Table of Contents

Introduction;

This C++ program is designed to manage a parking lot with limited space for rickshaws, cars, and buses. Users can input their choice to park a vehicle or view/delete parking records. The program calculates the total amount earned and keeps track of the number of vehicles parked, including a breakdown for each vehicle type. It ensures that the parking capacity does not exceed 50 vehicles and provides a simple menu interface for user interaction.

Problem Statement;

The program aims to address the challenge of managing a parking lot with limited space while keeping track of the number and type of vehicles parked. It also aims to provide a user-friendly interface for users to interact with the parking management system.

Objective;

The objectives of the project include:

- Efficiently managing the parking lot with limited space for rickshaws, cars, and buses.
- Allowing users to input their choice to park a vehicle or view/delete parking records.
- Calculating the total amount earned and keeping track of the number of vehicles parked, including a breakdown for each vehicle type.
- Ensuring that the parking capacity does not exceed 50 vehicles.
- Providing a simple menu interface for user interaction.

Methodology;

The program is implemented using C++ and utilizes a simple menu-driven approach for user interaction. It uses variables to keep track of the number of rickshaws, cars, and buses parked, as well as the total amount earned. The program includes conditional statements to handle user input and ensure that the parking capacity does not exceed 50 vehicles.

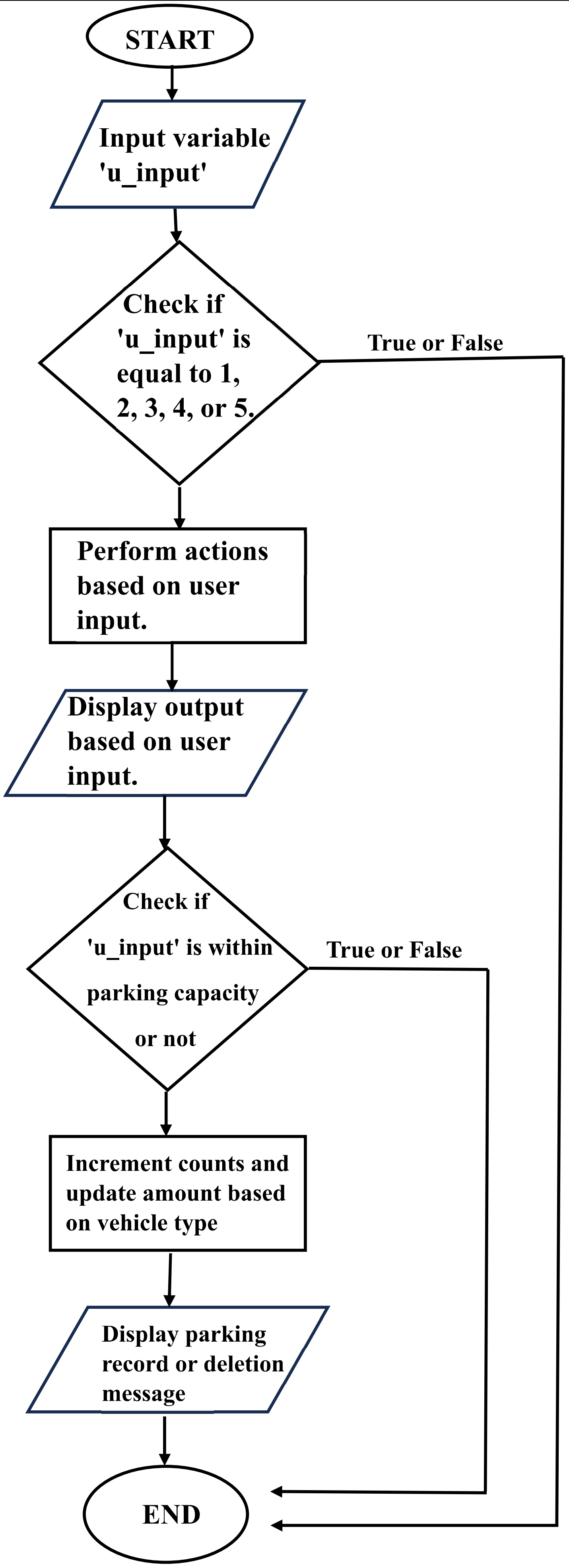
Program Implementation;

The program uses a while loop to continuously display the menu options and handle user input. It includes conditional statements to check the user input and perform the appropriate actions, such as incrementing the count of vehicles parked and updating the total amount earned. The source code provided demonstrates the implementation of the program.

Sudo code;

```
Initialize variables: c = 0, r = 0, b = 0, u_input, amount = 0, count = 0
while (true)
{
    Display menu options:
    "Press 1 for rickshaw"
    "Press 2 for cars"
    "Press 3 for bus"
    "Press 4 to show the record"
    "Press 5 to delete the record"
    Read user input (u_input)
    if (u_input == 1 && count <= 50)
    {
        r += 1
        amount += 100
        count += 1
    }
    else if (u_input == 2 && count <= 50)
    {
        c += 1
        amount += 200
        count += 1
    }
    else if (u_input == 3 && count <= 50)
    {
        b += 1
        amount += 300
        count += 1
    }
    else if (u_input)
    {
        Display record:
        "********************************"
        "Total amounts =", amount
        "Total vehicles parked =", count
        "Rickshaws parked =", r
        "Cars parked =", c
        "Buses parked =", b
        "********************************"
    }
    else if (u_input == 5)
    {
        Reset all variables to 0
        Display:
        "********************************"
        "Record Deleted"
        "********************************"
    }
    else
    {
        Display "Invalid number"
    }
}
Return 0
```

FLOWCHART


START
Input variable 'u_input'
Check if 'u_input' is equal to 1, 2, 3, 4, or 5.
True or False
Perform actions based on user input.
Display output based on user input.
Check if 'u_input' is within parking capacity or not
True or False
Increment counts and update amount based on vehicle type
Display parking record or deletion message
END

SOURCE CODE;

```
#include <iostream> using
namespace std;
int main()  { int c = 0; int r
= 0; int b = 0; int u_input;
int amount = 0, count = 0;
// menu while (true)
 {
cout << "press 1 for rickshaw:" << endl;
cout << "press 2 for cars:" << endl; cout
<< "press 3 for bus:" << endl; cout <<
"press 4 to show the record:" << endl;
cout << "press 5 to delete the record:" <<
endl; cin >> u_input;
if (u_input == 1)
 {
if (count <= 50)
{
r = r + 1; amount =
amount + 100;
count = count + 1;
}
 else
cout << "no more rickshaw parking is
full:" << endl;
}
 else if (u_input == 2)
{ if (count <=
50)
{
amount = amount + 200;
count = count + 1; c = c +
1; }
 else
cout << "car parking is full:" << endl;
}
 else if (u_input == 3) {
if (count <= 50)
{
b = b + 1; amount =
amount + 300;
count = count + 1;
}
 else
cout << "bus parking is full:" << endl;
}
 else if (u_input == 4)
 {
cout
<<"*********************************"
<< endl;
cout << "The total amounts =" << amount
<< endl; cout << "The total numbers of
vehicles parked=" << count << endl;
cout << "The total numbers of
rickshaws parked=" << r << endl; cout
<< "The total numbers of cars
parked=" << c << endl; cout << "The
total numbers of buses parked=" << b
<< endl; cout <<
"*********************************"
<<
endl << endl;
 }
 else if (u_input == 5)
{
 amount =
0;  count =
0;  r = 0;  c
= 0;   b = 0;
  cout <<
"*********************************"
<< endl;
 cout << "Record Deleted:" << endl;
cout <<
"*********************************"
<<
endl << endl;
 }
else
{
 cout << "invalid number:"<< endl;
  }
  }
  return 0;
  }
```

Outputs Are;

All Record;

```
press 1 for rickshaw: press 2 for
cars: press 3 for bus: press 4 to
show the record: press 5 to delete
the record:
4
********************************
The total amounts =0
The total numbers of vehicles parked=0
The total numbers of rickshaws parked=0
The total numbers of cars parked=0
The total numbers of buses parked=0
```

Car Record;

```
press 1 for rickshaw: press 2 for
cars: press 3 for bus: press 4 to
show the record: press 5 to delete
the record:
2 press 1 for rickshaw: press 2 for
cars: press 3 for bus: press 4 to
show the record: press 5 to delete
the record:
4
********************************
The total amounts =200
The total numbers of vehicles parked=1
The total numbers of rickshaws parked=0
The total numbers of cars parked=1
The total numbers of buses parked=0
```

Buses Record;

```
press 1 for rickshaw: press 2 for
cars: press 3 for bus: press 4 to
show the record: press 5 to delete
the record:
3 press 1 for rickshaw: press 2 for
cars: press 3 for bus: press 4 to
show the record: press 5 to delete
the record:
4
********************************
The total amounts =300
The total numbers of vehicles parked=1
The total numbers of rickshaws parked=0
The total numbers of cars parked=0
The total numbers of buses parked=1
```

Rickshaw Record;

```
 press 1 for rickshaw: press
2 for cars: press 3 for bus:
press 4 to show the record:
press 5 to delete the record:
1 press 1 for
rickshaw: press 2
for cars: press 3 for
bus:
press 4 to show the record:
press 5 to delete the record:
4
********************************
The total amounts =100
The total numbers of vehicles parked=1
The total numbers of rickshaws parked=1
The total numbers of cars parked=0
The total numbers of buses parked=0
********************************
```

Delete All Record;

```
press 1 for rickshaw:
press 2 for cars: press
3 for bus:
press 4 to show the record:
press 5 to delete the record:
5
********************************
Record Deleted:
********************************
```

Results;

The program provides an interactive interface for users to input their choices and view/delete parking records. It maintains a record of the total amount earned, the number of vehicles parked, and the breakdown for each vehicle type. Users can easily access this information through the menu interface.

Conclusion;

In conclusion, the car parking management system successfully achieves its objectives by effectively managing the parking lot, providing a user-friendly interface, and accurately tracking parking records. The program offers a practical solution for managing a parking lot with limited space.