out.println("3. Print elements");

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

int choice = sc.nextInt();

switch (choice) {

case 1:

System.out.println("Enter the element to enqueue:");

int enqueueElement = sc.nextInt();

priorityQueue.offer(enqueueElement);

System.out.println(enqueueElement + " enqueued successfully.");

break;

case 2:

if (!priorityQueue.isEmpty()) {

int dequeuedElement = priorityQueue.poll();

System.out.println(dequeuedElement + " dequeued successfully.");

} else {

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

}

break;

case 3:

System.out.println("Elements in the priority queue: " + priorityQueue);

break;

case 4:

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

System.exit(0);

default:

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

}

}

}

}