

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | |
|  | | DSA PROJECT | | | | |  | |
|  |  | | | | | | |  |
|  | | | |  |  | | | |
|  | | | | Muhammad Ali Zafar Sp21-bai-016 |  | | | |
|  | | | | 9th January 2023—Data Structures and Algorithm—Sir Rashid Mehmood Rafiq |  | | | |
|  | | |  | | |  | | |
|  | | |  | | |  | | |

**Problem Statement**

**You are required to design an application (console application) which provides its users to plan a journey between two cities of his/her choice and displaysthe available options of different routes from point A to point B, the services of different Bus Operators etc.The following are the details of minimum requirements that your project should meet to qualify.**

**a)City –You can select at least 60 cities (district headquarters) of Pakistan from all four provinces. You need to store the name of city only.**

**b)Road Network –the information of roads between two cities to be stored.**

**c)Bus Operators –there must be at least 5 different bus operators which operators on different routes, some of them may use same routes for example Islamabad to Lahore is operated by most Bus Operators.**

**d)Routes –The routes are the journeys between different cities/districts and operated by different bus operators.**

**1.Introduction**

The purpose of this project is to design an application that allows users to plan a journey between two cities and display the available options of different routes, as well as the services of different bus operators. The application is a console application that is able to store and retrieve information about cities, roads, bus operators, and routes. The user will be able to input the source and destination cities, and the application will output the available routes between those two cities, including the names of the bus operators that operate those routes. The application will store information about at least 60 cities in Pakistan, the roads between those cities, at least 5 different bus operators, and the routes operated by those bus operators. This application will provide a convenient way for users to plan their journeys and make informed decisions about their travel plans.

**2.Team Members and their Contribution**

**3.Block Diagram / Flow Chart**

FLOW CHART:

**START**

**INPUT:** City (60 cities from all four provinces of Pakistan)

**STORE:** City Name

**INPUT:** Road Network (between two cities)

**STORE:** Road Network

**INPUT:** Bus Operators (at least 5 different)

**STORE:** Bus Operators

**INPUT:** Routes (Journeys between cities/districts)

**STORE:** Routes

**DISPLAY:** Available Options of Different Routes from Point A to Point B

**DISPLAY:** Services of Different Bus Operators

**END**

**START**

Input City 1

Input City 2

Search for Road Network between City 1 & City 2

IF Road Network Found

Search for Bus Operators

IF Bus Operators Found

Search for Routes and Services

IF Routes & Services Found

Search for Fares

IF Fares Found

Display All Options

ELSE

Display Error Message

**END**

**4.Explanation of the Algorithm/Data Structure**

One possible algorithm for this application is to use a graph data structure to represent the roads between cities. The graph can be implemented using an adjacency list, where each vertex in the graph represents a city and each edge represents a road between two cities. The adjacency list can be implemented using a hash table, where the keys are the cities and the values are lists of adjacent cities.

When the user inputs the source and destination cities, a breadth-first search (BFS) algorithm can be used to find all the paths from the source city to the destination city. The BFS algorithm starts at the source city and explores all the neighboring cities, then moves to the next level of cities, and so on, until it reaches the destination city or runs out of cities to explore. The routes can then be retrieved from the list of paths found by the BFS algorithm.

1. Create a data structure to store the names of cities and the roads between them. This can be done by using a two dimensional array, where each row represents a city and each column represents the roads between cities.

2. Create a data structure to store the details of bus operators. This can be done by using a linked list, where each node represents a bus operator and contains their name, routes and other associated information.

3. Create a function to take input from the user for the source and destination cities. This can be done by using a simple prompt that asks the user to enter two cities.

4. Create a function to search for routes between the two cities. This can be done by using a breadth-first search algorithm, which will search through the two-dimensional array and find all possible routes between the two cities.

5. Create a function to search for available bus operators on the routes. This can be done by using a linear search algorithm, which will search through the linked list and find all available bus operators on the routes.

6. Create a function to display the available options of different routes from point A to point B, the services of different Bus Operators etc. This can be done by simply printing out the results of the search algorithms.

**5.C++ Codeof the Algorithm / Data Structure**

#include <iostream>

#include <string>

#include <vector>

#include <queue>

#include <map>

#include <algorithm>

using namespace std;

class City

{

public:

    string name;

    City(string name)

    {

        this->name = name;

    }

};

class Road

{

public:

    City \*from;

    City \*to;

    Road(City \*from, City \*to)

    {

        this->from = from;

        this->to = to;

    }

};

class BusOperator

{

public:

    string name;

    BusOperator(string name)

    {

        this->name = name;

    }

};

class Route

{

public:

    BusOperator \*busOperator;

    vector<Road \*> roads;

    Route(BusOperator \*busOperator)

    {

        this->busOperator = busOperator;

    }

    void addRoad(Road \*road)

    {

        roads.push\_back(road);

    }

};

class JourneyPlanner

{

public:

    vector<City \*> cities;

    vector<Road \*> roads;

    vector<BusOperator \*> busOperators;

    vector<Route \*> routes;

    void addCity(City \*city)

    {

        cities.push\_back(city);

    }

    void addRoad(Road \*road)

    {

        roads.push\_back(road);

    }

    void addBusOperator(BusOperator \*busOperator)

    {

        busOperators.push\_back(busOperator);

    }

    void addRoute(Route \*route)

    {

        routes.push\_back(route);

    }

    void planJourney(City \*from, City \*to, int distance)

    {

        // BFS Algorithm

        // Create a queue for BFS

        queue<City \*> q;

        // Create a vector to store visited cities

        vector<City \*> visited;

        // Create a map to store the routes

        map<City \*, City \*> parent;

        // Push the starting city to the queue

        q.push(from);

        // Mark the starting city as visited

        visited.push\_back(from);

        // Loop until the queue is empty

        while (!q.empty())

        {

            // Pop the top element from the queue

            City \*current = q.front();

            q.pop();

            // If the current city is the destination city, break the loop

            if (current == to)

            {

                break;

            }

            // Loop through all the roads from the current city

            for (int i = 0; i < roads.size(); i++)

            {

                // Check if the current city is the starting city of the road

                if (roads[i]->from == current)

                {

                    // Get the destination city of the road

                    City \*nextCity = roads[i]->to;

                    // Check if the destination city has already been visited

                    bool isVisited = false;

                    for (int j = 0; j < visited.size(); j++)

                    {

                        if (visited[j] == nextCity)

                        {

                            isVisited = true;

                            break;

                        }

                    }

                    // If the destination city has not been visited, mark it as visited and push it to the queue

                    if (!isVisited)

                    {

                        visited.push\_back(nextCity);

                        q.push(nextCity);

                        // Store the city and its parent in the map

                        parent[nextCity] = current;

                    }

                }

            }

        }

        // Print the route

        cout << "Route from " << from->name << " to " << to->name <<endl;

        cout << "Distance is :" << distance << "km" <<endl;

        // Retrieve the route

        vector<City \*> route;

        City \*current = to;

        while (current != from)

        {

            route.push\_back(current);

            current = parent[current];

        }

        route.push\_back(from);

        // Reverse the route

        reverse(route.begin(), route.end());

        // Print the route

        for (int i = 0; i < route.size(); i++)

        {

            cout << route[i]->name;

            if (i < route.size() - 1)

            {

                cout << " -> ";

            }

        }

        cout << endl;

        // Print the available bus operators

        cout << "Available bus operators:" << endl;

        for (int i = 0; i < routes.size(); i++)

        {

            bool isAvailable = true;

            // Check if the route covers all the cities in the route

            for (int j = 0; j < route.size(); j++)

            {

                bool isCovered = false;

                for (int k = 0; k < routes[i]->roads.size(); k++)

                {

                    if (route[j] == routes[i]->roads[k]->from || route[j] == routes[i]->roads[k]->to)

                    {

                        isCovered = true;

                        break;

                    }

                }

                if (!isCovered)

                {

                    isAvailable = false;

                    break;

                }

            }

            // If the route is available, print the bus operator

            if (isAvailable)

            {

                cout << routes[i]->busOperator->name << endl;

            }

        }

    }

};

int main()

{

    // Create the cities

    City \*islamabad = new City("Islamabad");

    City \*lahore = new City("Lahore");

    City \*karachi = new City("Karachi");

    City \*peshawar = new City("Peshawar");

    City \*quetta = new City("Quetta");

    City \*multan = new City("Multan");

    City \*faisalabad = new City("Faisalabad");

    City \*rawalpindi = new City("Rawalpindi");

    City \*hyderabad = new City("Hyderabad");

    City \*sialkot = new City("Sialkot");

    City \*gujranwala = new City("Gujranwala");

    City \*bahawalpur = new City("Bahawalpur");

    City \*sargodha = new City("Sargodha");

    City \*jhelum = new City("Jhelum");

    City \*mardan = new City("Mardan");

    City \*sheikhupura = new City("Sheikhupura");

    City \*kohat = new City("Kohat");

    City \*abbottabad = new City("Abbottabad");

    City \*mingora = new City("Mingora");

    City \*bahawalnagar = new City("Bahawalnagar");

    City \*larkana = new City("Larkana");

    City \*sukkur = new City("Sukkur");

    City \*mirpurKhas = new City("MirpurKhas");

    City \*jacobabad = new City("Jacobabad");

    City \*shikarpur = new City("Shikarpur");

    City \*kasur = new City("Kasur");

    City \*deraGhaziKhan= new City("DeraGhaziKhan");

    City \*khanewal = new City("Khanewal");

    City \*gujrat = new City("Gujrat");

    City \*muzaffarabad = new City("Muzaffarabad");

    City \*sahiwal = new City("Sahiwal");

    City \*okara = new City("Okara");

    City \*wahCantt = new City("WahCantt");

    City \*chiniot = new City("Chiniot");

    City \*kamoke = new City("Kamoke");

    City \*hafizabad = new City("Hafizabad");

    City \*khushab = new City("Khushab");

    City \*jhang = new City("Jhang");

    City \*mandiBahauddin = new City("MandiBahauddin");

    City \*murree = new City("Murree");

    City \*attock = new City("Attock");

    City \*kundian = new City("Kundian");

    City \*nowshera = new City("Nowshera");

    City \*gojra = new City("Gojra");

    City \*mansehra = new City("Mansehra");

    City \*chaman = new City("Chaman");

    City \*bannu = new City("Bannu");

    City \*daska = new City("Daska");

    City \*charsadda = new City("Charsadda");

    City \*tandoAdam = new City("TandoAdam");

    City \*karak = new City("Karak");

    City \*timergara = new City("Timergara");

    City \*kohlu = new City("Kohlu");

    City \*deraIsmailKhan = new City("DeraIsmailKhan");

    City \*khuzdar = new City("Khuzdar");

    City \*turbat = new City("Turbat");

    City \*swabi = new City("Swabi");

    City \*mianwali = new City("Mianwali");

    City \*chitral = new City("Chitral");

    City \*gawadar = new City("Gawadar");

    City \*nasirabad = new City("Nasirabad");

    City \*taxila = new City("taxila");

    City \*jauharabad = new City("Jauharabad");

    City \*shahdadkot = new City("Shahdadkot");

    City \*skardu = new City("Skardu");

    City \*nankanaSahib = new City("NankanaSahib");

    City \*khanpur = new City("Khanpur");

    City \*alang = new City("alang");

    // Link between 60 cities

    Road \*islamabadToLahore = new Road(islamabad, lahore);

    Road \*islamabadToPeshawar = new Road(islamabad, peshawar);

    Road \*peshawarToKohat = new Road(peshawar, kohat);

    Road \*lahoreToFaisalabad = new Road(lahore, faisalabad);

    Road \*faisalabadToMultan = new Road(faisalabad, multan);

    Road \*multanToHyderabad = new Road(multan, hyderabad);

    Road \*hyderabadToKarachi = new Road(hyderabad, karachi);

    Road \*abbottabadToKhanpur = new Road(abbottabad, khanpur);

    Road \*khanpurToTaxila = new Road(khanpur, taxila);

    Road \*taxilaToIslamabad = new Road(taxila, islamabad);

    Road \*islamabadToDeraIsmailKhan = new Road(islamabad, deraIsmailKhan);

    Road \*deraIsmailKhanToAlang = new Road(deraIsmailKhan, alang);

    Road \*peshawarToMardan= new Road(peshawar, mardan);

    Road \*alangToQuetta = new Road(alang, quetta);

    Road \*kohatToAbbottabad = new Road(kohat, abbottabad);

    Road \*quettaToKarachi = new Road(quetta, karachi);

    Road \*multanToQuetta = new Road(multan, quetta);

    Road \*bahawalnagarToLarkana = new Road(bahawalnagar, larkana);

    Road \*larkanaToSukkur = new Road(larkana, sukkur);

    Road \*sukkurToMirpurKhas = new Road(sukkur, mirpurKhas);

    Road \*mirpurKhasToJacobabad = new Road(mirpurKhas, jacobabad);

    Road \*jacobabadToShikarpur = new Road(jacobabad, shikarpur);

    Road \*shikarpurToKasur = new Road(shikarpur, kasur);

    Road \*kasurToDeraIsmailKhan = new Road(kasur, deraIsmailKhan);

    Road \*deraIsmailKhanToKhanewal = new Road(deraIsmailKhan, khanewal);

    Road \*khanewalToGujrat = new Road(khanewal, gujrat);

    Road \*gujratToMuzaffarabad = new Road(gujrat, muzaffarabad);

    Road \*muzaffarabadToSahiwal = new Road(muzaffarabad, sahiwal);

    Road \*lahoreToOkara = new Road(lahore, okara);

    Road \*okaraToWahCantt = new Road(okara, wahCantt);

    Road \*wahCanttToChiniot = new Road(wahCantt, chiniot);

    Road \*chiniotToKamoke = new Road(chiniot, kamoke);

    Road \*kamokeToHafizabad = new Road(kamoke, hafizabad);

    Road \*hafizabadToKhushab = new Road(hafizabad, khushab);

    Road \*khushabToJhang = new Road(khushab, jhang);

    Road \*jhangToMandiBahauddin = new Road(jhang, mandiBahauddin);

    Road \*mandiBahauddinToMurree = new Road(mandiBahauddin, murree);

    Road \*murreeToAttock = new Road(murree, attock);

    Road \*attockToKundian = new Road(attock, kundian);

    Road \*kundianToNowshera = new Road(kundian, nowshera);

    Road \*nowsheraToGojra = new Road(nowshera, gojra);

    Road \*gojraToMansehra = new Road(gojra, mansehra);

    Road \*mansehraToChaman = new Road(mansehra, chaman);

    Road \*chamanToBannu = new Road(chaman, bannu);

    Road \*bannuToDaska = new Road(bannu, daska);

    Road \*daskaToCharsadda = new Road(daska, charsadda);

    Road \*charsaddaToTandoAdam = new Road(charsadda, tandoAdam);

    Road \*tandoAdamToKarak = new Road(tandoAdam, karak);

    Road \*karakToTimergara = new Road(karak, timergara);

    Road \*timergaraToKohlu = new Road(timergara, kohlu);

    Road \*kohluToDeraIsmailKhan = new Road(kohlu, deraIsmailKhan);

    Road \*deraIsmailKhanToKhuzdar = new Road(deraIsmailKhan, khuzdar);

    Road \*khuzdarToTurbat = new Road(khuzdar, turbat);

    Road \*turbatToSwabi = new Road(turbat, swabi);

    Road \*swabiToMianwali = new Road(swabi, mianwali);

    Road \*mianwaliToChitral = new Road(mianwali, chitral);

    Road \*chitralToGawadar = new Road(chitral, gawadar);

    Road \*gawadarToNasirabad = new Road(gawadar, nasirabad);

    Road \*nasirabadToTaxila = new Road(nasirabad,taxila);

    Road \*taxilaToJauharabad = new Road(taxila, jauharabad);

    Road \*jauharabadToShahdadkot = new Road(jauharabad, shahdadkot);

    Road \*shahdadkotToSkardu = new Road(shahdadkot, skardu);

    Road \*skarduToNankanaSahib = new Road(skardu, nankanaSahib);

    Road \*nankanaSahibToKhairpur = new Road(nankanaSahib, khanpur);

    Road \*khairpurToalang = new Road(khanpur, alang);

    // Create the bus operators

    BusOperator \*operator1 = new BusOperator("Daewoo Express");

    BusOperator \*operator2 = new BusOperator("Skyways");

    BusOperator \*operator3 = new BusOperator("Waraich Express:");

    BusOperator \*operator4 = new BusOperator("Faisal Movers");

    BusOperator \*operator5 = new BusOperator("Koshitan Express:");

    // Create the routes

    Route \*route1 = new Route(operator1);

    route1->addRoad(islamabadToLahore);

    route1->addRoad(peshawarToKohat);

    route1->addRoad(peshawarToMardan);

    route1->addRoad(faisalabadToMultan);

    route1->addRoad(lahoreToFaisalabad);

    route1->addRoad(multanToHyderabad);

    Route \*route2 = new Route(operator2);

    route2->addRoad(islamabadToLahore);

    route2->addRoad(multanToHyderabad);

    route2->addRoad(hyderabadToKarachi);

    route2->addRoad(multanToQuetta);

    route2->addRoad(faisalabadToMultan);

    route2->addRoad(lahoreToFaisalabad);

    route2->addRoad(lahoreToOkara);

    Route \*route3 = new Route(operator3);

    route3->addRoad(kohatToAbbottabad);

    route3->addRoad(taxilaToIslamabad);

    route3->addRoad(abbottabadToKhanpur);

    route3->addRoad(islamabadToDeraIsmailKhan);

    route3->addRoad(deraIsmailKhanToAlang);

    route3->addRoad(abbottabadToKhanpur);

    route3->addRoad(hyderabadToKarachi);

    route3->addRoad(islamabadToLahore);

    route3->addRoad(lahoreToFaisalabad);

    route3->addRoad(faisalabadToMultan);

    route3->addRoad(multanToHyderabad);

    route3->addRoad(islamabadToDeraIsmailKhan);

    route3->addRoad(deraIsmailKhanToAlang);

    route3->addRoad(alangToQuetta);

    route3->addRoad(quettaToKarachi);

    Route \*route4 = new Route(operator4);

    route4->addRoad(islamabadToLahore);

    route4->addRoad(peshawarToMardan);

    route4->addRoad(lahoreToFaisalabad);

    route4->addRoad(islamabadToDeraIsmailKhan);

    route4->addRoad(deraIsmailKhanToAlang);

    route4->addRoad(alangToQuetta);

    route4->addRoad(quettaToKarachi);

    Route \*route5 = new Route(operator5);

    route5->addRoad(islamabadToLahore);

    route5->addRoad(khanpurToTaxila);

    route5->addRoad(islamabadToDeraIsmailKhan);

    route5->addRoad(deraIsmailKhanToAlang);

    route5->addRoad(alangToQuetta);

    route5->addRoad(quettaToKarachi);

    // Create the journey planner

    JourneyPlanner \*journeyPlanner = new JourneyPlanner();

    // Add the cities

    journeyPlanner->addCity(islamabad);

    journeyPlanner->addCity(lahore);

    journeyPlanner->addCity(karachi);

    journeyPlanner->addCity(bahawalpur);

    // Add the roads

    journeyPlanner->addRoad(islamabadToLahore);

    journeyPlanner->addRoad(hyderabadToKarachi);

    journeyPlanner->addRoad(islamabadToPeshawar);

    journeyPlanner->addRoad(peshawarToMardan);

    journeyPlanner->addRoad(lahoreToFaisalabad);

    journeyPlanner->addRoad(faisalabadToMultan);

    journeyPlanner->addRoad(lahoreToFaisalabad);

    journeyPlanner->addRoad(multanToHyderabad);

    journeyPlanner->addRoad(hyderabadToKarachi);

    journeyPlanner->addRoad(abbottabadToKhanpur);

    journeyPlanner->addRoad(khanpurToTaxila);

    journeyPlanner->addRoad(taxilaToIslamabad);

    journeyPlanner->addRoad(islamabadToDeraIsmailKhan);

    journeyPlanner->addRoad(deraIsmailKhanToAlang);

    journeyPlanner->addRoad(peshawarToKohat);

    journeyPlanner->addRoad(quettaToKarachi);

    journeyPlanner->addRoad(alangToQuetta);

    journeyPlanner->addRoad(multanToQuetta);

    journeyPlanner->addRoad(lahoreToOkara);

    // Add the bus operators

    journeyPlanner->addBusOperator(operator1);

    journeyPlanner->addBusOperator(operator2);

    journeyPlanner->addBusOperator(operator3);

    journeyPlanner->addBusOperator(operator4);

    journeyPlanner->addBusOperator(operator5);

    // Add the routes

    journeyPlanner->addRoute(route1);

    journeyPlanner->addRoute(route2);

    journeyPlanner->addRoute(route3);

    journeyPlanner->addRoute(route4);

    journeyPlanner->addRoute(route5);

    // Plan the journey

    journeyPlanner->planJourney(islamabad, lahore, 312);

    journeyPlanner->planJourney(islamabad, multan, 537);

    journeyPlanner->planJourney(islamabad, karachi, 1410);

    journeyPlanner->planJourney(islamabad, faisalabad, 317);

    journeyPlanner->planJourney(islamabad, hyderabad, 1224);

    journeyPlanner->planJourney(abbottabad, islamabad, 108);

    journeyPlanner->planJourney(islamabad, quetta, 894);

    journeyPlanner->planJourney(lahore, karachi, 1211);

    journeyPlanner->planJourney(deraIsmailKhan, karachi , 1238);

    journeyPlanner->planJourney(faisalabad, quetta , 825);

    journeyPlanner->planJourney(lahore, okara , 129);

    return 0;

}

**6.Test Data Description**

In this data set 60 cities are present from four provinces.

City \*islamabad = new City("Islamabad");

    City \*lahore = new City("Lahore");

    City \*karachi = new City("Karachi");

    City \*peshawar = new City("Peshawar");

    City \*quetta = new City("Quetta");

    City \*multan = new City("Multan");

    City \*faisalabad = new City("Faisalabad");

    City \*rawalpindi = new City("Rawalpindi");

    City \*hyderabad = new City("Hyderabad");

    City \*sialkot = new City("Sialkot");

    City \*gujranwala = new City("Gujranwala");

    City \*bahawalpur = new City("Bahawalpur");

    City \*sargodha = new City("Sargodha");

    City \*jhelum = new City("Jhelum");

    City \*mardan = new City("Mardan");

    City \*sheikhupura = new City("Sheikhupura");

    City \*kohat = new City("Kohat");

    City \*abbottabad = new City("Abbottabad");

    City \*mingora = new City("Mingora");

    City \*bahawalnagar = new City("Bahawalnagar");

    City \*larkana = new City("Larkana");

    City \*sukkur = new City("Sukkur");

    City \*mirpurKhas = new City("MirpurKhas");

    City \*jacobabad = new City("Jacobabad");

    City \*shikarpur = new City("Shikarpur");

    City \*kasur = new City("Kasur");

    City \*deraGhaziKhan= new City("DeraGhaziKhan");

    City \*khanewal = new City("Khanewal");

    City \*gujrat = new City("Gujrat");

    City \*muzaffarabad = new City("Muzaffarabad");

    City \*sahiwal = new City("Sahiwal");

    City \*okara = new City("Okara");

    City \*wahCantt = new City("WahCantt");

    City \*chiniot = new City("Chiniot");

    City \*kamoke = new City("Kamoke");

    City \*hafizabad = new City("Hafizabad");

    City \*khushab = new City("Khushab");

    City \*jhang = new City("Jhang");

    City \*mandiBahauddin = new City("MandiBahauddin");

    City \*murree = new City("Murree");

    City \*attock = new City("Attock");

    City \*kundian = new City("Kundian");

    City \*nowshera = new City("Nowshera");

    City \*gojra = new City("Gojra");

    City \*mansehra = new City("Mansehra");

    City \*chaman = new City("Chaman");

    City \*bannu = new City("Bannu");

    City \*daska = new City("Daska");

    City \*charsadda = new City("Charsadda");

    City \*tandoAdam = new City("TandoAdam");

    City \*karak = new City("Karak");

    City \*timergara = new City("Timergara");

    City \*kohlu = new City("Kohlu");

    City \*deraIsmailKhan = new City("DeraIsmailKhan");

    City \*khuzdar = new City("Khuzdar");

    City \*turbat = new City("Turbat");

    City \*swabi = new City("Swabi");

    City \*mianwali = new City("Mianwali");

    City \*chitral = new City("Chitral");

    City \*gawadar = new City("Gawadar");

    City \*nasirabad = new City("Nasirabad");

    City \*taxila = new City("taxila");

    City \*jauharabad = new City("Jauharabad");

    City \*shahdadkot = new City("Shahdadkot");

    City \*skardu = new City("Skardu");

    City \*nankanaSahib = new City("NankanaSahib");

    City \*khanpur = new City("Khanpur");

    City \*alang = new City("alang");

**7.Performance of the Code**

****

**![Text

Description automatically generated]()**

**![Text

Description automatically generated]()**