*Daewoo Bus Services is actively working towards enhancing the efficiency of their system for their users. To cater to their passengers, they aim to develop a user-friendly application that enables users to easily search for buses. By simply entering the departure location, arrival location, and desired date of travel, users will be presented with a list of scheduled buses. Each bus will display its status, total seats, available seats, departure time, and arrival time. Upon selecting a bus, users will be shown only the available seats for booking. Additionally, each bus has 5 reserved seats, which can be utilized for standby passengers or last-minute bookings. Standby passengers will be allocated seats based on their booking timings, prioritizing those who have been waiting for an extended period.*

*These are the facilities provided to the passengers. Another module needs to be implemented specifically for the bus captains. Each bus captain will have access to a comprehensive route map, detailing all locations and terminals between the departure and arrival points. This map will assist captains in selecting the most efficient route, particularly in the event of unexpected circumstances.*

Furthermore, Daewoo Bus Services also offers cargo services. Each bus reserves a weight capacity of approximately 200kg for cargo shipments. When a staff member enters the weight and Profit for each parcel, the system will provide an optimal strategy for shipping the products on the morning shift buses. The strategy prioritizes selecting parcels that yield higher profitability and can be accommodated within the weight constraints.

**The complexity of the given code:**

* addParcelCargo: complexity =0(n)
* displaySeat: complexity =0(1)
* viewAllPassengers: complexity =0(n\*m)
* selectStandbyPassengers: 0(m \* n)
* initialize Bus: complexity =0(1)
* displayBus: complexity =0(1)
* viewAllBuses complexity =0(m)
* viewAvailableSeats: complexity =0(n)
* displayReservedSeats complexity =0(n\*m)
* bookSeat: complexity =0(n)
* viewStandbyPassengers: complexity =0(n\*m)
* main: complexity =0(n)
* Here, 'n' represents the number of parcels, the number of reserved or standby passengers and 'm' represents the number of buses.

* Overall complexity is 0(n\*m)

**Data Structures used in the code:**

**Structures:**

* **Parcel:** Represents a cargo parcel with weight and profit attributes.
* **Vector:** A dynamic array container used to store and manage the parcels and buses
* **Seat:** Represents a seat with a seat number, passenger name, and reservation time attributes.
* **Bus:** Represents a bus with attributes such as bus number, departure/arrival locations and times, total seats, available seats, reserved seats, standby passengers, cargo parcels, status, shift, weight capacity, and total profit.
* **Route:** Represents routes with attributes such as string name, distance, time, weather, description, road update

**Functions & Algorithms:**

* **viewWeightCapacityAndProfit:** Displays the weight capacity and profit of a bus.
* **displayCargoParcels**: Displays the weight and profit of all cargo parcels.
* **book Cargo**: Books a cargo parcel for a specific bus.
* **displaySeat:** Displays information about a seat, including seat number, passenger name, and reservation time.
* **viewAllPassengers**: Displays the reserved seats for each bus.
* **selectStandbyPassengers:** Selects standby passengers based on their reservation time and moves them to reserved seats.
* **initializeBus**: Initializes a bus with provided details and creates available seats and extra seats.
* **display Bus:** Displays information about a bus, including its attributes and seat availability.
* **viewAllBuses:** Displays information about all buses.
* **viewAvailableSeats:** Displays the number of available seats for each bus.
* **displayReservedSeats:** Displays the reserved seats for each bus.
* bookSeat: Books a seat for a specific bus, considering seat availability and standby options.
* **viewStandbyPassengers:** Displays the standby passengers for each bus.
* **View Routes:** displays the route of each bus
* **Queue:** is used to manage the standby passengers based on the First-In-First-Out (FIFO) method
* **Algorithms:** used to sort the specific buses, routes, seats, or cargo parcels

**Main Function:**

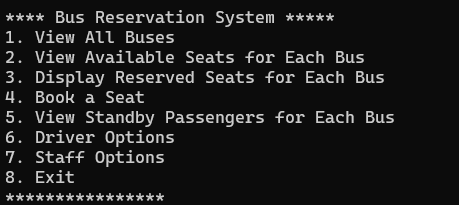
* Initializes a list of buses with some predefined values.
* Provides a menu-based interface for users to interact with the bus reservation system.
* Users can choose options to view buses, available seats, reserved seats, book seats, view standby passengers, driver options, staff options, or exit the system

**Reasons:**

Here are the reasons why used structures, functions, and algorithms

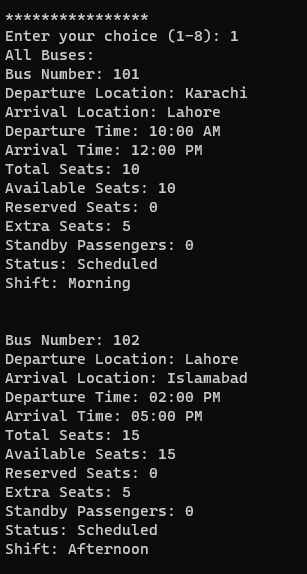
* The Bus structure represents a bus object and stores its properties such as bus number, weight capacity, profit, routes, seats, and cargo parcels. It helps organize and encapsulate the data related to each bus.
* The Route structure represents a bus route and stores its properties such as origin, destination, departure time, and arrival time. It allows for managing and tracking the routes associated with each bus.
* The Seat structure represents a seat on a bus and stores its properties such as seat number, availability, passenger name, and reservation time. It helps manage and track the seating arrangements for each bus.
* The vector container is used to store multiple instances of the Bus structure. It allows for dynamic resizing and efficient random access to elements. In this code, it is used to maintain a collection of buses, enabling the program to add, remove, and access bus objects easily.
* The queue container is used to implement a standby list for passengers. It follows the First-InFirst-Out (FIFO) principle, where the first passenger to join the standby list is the first to be processed. In this code, it stores the names of passengers who are on standby for a seat reservation.
* The use of vector and queue containers enhances the functionality and efficiency of the bus reservation system, allowing for flexible storage, retrieval, and processing of bus objects and passenger data.
* Functions provide modularity, reusability, and abstraction. Some of the functions used in the code include displaying bus details, booking seats and cargo parcels, viewing passengers and standby lists, initializing objects, and handling user input and output. These functions contribute to the overall functionality and organization of the program.
* Sorting algorithms can be used to find specific buses, routes, seats, or cargo parcels. Sorting algorithms can be employed to arrange data in a desired order, such as sorting passengers by reservation time. These algorithms help optimize the operations and improve the efficiency of the program.

**Output:**



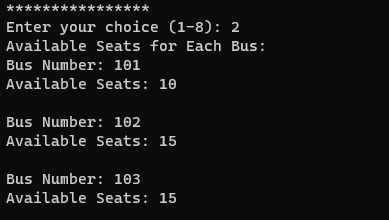
**Option :1**

If you select 1option then show all buses with Bus Number, Departure Location, Arrival Location, Departure Time, Arrival Time, Total Seats, Available Seats, Reserved Seats, Extra Seats,Standby Passengers, Status ,Shift.



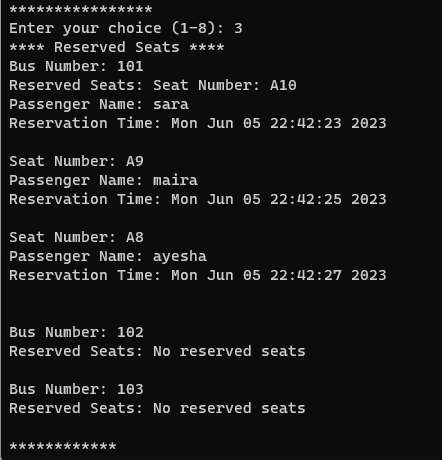
**Option:02**

If you want to select the option 2 then show only available seats of the buses



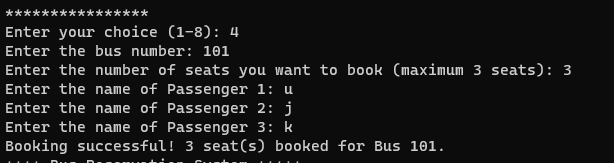
**Option:03**

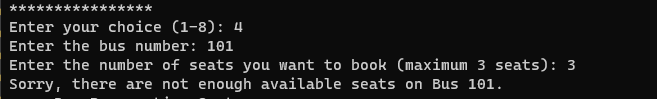
show reserved seats with date and time



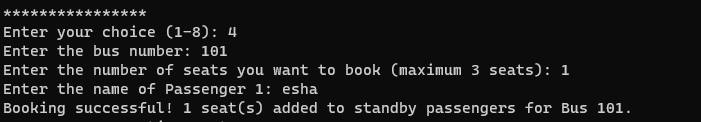
**Option:04**

If you want to select option 4 then enter the bus number then you enter the number of max 3 seats for normal users

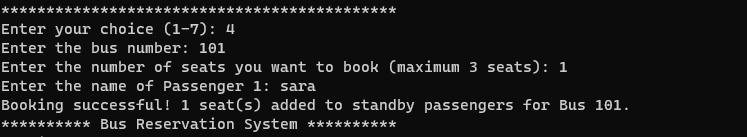


If the available space is full and you want to enter 3 seats then the message generates there are not enough seats  


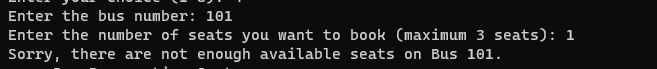
If available seats are full then a seat adding to the standby passenger



only one for standby passengers Then you enter the passenger name after that booking is successful

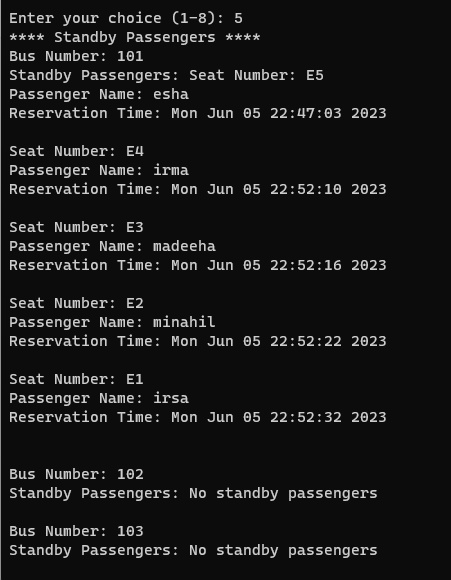


If standby passengers’ seats fully allocate then generate a message not enough seats

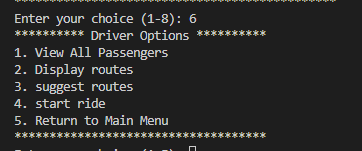


**Option:05**

If you select the option 5 show the complete booking detail about strand by passenger

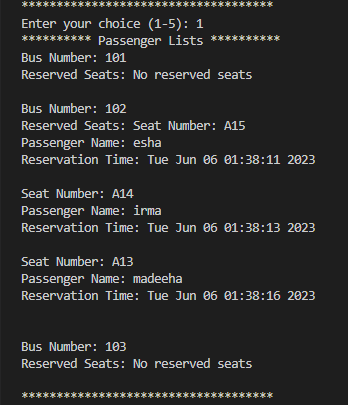


**Option:06**

****

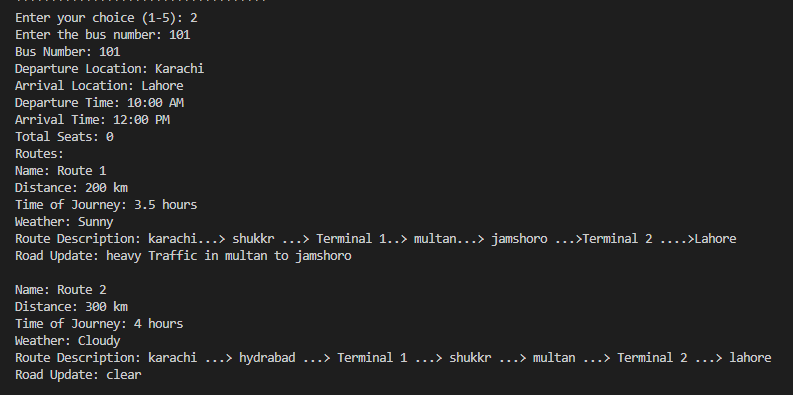
When you select option 6 show sub options then you select sub-option 1 display all the details of passengers

**Sub option 1**

****

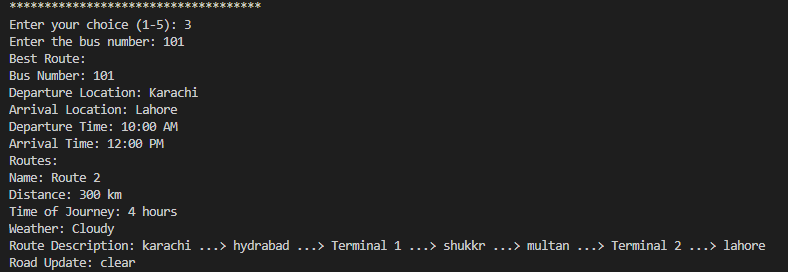
**Sub option:02**

Select sub-option 3 to show the all routes off the selected bus along with the route description, weather, time of journey, and road update and total distance



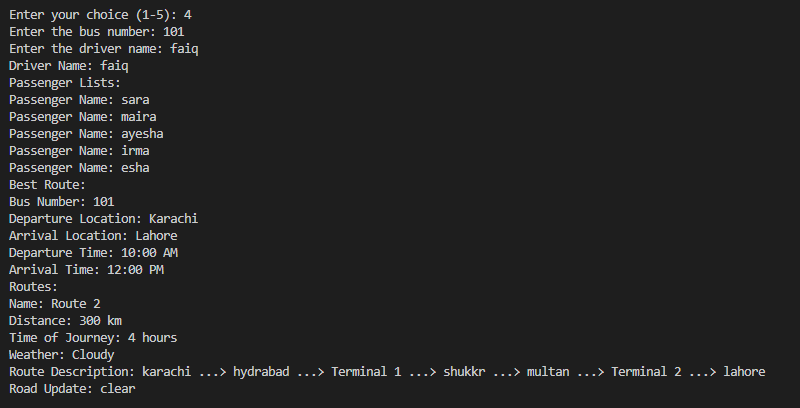
**Sub option:03**

select best route on the basis of weather roads distance and time



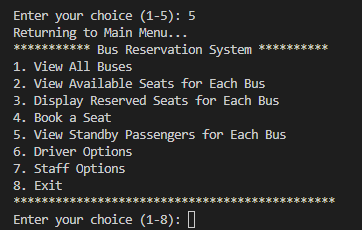
**Sub option:04**

After selecting this you enter bus number then enter driver name then show the list of passengers then show best route

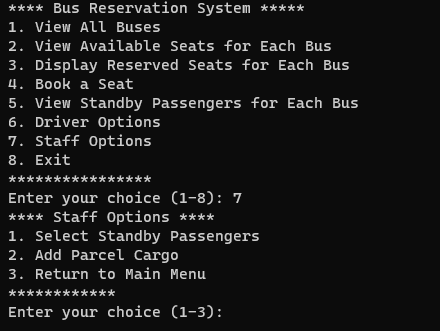
****

**Sub option:05**

You select option 6 then select sub-option 5 returning to main menu

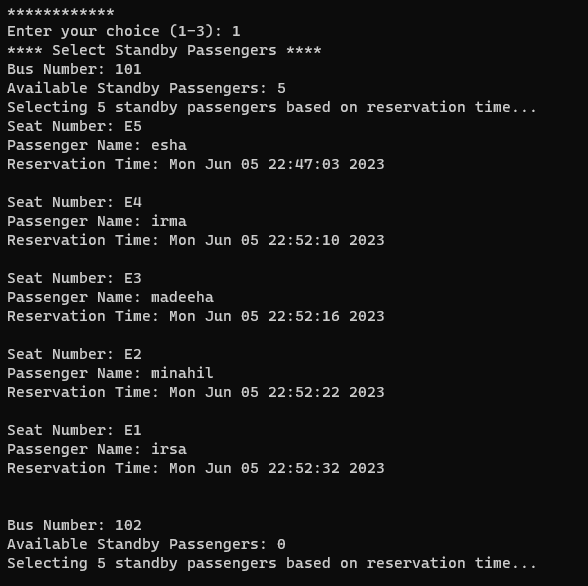


**Option:07**

****

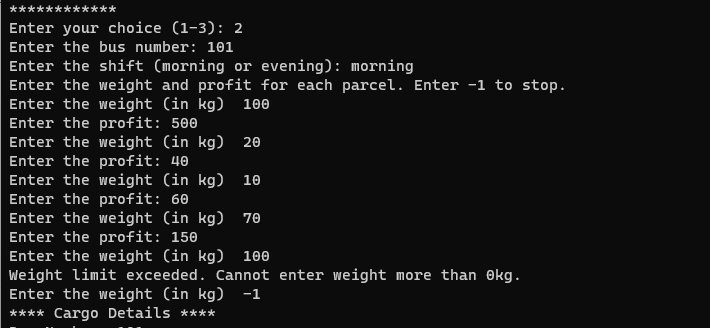
**Sub option:01**

If you select sub-option 1 then staff select standby passenger on the basis of first in and first out (FIFO)method

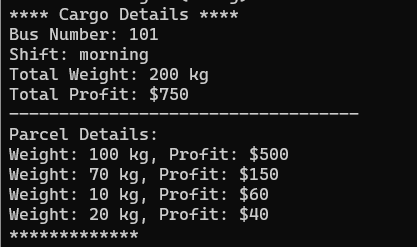


**Sub option:02**

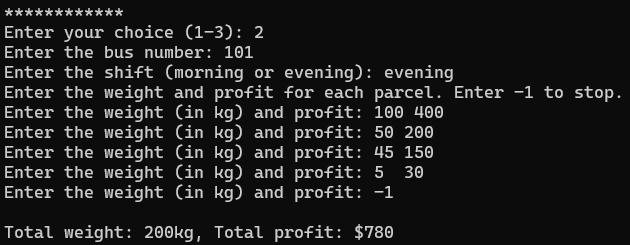
When you select 3 sub-option for cargo booking you can enter less then or equal to 200kg weight if enter greater then 200kg message generate you exceed to limit cannot enter weight more then 0kg then you enter -1 to stop



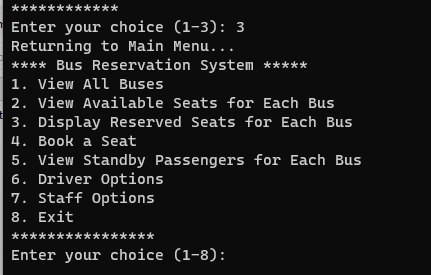
After that you see the cargo detail if you select morning then your parcel select according to the higher profit priority and then display total weight and total profit.



But in the evening only the total weight and profit



**Sub option:03**

Then you select sub option 3 back to the main menu  


**Option:08**

Show to exit the program

