NAME: MUHAMMAD UMER KHAN

ENROLLMENT: 02-235221-021

DSA CCP

```
#include <iostream>
#include <unordered_map>
#include <string>
#include <string.h>
using namespace std;

const int V = 3;
const int MAX_SEATS = 45;
const int RESERVED_SEATS = 5;


struct Parcel{
	double weight, profit;
};

struct Bus {
	int busId;
	string departureLocation;
	string arrivalLocation;
	string date;
	string status;
	int totalSeats;
	int availableSeats;
	string departureTime;
	string arrivalTime;
	int reservedcount;
	bool seats[MAX_SEATS];//MAX SEATS WILL BE 45 5OR EACH BUS
	int reservedSeats[MAX_SEATS];
};

struct StandbyPassenger {
	int busIndex;
	int seatNumber;
	string bookingTime;
};

struct ReservedUser {
	string name;
	int payment;
};


Bus busDatabase[20];
ReservedUser rpassengers[10];
int busCount = 0;
string startPath, endPath;
vector<vector<int>> graph = {
	{ 0, 10, 20,15,30,27 },  // Example distances between locations
	{ 10, 0, 5,22,32,13 },
	{ 20, 5, 0, 17, 33, 26 },
```

```
        { 20, 14, 5, 6, 30, 22 },
        { 11, 1, 5, 2, 35, 13 },
        { 9, 5, 20, 8, 33, 23 }
};

vector<string> cities = {
        "Karachi",
        "Sukkur",
        "Lahore",
        "Hyderabad",
        "Peshawar",
        "Islamabad",
};


//DISPLAYING ALL THE BUSES (BUS IDS AND THEIR INDEXES)
void displayAllBuses(){

        cout << "\nBus ID's\n";
        for (int i = 0; i < busCount; i++) {
                Bus& bus = busDatabase[i];
                cout << bus.busId << endl;
        }
}

//BINARY SEARCHING FOR BUS
int binarySearch(int id){
        int low = 0;
        int high = busCount-1;
        int mid = low + high / 2;

        while (low <= high) {
                mid = low + high / 2;

                if (busDatabase[mid].busId == id) {
                        return mid;
                }
                else if (busDatabase[mid].busId < id) {
                        low = mid + 1;
                }
                else {
                        high = mid - 1;
                }
        }
        return -1;
}


void busSearching(string departure, string arrival, string date,int choice,int id) {

        cout << "Available Buses:" << endl;
        cout << "Bus ID\tDeparture\tArrival\t\tDate\t\tStatus\t\tTotal Seats\tAvailable
Seats" << endl;

        //LINEAR SEARCHING BY DEPARTURE,ARRIVAL LOCATION AND DATE OF TRAVEL
        if (choice == 1){
                for (int i = 0; i < busCount; i++) {
                        Bus& bus = busDatabase[i];
```

```
                    if (bus.departureLocation == departure && bus.arrivalLocation ==
arrival && bus.date == date) {
                            cout << i << "\t" << bus.departureLocation << "\t\t" <<
bus.arrivalLocation << "\t\t" << bus.date
                                    << "\t" << bus.status << "\t" << bus.totalSeats <<
"\t\t" << bus.availableSeats << endl;
                    }
                }
        }
        //BINARY SEARCHING BY BUS ID
        else if (choice == 2){
                int searchRes = binarySearch(id);
                if (searchRes != -1){
                        Bus& bus = busDatabase[searchRes];
                        cout << searchRes << "\t" << bus.departureLocation << "\t\t" <<
bus.arrivalLocation << "\t\t" << bus.date
                            << "\t" << bus.status << "\t" << bus.totalSeats << "\t\t" <<
bus.availableSeats << endl;

                }
                else{
                        cout << "\nBUS NOT FOUND\n";
                }
        }
}

//BOOKING BUS SEATS
int bookBusSeat(int busIndex, int seatNumber, bool isReservation) {
        Bus& bus = busDatabase[busIndex];

        if (seatNumber < 1 || seatNumber > bus.totalSeats) {
                cout << "Invalid seat number." << endl;
                return -1;
        }

        if (bus.seats[seatNumber - 1]) {
                cout << "Seat " << seatNumber << " is already occupied." << endl;
                return -1;
        }

        bus.seats[seatNumber - 1] = true;

        if (isReservation) {
                bus.reservedcount++;
                bus.reservedSeats[bus.reservedcount] = seatNumber;
                return seatNumber;
        }
        else {
                bus.availableSeats--;
                return seatNumber;
        }
}

//DISPLAYING BUS AVAILABLE SEATS
void displayBusSeats(int busIndex) {
        Bus& bus = busDatabase[busIndex];

        cout << "Available Seats for Bus " << busIndex << ":" << endl;
```

```
	cout << "Seat Number" << endl;

	for (int i = 0; i < bus.totalSeats; i++) {
		if (!bus.seats[i]) {
			cout << "Seat No: " << i + 1 << " - Available" << endl;
		}
		else {
			cout << "Seat No: " << i + 1 << " - Reserved/Booked" << endl;
		}
	}
}

void bubbleSortForStandByPassengers(ReservedUser arr[], int n) {
	for (int i = 0; i < n - 1; i++) {
		for (int j = 0; j < n - i - 1; j++) {
			if (arr[j].payment < arr[j + 1].payment) {
				swap(arr[j], arr[j + 1]);
			}
		}
	}
}

//ROUTE MAP FOR CAPTAINS
const int INF = numeric_limits<int>::max();
//FINDING THE CITY BASED ON IT'S INDEX NUMBER
//USED & FOR REFERENCING TO THE ORIGINAL OBJECT
int findCityIndex( vector<string>& cities, string& cityName)
{
	for (int i = 0; i < cities.size(); ++i) {
		if (cities[i] == cityName) {
			return i;
		}
	}
	return -1;
}

//PRINTING ALL THE POSSIBLE ROUTES (BASED ON CITIES)
void printRoute( vector<string>& cities, vector<int>& route)
{
	cout << "Route: ";
	for (int i = 0; i < route.size(); ++i) {
		cout << cities[route[i]];
		if (i != route.size() - 1) {
			cout << " -> ";
		}
	}
	cout << endl;
}

//FINDING THE ROUTES,USED 2D VECTOR FOR GRAPH
void findRoutes( vector<vector<int>>& graph, int curr, int dest, vector<int>& path,
vector<bool>& visited,  vector<string>& cities)
{
	visited[curr] = true;
	path.push_back(curr);

	if (curr == dest) {
		printRoute(cities, path);
```

```
        }
        else {
                for (int i = 0; i < graph[curr].size(); ++i) {
                        if (graph[curr][i] != 0 && !visited[i]) {
                                findRoutes(graph, i, dest, path, visited, cities);
                        }
                }
        }

        visited[curr] = false;
        path.pop_back();
}

//DIJKSTRA ALGORITHM FOR FINDING THE BEST ROUTE
void dijkstra(vector<vector<int>>& graph, int src, int dest, vector<string>& cities)
{
        //TOTAL NUMBER OF DISTANCES
        int numLocations = graph.size();
        vector<bool> visited(numLocations, false);
        vector<int> path;

        cout << "All Routes: " << endl;
        findRoutes(graph, src, dest, path, visited, cities);
}


//CARGO SHIPMENT SHORTING BASED ON PROFIT (MAX WEIGHT = 200)
void heapify(Parcel arr[], int n, int i) {
        int largest = i;
        int left = 2 * i + 1;
        int right = 2 * i + 2;

        if (left < n && arr[left].profit < arr[largest].profit)
                largest = left;

        if (right < n && arr[right].profit < arr[largest].profit)
                largest = right;

        if (largest != i) {
                swap(arr[i], arr[largest]);
                heapify(arr, n, largest);
        }
}

void heapSort(Parcel arr[], int n) {
        for (int i = n / 2 - 1; i >= 0; i--)
                heapify(arr, n, i);

        for (int i = n - 1; i >= 0; i--) {
                swap(arr[0], arr[i]);
                heapify(arr, i, 0);
        }
}

//CARGO SHIPMENT
void assignParcelsToBus(int busIndex, Parcel cargoParcels[], int numParcels) {
        // Check if busIndex is valid
        if (busIndex < 0) {
```

```
            cout << "Invalid bus index." << endl;
            return;
        }

        Bus& bus = busDatabase[busIndex];

        heapSort(cargoParcels, numParcels);

        double totalWeight = 0;
        double totalProfit = 0;
        int discardedParcels = 0;

        for (int i = 0; i < numParcels; i++) {
            if (totalWeight + cargoParcels[i].weight <= 200) {
                totalWeight += cargoParcels[i].weight;
                totalProfit += cargoParcels[i].profit;
                cout << "Assigned Parcel " << (i + 1) << " with weight: " <<
cargoParcels[i].weight << "kg and with profit: " << cargoParcels[i].profit << " Rs, to
Bus " << busIndex << endl;
            }
            else {
                cout << "Discarded Parcels: " << (i + 1) << " with weight: " <<
cargoParcels[i].weight << "kg and with profit: " << cargoParcels[i].profit << " Rs" <<
endl;
                discardedParcels++;
            }
        }

        cout << "\nTotal Weight: " << totalWeight << " kg" << endl;
        cout << "Total Profit: " << totalProfit << " Rs" << endl;
        cout << "Discarded Parcels: " << discardedParcels << endl;
}


int rpcount = 0;
void passengerMenu(){

        int choice, seatNumber, busIndex, bookedSeat,
paymentAmount,reservationChoice,option,busId;
        string departure, arrival, date, userName,bookOrRes;
        bool isReserved = false;

        cout << "Press 1 to search any bus\n";
        mainmenu:
        cout << "Press 2 to display all the available bus seats\n";
        cout << "Press 3 to book any available bus seat\n";
        cin >> choice;
        switch (choice)
        {

        case 1:

            cout << "Press 1 to search any bus by departure,arrival location and date
of travel\n";
            cout << "Press 2 to display all the buses and search any bus by it's unique
ID\n";
            cin >> option;
```

```
            if (option == 1){
                  cout << "Enter departure location: ";
                  cin >> departure;
                  cout << "Enter arrival location: ";
                  cin >> arrival;
                  cout << "Enter date of travel (YYYY-MM-DD): ";
                  cin >> date;
                  busSearching(departure, arrival, date,1,0);
            }else if (option == 2){
                  busidmenu:
                  displayAllBuses();
                  cout << "Enter any one of the above Bus ID: ";
                  cin >> busId;
                  if (busId < 0){
                        cout << "\nInvalid Bus ID!\n";
                        goto busidmenu;
                  }
                  else{
                        busSearching(departure, arrival, date, 2, busId);
                  }
            }
            goto mainmenu;
            break;
```

```
      case 2:
            cout << "Please enter the serial no of the bus of which you want to see the
available seats: ";
            cin >> busIndex;
            displayBusSeats(busIndex);
            goto mainmenu;
            break;
      case 3:

            cout << "Please enter the serial no of the bus in which you want to travel:
";
            cin >> busIndex;
            cout << "\nDo you want to book a seat or reserve a seat? Press 0 for
booking and 1 for reservation\n";
            cin >> reservationChoice;
            if (reservationChoice == 1){
                  bookOrRes = "reserved";
                  cout << "\nPlease enter your name: ";
                  cin >> userName;
                  cout << "\nWhich seat number do you want to reserve?\n: ";
                  cin >> seatNumber;
                  cout << "Please pay 500 or more deposit for the reservation(***you
will get the seat if the staff of bus approves it***)\n";
                  cin >> paymentAmount;
                  ReservedUser newUser;
                  newUser.name = userName;
                  newUser.payment = paymentAmount;
                  rpassengers[rpcount] = newUser;
                  rpcount++;
                  isReserved = true;
            }
            else{
                  booking:
                  bookOrRes = "booked";
```

```
                    cout << "\nWhich seat number do you want to book?\n: ";
                    cin >> seatNumber;
                }

                bookedSeat = bookBusSeat(busIndex, seatNumber,isReserved);
                if (bookedSeat != -1) {
                    cout << "Seat " << bookedSeat << " has been " << bookOrRes << "." <<
endl;
                }
                else{
                    goto mainmenu;
                }

                goto mainmenu;
                break;

        default:
                cout << "You entered an invalid option!";
                goto mainmenu;
                break;
        }
}

void captainMenu(){

        //USING BUILT IN HASH TABLE FOR THE LOGIN FUNCTIONALITY OF THE CAPTAIN
        // Create a hash table to store username-password pairs
        unordered_map<string, string> credentials;

        //Sample username-password pairs in the hash table
        credentials["cap1"] = "pass1";
        credentials["cap2"] = "pass2";
        credentials["cap3"] = "pass3";

        // Prompt the user for login credentials
        string username, password;
        main:
        cout << "Enter username: ";
        cin >> username;
        cout << "Enter password: ";
        cin >> password;

        // Check if the provided username exists and the password matches
        if (credentials.find(username) != credentials.end() && credentials[username] ==
password)
        {
                cout << "Welcome " << username  << " !" << endl;

                cin.ignore();
                string sourceCity, destCity;
                main_city:
                cout << "Enter the source city name: ";
                getline(cin, sourceCity);

                cout << "Enter the destination city name: ";
                getline(cin, destCity);

                int sourceIndex = findCityIndex(cities, sourceCity);
```

```
            int destIndex = findCityIndex(cities, destCity);

            if (sourceIndex != -1 && destIndex != -1) {
                  dijkstra(graph, sourceIndex, destIndex, cities);
                  char ch = 'N';
                  cout << "\nDo you want to see more routes?Press Y or N\n";
                  if (ch == 'Y' || ch == 'y'){
                        goto main_city;
                  }
            }
            else {
                  cout << "Invalid city name!" << endl;
            }

      }
      else
      {
            cout << "Invalid username or password." << endl;
            goto main;
      }
}

void staffMenu(){
      Parcel cargoParcels[50];
      int numParcels;

      cout << "Enter the number of parcels: ";
      cin >> numParcels;

      cout << "Enter the weight and profit of each parcel:\n";
      for (int i = 0; i < numParcels; i++) {
            cout << "Parcel " << (i + 1) << ":\n";
            cout << "Weight: ";
            cin >> cargoParcels[i].weight;
            cout << "Profit: ";
            cin >> cargoParcels[i].profit;
      }

      int busIndex;
      cout << "Enter the bus index to assign the parcels: ";
      cin >> busIndex;

      assignParcelsToBus(busIndex, cargoParcels, numParcels);
}


int main(){

      busCount = 6;
      busDatabase[0] = { 20230525, "Karachi", "Lahore", "2023-05-25", "Scheduled", 20,
20, "08:00", "13:00", 0 };
      busDatabase[1] = { 20230527, "Sukkur", "Karachi", "2023-05-27", "Scheduled", 30,
30, "11:00", "16:00", 0 };
      busDatabase[2] = { 20230625, "Lahore", "Hyderabad", "2023-06-25", "Scheduled", 10,
10, "01:00", "8:00", 0 };
      busDatabase[3] = { 20230526,"Lahore", "Islamabad", "2023-05-26", "Scheduled", 25,
25, "09:00", "12:00", 0 };
```

```
        busDatabase[4] = { 20230627, "Karachi","Peshawar", "2023-06-27", "Scheduled", 8,
8, "11:00", "16:00", 0 };
        busDatabase[5] = { 20230626, "Islamabad", "Karachi", "2023-06-26", "Scheduled",
15, 15, "3:00", "9:00", 0 };


        int choice;
        mainmenu:
        cout << "PRESS 1 FOR PASSENGER\n";
        cout << "PRESS 2 FOR BUS CAPTAIN\n";
        cout << "PRESS 3 FOR STAFF MEMBER\n";
        cin >> choice;

        switch (choice)
        {

        case 1:
                passengerMenu();
                break;

        case 2:
                captainMenu();
                break;

        case 3:
                staffMenu();
                break;

        default:
                cout << "Invalid option entered!Please try again\n";
                goto mainmenu;
                break;
        }

        system("pause");
        return 0;

}
```

LINEAR SEARCHING BY DEPARTURE, ARRIVAL LOCATION AND DATE OF TRAVEL

```
PRESS 1 FOR PASSENGER
PRESS 2 FOR BUS CAPTAIN
PRESS 3 FOR STAFF MEMBER
1
Press 1 to search any bus
Press 2 to display all the available bus seats
Press 3 to book any available bus seat
1
Press 1 to search any bus by departure,arrival location and date of travel
Press 2 to display all the buses and search any bus by it's unique ID
1
Enter departure location: Karachi
Enter arrival location: Lahore
Enter date of travel (YYYY-MM-DD): 2023-05-25
Available Buses:
Bus ID  Departure       Arrival         Date            Status          Total Seats     Available Seats
0       Karachi         Lahore          2023-05-25      Scheduled       20              20
```

BINARY SEARCHING BY BUS UNIQUE ID

```
PRESS 1 FOR PASSENGER
PRESS 2 FOR BUS CAPTAIN
PRESS 3 FOR STAFF MEMBER
1
Press 1 to search any bus
Press 2 to display all the available bus seats
Press 3 to book any available bus seat
Press 4 to exit
1
Press 1 to search any bus by departure,arrival location and date of travel
Press 2 to display all the buses and search any bus by it's unique ID
2

Bus ID's
20230525
20230527
20230625
20230526
20230627
20230626
Enter any one of the above Bus ID: 20230625
Available Buses:
Bus ID  Departure       Arrival         Date            Status          Total Seats     Available Seats
2       Lahore          Hyderabad               2023-06-25      Scheduled       10              10
Press 1 to search any bus
Press 2 to display all the available bus seats
Press 3 to book any available bus seat
Press 4 to exit
```

DISPLAYING ALL AVAILABLE BUS SEATS

```
Press 1 to search any bus
Press 2 to display all the available bus seats
Press 3 to book any available bus seat
Press 4 to exit
2
Please enter the serial no of the bus of which you want to see the available seats: 2
Available Seats for Bus 2:
Seat Number
Seat No: 1 - Available
Seat No: 2 - Available
Seat No: 3 - Available
Seat No: 4 - Available
Seat No: 5 - Available
Seat No: 6 - Available
Seat No: 7 - Available
Seat No: 8 - Available
Seat No: 9 - Available
Seat No: 10 - Available
Press 1 to search any bus
Press 2 to display all the available bus seats
Press 3 to book any available bus seat
Press 4 to exit
```

BOOKING A BUS SEAT

```
Do you want to book a seat or reserve a seat? Press 0 for booking and 1 for reservation
0

Which seat number do you want to book?
: 10
Seat 10 has been booked.
Press 1 to search any bus
Press 2 to display all the available bus seats
Press 3 to book any available bus seat
Press 4 to exit
2
Please enter the serial no of the bus of which you want to see the available seats: 2
Available Seats for Bus 2:
Seat Number
Seat No: 1 - Available
Seat No: 2 - Available
Seat No: 3 - Available
Seat No: 4 - Available
Seat No: 5 - Available
Seat No: 6 - Available
Seat No: 7 - Available
Seat No: 8 - Available
Seat No: 9 - Available
Seat No: 10 - Reserved/Booked
```

MAKING A RESERVATION ON A BUS SEAT (BY PAYING SOME DEPOSIT AMOUNT)

```
Please enter the serial no of the bus in which you want to travel: 2

Do you want to book a seat or reserve a seat? Press 0 for booking and 1 for reservation
1

Please enter your name: OMER

Which seat number do you want to reserve?
: 3
Please pay 500 or more deposit for the reservation(***you will get the seat if the staff of bus approves it***)
1000
Seat 3 has been reserved.
Press 1 to search any bus
Press 2 to display all the available bus seats
Press 3 to book any available bus seat
Press 4 to exit
2
Please enter the serial no of the bus of which you want to see the available seats: 2
Available Seats for Bus 2:
Seat Number
Seat No: 1 - Available
Seat No: 2 - Available
Seat No: 3 - Reserved/Booked
Seat No: 4 - Available
Seat No: 5 - Available
```

FINDING THE BEST POSSIBLE ROUTES (ROUTE MAP FOR CAPTAINS)

```
PRESS 1 FOR PASSENGER
PRESS 2 FOR BUS CAPTAIN
PRESS 3 FOR STAFF MEMBER
2
Enter username: cap1
Enter password: pass1
Welcome cap1 !
Enter the source city name: Karachi
Enter the destination city name: Lahore
All Routes:
Route: Karachi -> Sukkur -> Lahore
Route: Karachi -> Sukkur -> Hyderabad -> Lahore
Route: Karachi -> Sukkur -> Hyderabad -> Peshawar -> Lahore
Route: Karachi -> Sukkur -> Hyderabad -> Peshawar -> Islamabad -> Lahore
Route: Karachi -> Sukkur -> Hyderabad -> Islamabad -> Lahore
Route: Karachi -> Sukkur -> Hyderabad -> Islamabad -> Peshawar -> Lahore
Route: Karachi -> Sukkur -> Peshawar -> Lahore
Route: Karachi -> Sukkur -> Peshawar -> Hyderabad -> Lahore
Route: Karachi -> Sukkur -> Peshawar -> Hyderabad -> Islamabad -> Lahore
Route: Karachi -> Sukkur -> Peshawar -> Islamabad -> Lahore
Route: Karachi -> Sukkur -> Peshawar -> Islamabad -> Hyderabad -> Lahore
Route: Karachi -> Sukkur -> Islamabad -> Lahore
Route: Karachi -> Sukkur -> Islamabad -> Hyderabad -> Lahore
Route: Karachi -> Sukkur -> Islamabad -> Hyderabad -> Peshawar -> Lahore
Route: Karachi -> Sukkur -> Islamabad -> Peshawar -> Lahore
Route: Karachi -> Sukkur -> Islamabad -> Peshawar -> Hyderabad -> Lahore
Route: Karachi -> Lahore
Route: Karachi -> Hyderabad -> Sukkur -> Lahore
Route: Karachi -> Hyderabad -> Sukkur -> Peshawar -> Lahore
Route: Karachi -> Hyderabad -> Sukkur -> Peshawar -> Islamabad -> Lahore
```

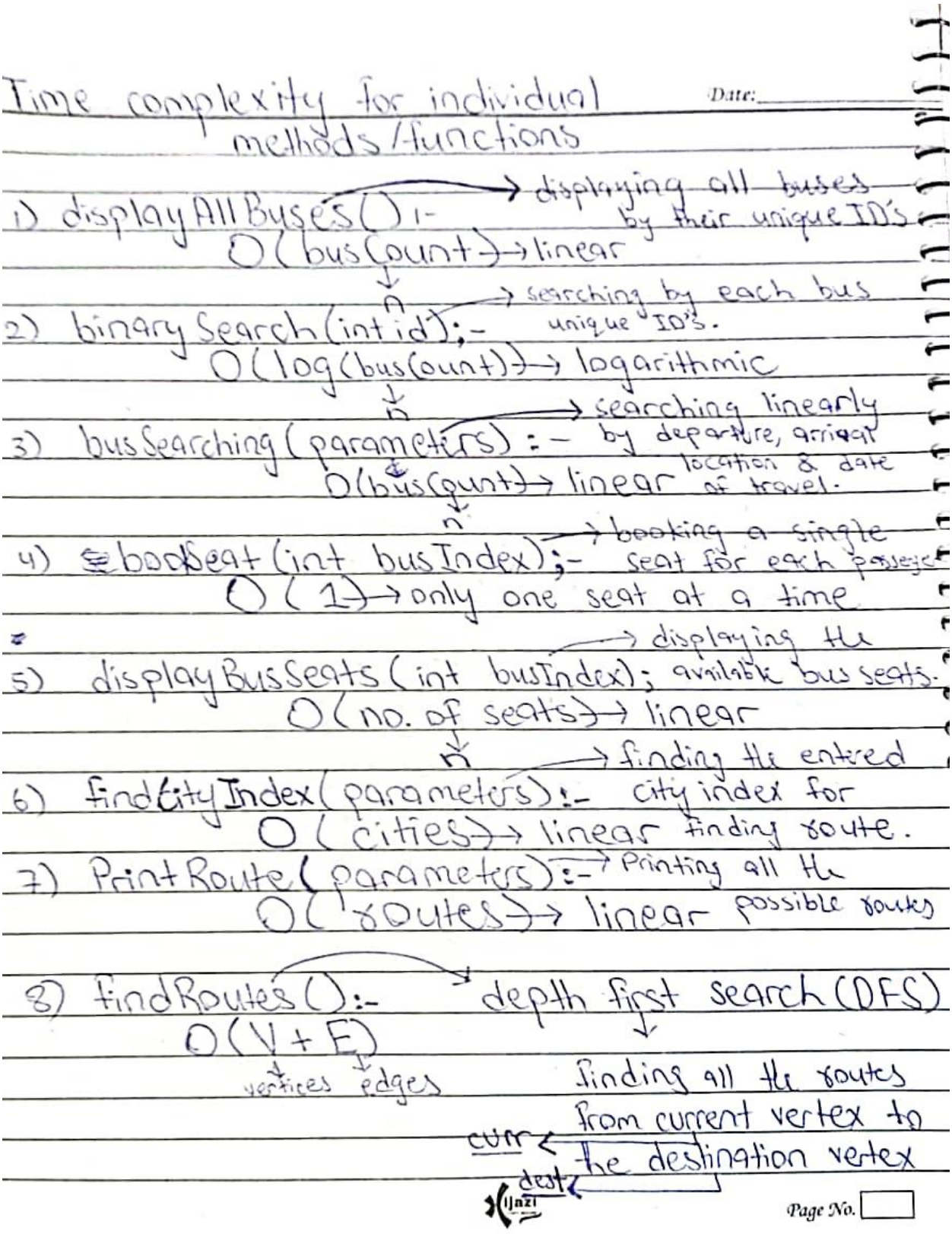
CARGO SHIPMENT (STAFF MEMBER)

```
PRESS 1 FOR PASSENGER
PRESS 2 FOR BUS CAPTAIN
PRESS 3 FOR STAFF MEMBER
3
Press 1 for cargo shipment
Press 2 for selection of standyPassengers
Press 3 to exit
1
Enter the number of parcels: 10
Enter the weight and profit of each parcel:
Parcel 1:
Weight: 10
Profit: 100
Parcel 2:
Weight: 20
Profit: 50
Parcel 3:
Weight: 30
Profit: 1000
Parcel 4:
Weight: 15
Profit: 500
Parcel 5:
Weight: 50
Profit: 280
Parcel 6:
Weight: 35
Profit: 1500
Parcel 7:
Weight: 10
Profit: 550
Parcel 8:
Weight: 40
Profit: 200
Parcel 9:
Weight: 5
Profit: 250
Parcel 10:
Weight: 100
Profit: 256
Enter the bus index to assign the parcels: 2
Assigned Parcel 1 with weight: 35kg and with profit: 1500 Rs, to Bus 2
Assigned Parcel 2 with weight: 30kg and with profit: 1000 Rs, to Bus 2
Assigned Parcel 3 with weight: 10kg and with profit: 550 Rs, to Bus 2
Assigned Parcel 4 with weight: 15kg and with profit: 500 Rs, to Bus 2
Assigned Parcel 5 with weight: 50kg and with profit: 280 Rs, to Bus 2
Discarded Parcels: 6 with weight: 100kg and with profit: 256 Rs
Assigned Parcel 7 with weight: 5kg and with profit: 250 Rs, to Bus 2
Assigned Parcel 8 with weight: 40kg and with profit: 200 Rs, to Bus 2
Assigned Parcel 9 with weight: 10kg and with profit: 100 Rs, to Bus 2
Discarded Parcels: 10 with weight: 20kg and with profit: 50 Rs

Total Weight: 195 kg
Total Profit: 4380 Rs
Discarded Parcels: 2
```

SELECTION OF STANDBY PASSENGERS

```
PRESS 1 FOR PASSENGER
PRESS 2 FOR BUS CAPTAIN
PRESS 3 FOR STAFF MEMBER
1
Press 1 to search any bus
Press 2 to display all the available bus seats
Press 3 to book any available bus seat
Press 4 to exit
3
Please enter the serial no of the bus in which you want to travel: 0

Do you want to book a seat or reserve a seat? Press 0 for booking and 1 for reservation
1

Please enter your name: omer

Which seat number do you want to reserve?
: 10
Please pay 500 or more deposit for the reservation(***you will get the seat if the staff of bus approves it***)
500
Seat 10 has been reserved.
Press 1 to search any bus
Press 2 to display all the available bus seats
Press 3 to book any available bus seat
Press 4 to exit
3
Please enter the serial no of the bus in which you want to travel: 0

Do you want to book a seat or reserve a seat? Press 0 for booking and 1 for reservation
1
 Press 4 to exit
nu4
< PRESS 1 FOR PASSENGER
< PRESS 2 FOR BUS CAPTAIN
< PRESS 3 FOR STAFF MEMBER
< 3
 cPress 1 for cargo shipment
 (Press 2 for selection of standyPassengers
  Press 3 to exit
  2
:
  Top 5 Reserved Passengers:
utName: irma, Payment: 800
utName: sharjeel, Payment: 550
  Name: omer, Payment: 500
  Name: shan, Payment: 100
```

TIME COMPLEXITIES FOR INDIVIDUAL FUNCTIONS/METHODS



Bubble sort for Stand by Passengers (parameters)

↓

$O(n^2)$

9) Dijkstra():- Date:

$O((V+E) * \log(V))$

V: vertices, E: edges

$\log(V)$ → logarithmic

uses a priority queue to select the next vertex with the shortest distance.

10) heapify():-
$O(\log n)$

11) heapSort():- → sorting the parcels using heapify method with the parcel's profit.
$O(n \log n)$

12) assignParcelsToBus:-
$O(n \log n)$
n is the number of parcels (numParcels)
assigning only parcels which are higher in profit ↓ by heapsort
max weight of total parcels will be 200.

13) PassengerMenu():- $O(1)$
14) CaptainMenu():- $O(1)$
15) Staff menu():- $O(1)$