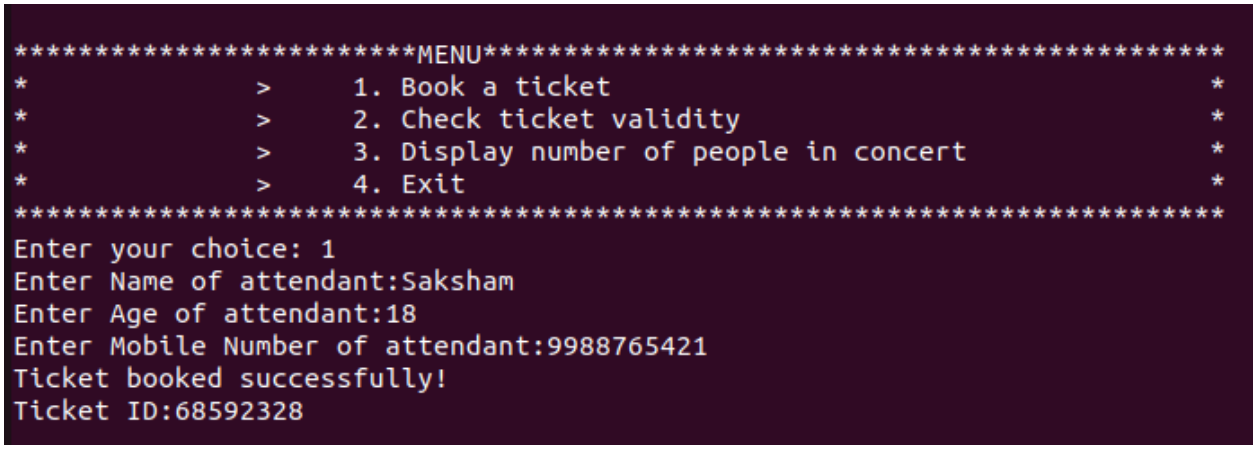
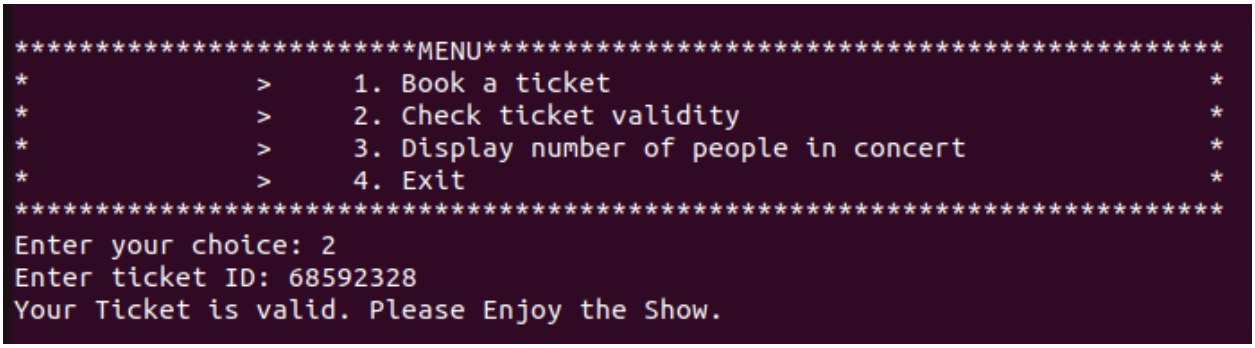
Problem no.2: 1. Imagine that you are organizing a music festival and need to manage the ticket sales at the entrance. You have a limited number of tickets and want to ensure that everyone who has purchased a ticket can enter the festival without any issues. However, you also need to keep track of the number of people who have entered the festival, and ensure that nobody enters without a valid ticket.

How can you manage the ticket sales and ensure that only those with valid tickets are allowed to
enter the festival? Can you think of a way to keep track of the number of people who have entered the festival using a data structure? Can you also find a way to quickly check whether a person has a valid ticket or not, using a simple algorithm?

Output Images:



Booking a ticket



Valid ticket

```
************************MENU***************************************************
*              >     1. Book a ticket                                           *
*              >     2. Check ticket validity                                   *
*              >     3. Display number of people in concert                     *
*              >     4. Exit                                                    *
*********************************************************************************
Enter your choice: 2
Enter ticket ID: 687548291
Your Ticket is invalid.
```

Invalid Ticket

```
************************MENU***************************************************
*              >     1. Book a ticket                                           *
*              >     2. Check ticket validity                                   *
*              >     3. Display number of people in concert                     *
*              >     4. Exit                                                    *
*********************************************************************************
Enter your choice: 2
Enter ticket ID: 68592328
Ticket has been already used.
```

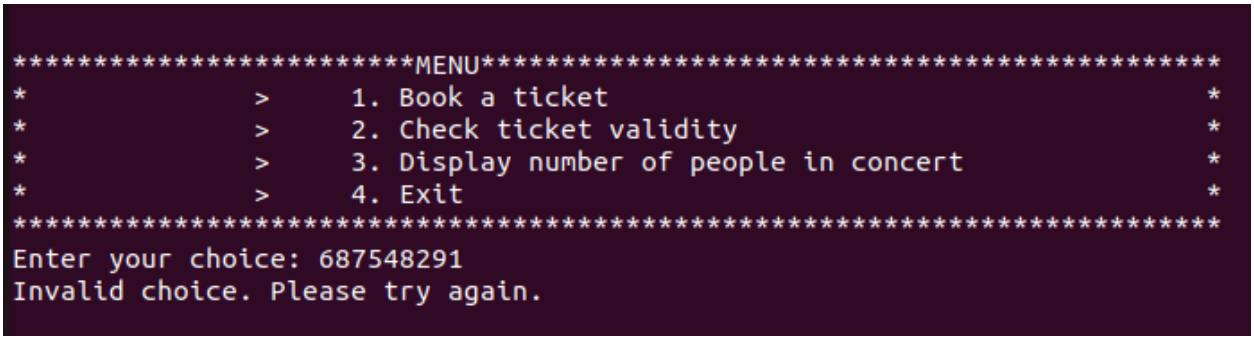
Already used Ticket.

```
************************MENU***************************************************
*              >     1. Book a ticket                                           *
*              >     2. Check ticket validity                                   *
*              >     3. Display number of people in concert                     *
*              >     4. Exit                                                    *
*********************************************************************************
Enter your choice: 3
Number of people attending the concert: 1
```

People attending the concert.

```
************************MENU***************************************************
*              >     1. Book a ticket                                           *
*              >     2. Check ticket validity                                   *
*              >     3. Display number of people in concert                     *
*              >     4. Exit                                                    *
*********************************************************************************
Enter your choice: 4
Exiting..... Have a nice day!
```

Exiting



Entering Invalid Choice

TIME COMPLEXITY:

The time complexity for the different functions used in the code are as follows:

The generate id function has a time complexity of O(1), as it only involves generating a random number, which is a constant time operation.

The book function has a time complexity of O(n^2), where n is the number of tickets sold. The outer loop runs n times, and the inner loop also runs n times at worst. Therefore, the time complexity of the book function is O(n^2).

The validate function has a time complexity of O(n), where n is the number of tickets sold. This is because the function has a single loop that iterates over the tickets array, which has a maximum length of n.

The display function has a time complexity of O(n). This is because the function has a single loop that iterates over the tickets array, which has a maximum length of n.

The time complexity of the entire function will depend on which particular function we call from the menu defined in the main block. For the book function, the time complexity will be the worst at O(n^2), while for the rest of the other functions, the time complexity would be O(n)

SPACE COMPLEXITY:

The main data structure which we used here is an array of structures called as tickets. The array has a fixed size of 100, and each structure has five fields that take up memory. The respective fields are: ‘id’, ‘name’, ‘age’, ‘mobile number’, ‘valid’, and ‘used’.

The amount of memory used by this program will depend on the number of tickets sold and the size of each ticket structure. The maximum number of tickets that can be sold is 100, and each ticket structure takes up a fixed amount of memory regardless of the number of fields filled in for each ticket.

Therefore, the space complexity of this program is O(1) because the amount of memory used by the program is constant and does not depend on the size of the input.

ALGORITHM:

Declare a structure named Ticket with members id, name, age, mobile num, valid, and used.

Declare an array named tickets of type Ticket with a size of 100.
Declare an integer variable named total tickets and initialize it to 0.

Define a function named generate id which generates a random integer between 0 and 99999999 and returns it.

Define the main function which displays a menu to the user and calls the appropriate functions based on the user's choice.

Define a function named book which books a ticket by:

- Checking if the total number of tickets sold is less than 100.
- Generating a unique ticket ID by calling the generate_id function.
- Checking if the generated ticket ID is already present in the tickets array. If yes, generate a new ticket ID and check again.
- Adding the ticket to the tickets array with the generated ticket ID, name, age, and mobile_num.
- Setting the valid flag to 1.
- Incrementing the total tickets variable.
- Displaying the ticket ID and a success message.

Define a function named validate which validates a ticket by:

- Taking the ticket ID as input.
- Searching for the ticket in the tickets array by matching the ticket ID.
- If the ticket is found, check if it has already been used or not. If yes, display a message that the ticket has already been used. If not, set the used flag to 1 and display a message that the ticket is valid.
- If the ticket is not found, display a message that the ticket is invalid.

Define a function named display which displays the number of people attending the concert by:

- Initializing a count variable to 0.
- Looping through the tickets array and counting the number of tickets with the valid flag set to 1.
- Displaying the count variable.

WHAT WE COULD HAVE DONE BETTER?

The time complexity in our program has the worst case of O(n^2) in the booking function. Although this is a small-scale program and time complexity doesn't matter that much in this case, but if the same program would have been used on a bigger scale, so due to the worst time complexity, the execution of the program would have taken a lot of time.

Program written by:

- Arnav Aditya (2210110189)
- Arnav Jalan (2210110192)
- Rachit Sen (2210110489)