
UNIVERSITY OF
ENGINEERING AND TECHNOLOGY
MARDAN

Name:
**Abdul Haseeb**

Reg. No:
**22MDSWE197**

DS&A Lab 2

Submitted to:
**Engr. Sohail**

**1: Declare and initialize an array of size taken from user as input, display array values on screen and find sum of array elements and finally display the summation result.**

**2: Declare an array of size 8. Receive all the elements from the user as input using for- loop. Display the entered elements the array in reverse order**

**3: Use an int pointer and an int variable and displays the variables address**

**4: Receive input from user, store it in integer variable, display what the user entered, via the variable's address.**

**5: Create a float array of 10 elements. Using pointer arithmetic, display the elements at odd numbered positions.**

**6: Create a class named "Car" with attributes xPosition, yPosition, speed. The class should have methods such as "accelerate", "decelerate" to increment and decrement the speed of the car while for change in xPosition & yPosition, there should be methods such as moveForward, moveBackwards, turnLeft and turnRight. An extra method "currState" should display all the data members of the object.**

**7: Create an array of 10 cars, randomly assigning values to xPosition, yPosition using the parameterized constructor of the Car class. Using a loop of 100 iterations, update and display the state of all the cars. For updating the status, randomly select a car and then randomly move it in 1 of the 4 directions. The loop should also notify if any car collides with another car.**

**Note:**
**1. for randomly selecting a car, use rand() function to generate a random value between 0 to 9. Use that number for the index value to read the car object.**
**2. For randomly selecting a direction to move, use rand() function to generate a random value between 1 and 4. Use Switch statement to call the move function associated with the value**

**1: Declare and initialize an array of size taken from user as input, display array values on screen and find sum of array elements and finally display the summation result.**

```cpp
#include<iostream>

using namespace std;

int main(){
    int size;
    cout<<"Please enter the size of the array: ";
    cin>>size;

   int Array[size];

   for(int i=0; i<size; i++ ){
    cout<<"Please enter the item at index "<<i<<": ";
    cin>>Array[i];


   }

   cout<<"\nNow printing the elements of the array\n";

   for(int i=0; i<size; i++){
    cout<<"The element at the Array["<<i<<"] is: "<<Array[i]<<endl;
   }


}
```

## OUTPUT:

```
Please enter the size of the array: 5
Please enter the item at index 0: 23
Please enter the item at index 1: 53
Please enter the item at index 2: 87
Please enter the item at index 3: 32
Please enter the item at index 4: 54

Now printing the elements of the array
The element at the Array[0] is: 23
The element at the Array[1] is: 53
The element at the Array[2] is: 87
The element at the Array[3] is: 32
The element at the Array[4] is: 54
```

**2: Declare an array of size 8. Receive all the elements from the user as input using for- loop. Display the entered elements the array in reverse order**

```cpp
#include<iostream>

using namespace std;

int main(){

    cout<<"Pease enter the 8 integer: \n";
    int Array[8];

    for(int i=0; i<8 ; i++){
        cout<<"Please enter the no."<<i<<" item: ";
        cin>>Array[i];
    }

    cout<<"\nNow displaying the values of the array:\n\n";

    for(int i=7; i>=0; i--){
        cout<<"The number at index["<<i<<"] is: "<<Array[i]<<endl;
    }
}
```

## OUTPUT:

```
Pease enter the 8 integer:
Please enter the no.0 item: 32
Please enter the no.1 item: 14
Please enter the no.2 item: 2
Please enter the no.3 item: 41
Please enter the no.4 item: 34
Please enter the no.5 item: 6
Please enter the no.6 item: 3
Please enter the no.7 item: 2

Now displaying the values of the array:

The number at index[7] is: 2
The number at index[6] is: 3
The number at index[5] is: 6
The number at index[4] is: 34
The number at index[3] is: 41
The number at index[2] is: 2
The number at index[1] is: 14
The number at index[0] is: 32
```

**3: Use an int pointer and an int variable and displays the variables address**

```cpp
#include<iostream>

using namespace std;

int main(){
    int a=14;


    cout<<"The adress of a is: "<<(&a)<<" \nand its value is: "<<*(&a);
}
```

## OUTPUT:

```
The adress of a is: 0xf0037ffdcc
and its value is: 14
```

**4: Receive input from user, store it in integer variable, display what the user entered, via the variable's address.**

```cpp
#include<iostream>
using namespace std;

int main(){
    cout<<"Please enter the number: ";
    int num;
    int* ptr = &num;
    cin>>*ptr;
    cout<<"You entered: "<<*(&num);

}
```

## OUTPUT:

```
Please enter the number: 256
You entered: 256
```

**5: Create a float array of 10 elements. Using pointer arithmetic, display the elements at odd numbered positions.**

```cpp
#include<iostream>

using namespace std;

int main(){
    cout<<"Please enter the 5 float values\n\n";

    float Array[10];

    for(int i=0; i<10; i++){

        cout<<"Please enter the float at ["<<i<<"] : ";
        cin>>Array[i];

    }

    cout<<"\nNow using pointer arithematic to print the values of the float array:\n\n";

    for(int i=1; i<10; i+=2){
        cout<<"\nThe float value at the index["<<i<<"] is: "<<*(Array+i);
    }
}
```

## OUTPUT 5:

```
Please enter the 10 float values

Please enter the float at [0] : 12
Please enter the float at [1] : 32
Please enter the float at [2] : 54
Please enter the float at [3] : 2
Please enter the float at [4] : 76
Please enter the float at [5] : 21
Please enter the float at [6] : 56
Please enter the float at [7] : 23
Please enter the float at [8] : 56
Please enter the float at [9] : 26

Now using pointer arithematic to print the values of the float array:


The float value at the index[1] is: 32
The float value at the index[3] is: 2
The float value at the index[5] is: 21
The float value at the index[7] is: 23
The float value at the index[9] is: 26
```

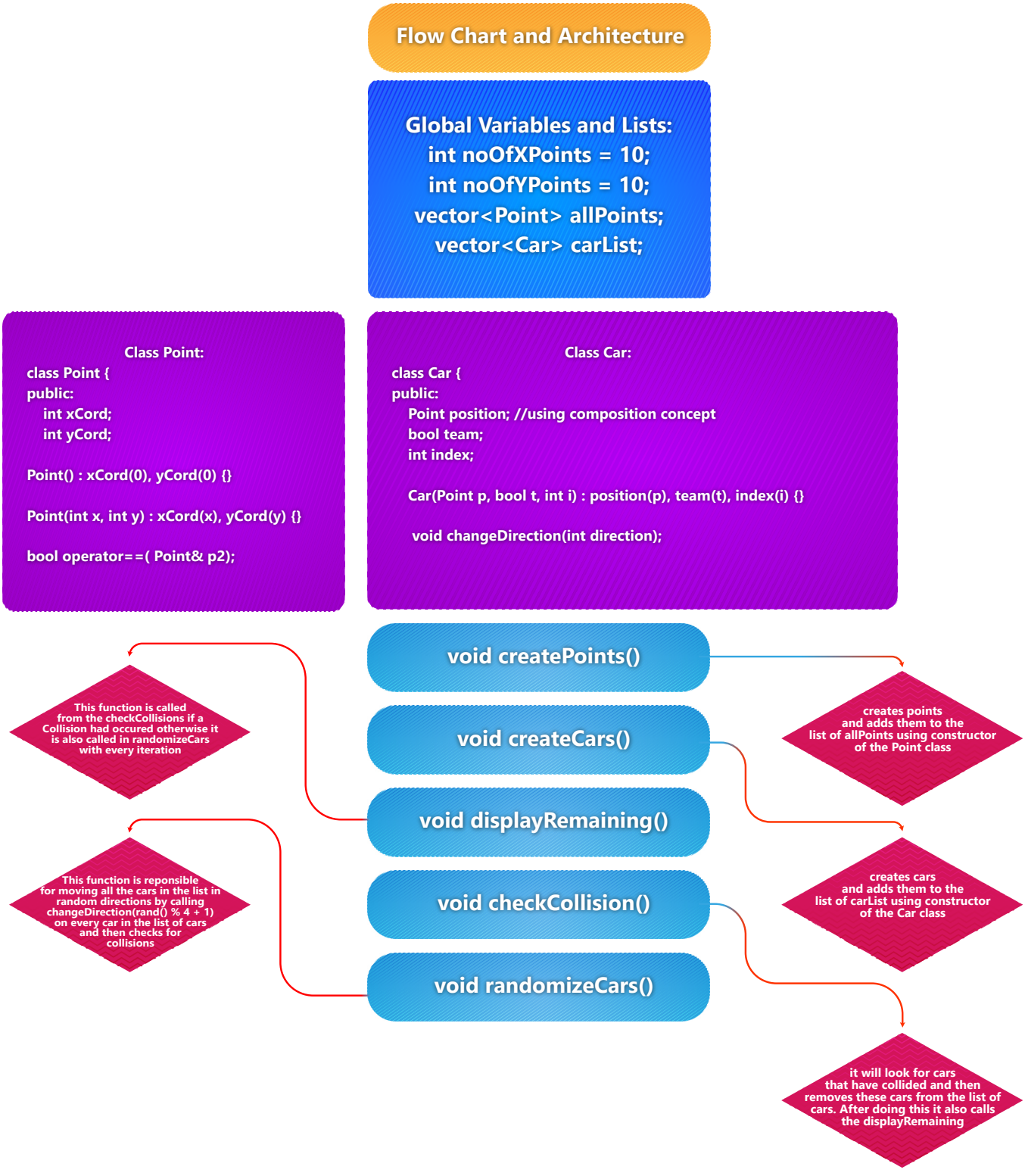
**6: Create a class named “Car” with attributes xPosition, yPosition, speed. The class should have methods such as “accelerate”, “decelerate” to increment and decrement the speed of the car while for change in xPosition & yPosition, there should be methods such as moveForward, moveBackwards, turnLeft and turnRight. An extra method “currState” should display all the data members of the object.**

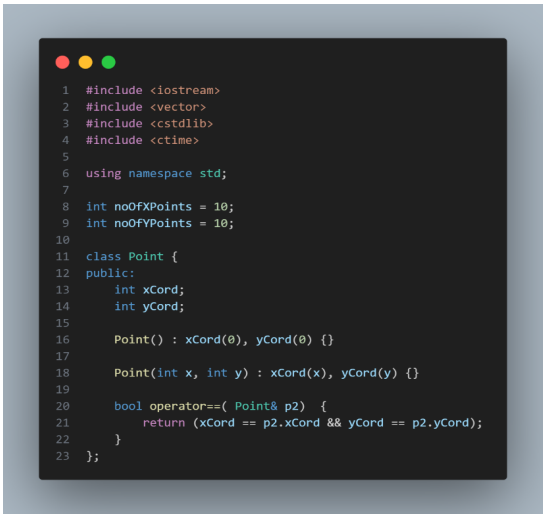
```cpp
#include <iostream>

using namespace std;

class Car
{
    int xCord = 0;
    int yCord = 0;
    int speed = 0;

public:
    void aceelerate()
    {
        speed++;
        cout << "Accelerating, speed: " << speed << endl;
    }
    void decelerate()
    {
        speed--;
        cout << "Decelerating, speed: " << speed << endl;
    }

    void turnRight()
    {
        cout << "Turning right" << endl;
        xCord++;
    }

    void turnLeft()
    {
        cout << "Turning left" << endl;
        xCord--;
    }

    void moveForward()
    {
        cout << "Moving forward" << endl;

        yCord += speed;
    }

    void moveBack()
    {
        cout << "Moving Back" << endl;
        yCord -= speed;
    }

    void carState()
    {
        cout << "The car is now at (" << xCord << ", " << yCord << ") \n";
        cout << "speed is: " << speed << endl;
    }
};

// enum direction {right, left, forward, back};

int main()
{

    Car *car = new Car();

    car->aceelerate();
    car->aceelerate();

    car->moveForward();
    car->moveForward();

    car->carState();
    car->turnLeft();
    car->turnLeft();

    car->carState();
}
```

## OUTPUT 6:

```
Accelerating, speed: 1
Accelerating, speed: 2
Moving forward
Moving forward
The car is now at (0, 4)
speed is: 2
Turning left
Turning left
The car is now at (-2, 4)
speed is: 2
```

**7: Create an array of 10 cars, randomly assigning values to xPosition, yPosition using the parameterized constructor of the Car class. Using a loop of 100 iterations, update and display the state of all the cars. For updating the status, randomly select a car and then randomly move it in 1 of the 4 directions. The loop should also notify if any car collides with another car.**

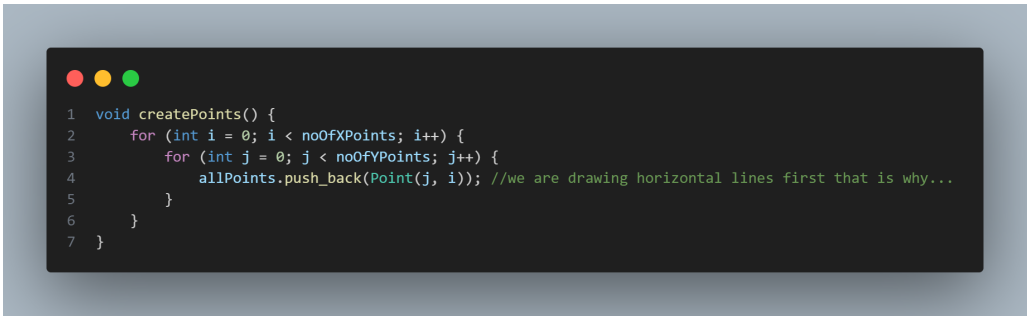




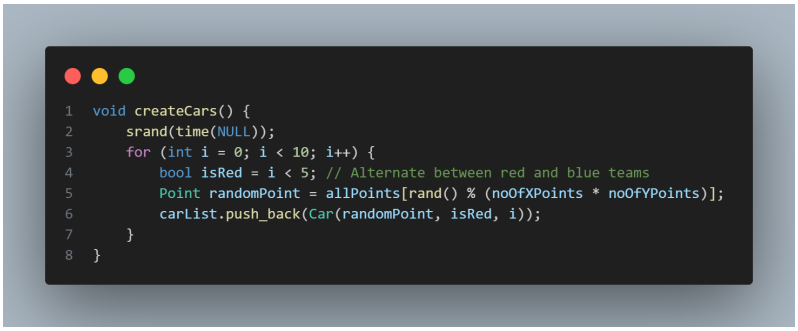
```cpp
#include <iostream>
#include <vector>
#include <cstdlib>
#include <ctime>

using namespace std;

int noOfXPoints = 10;
int noOfYPoints = 10;

class Point {
public:
    int xCord;
    int yCord;

    Point() : xCord(0), yCord(0) {}

    Point(int x, int y) : xCord(x), yCord(y) {}

    bool operator==( Point& p2)  {
        return (xCord == p2.xCord && yCord == p2.yCord);
    }
};
```

```cpp

vector<Point> allPoints;
vector<Car> carList;
```

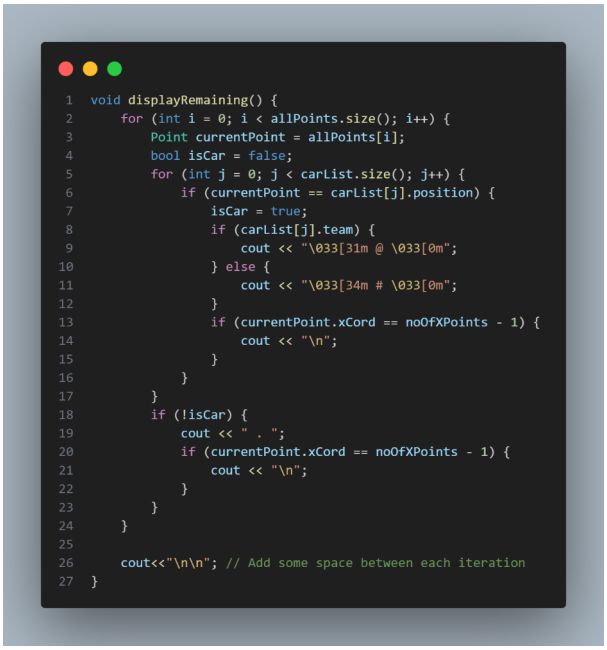
```cpp
class Car {
public:
    Point position;
    bool team;
    int index;

    Car(Point p, bool t, int i) : position(p), team(t), index(i) {}

    void changeDirection(int direction) {
        switch (direction) {
            case 1:
                if (position.yCord != 0) {
                    position.yCord--;
                }
                break;
            case 2:
                if (position.yCord != noOfYPoints - 1) {
                    position.yCord++;
                }
                break;
            case 3:
                if (position.xCord != 0) {
                    position.xCord--;
                }
                break;
            case 4:
                if (position.xCord != noOfXPoints - 1) {
                    position.xCord++;
                }
                break;
            default:
                cout << "Invalid direction" << endl;
                break;
        }
    }
};
```



```cpp
void createPoints() {
    for (int i = 0; i < noOfXPoints; i++) {
        for (int j = 0; j < noOfYPoints; j++) {
            allPoints.push_back(Point(j, i)); //we are drawing horizontal lines first that is why...
        }
    }
}
```



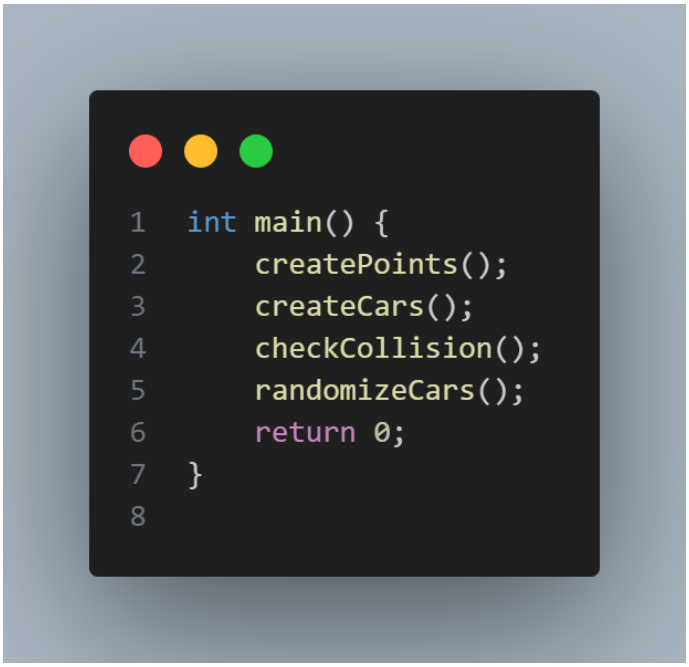
```cpp
void createCars() {
    srand(time(NULL));
    for (int i = 0; i < 10; i++) {
        bool isRed = i < 5; // Alternate between red and blue teams
        Point randomPoint = allPoints[rand() % (noOfXPoints * noOfYPoints)];
        carList.push_back(Car(randomPoint, isRed, i));
    }
}
```



```cpp
void displayRemaining() {
    for (int i = 0; i < allPoints.size(); i++) {
        Point currentPoint = allPoints[i];
        bool isCar = false;
        for (int j = 0; j < carList.size(); j++) {
            if (currentPoint == carList[j].position) {
                isCar = true;
                if (carList[j].team) {
                    cout << "\033[31m @ \033[0m";
                } else {
                    cout << "\033[34m # \033[0m";
                }
                if (currentPoint.xCord == noOfXPoints - 1) {
                    cout << "\n";
                }
            }
        }
        if (!isCar) {
            cout << " . ";
            if (currentPoint.xCord == noOfXPoints - 1) {
                cout << "\n";
            }
        }
    }

    cout<<"\n\n"; // Add some space between each iteration
}
```

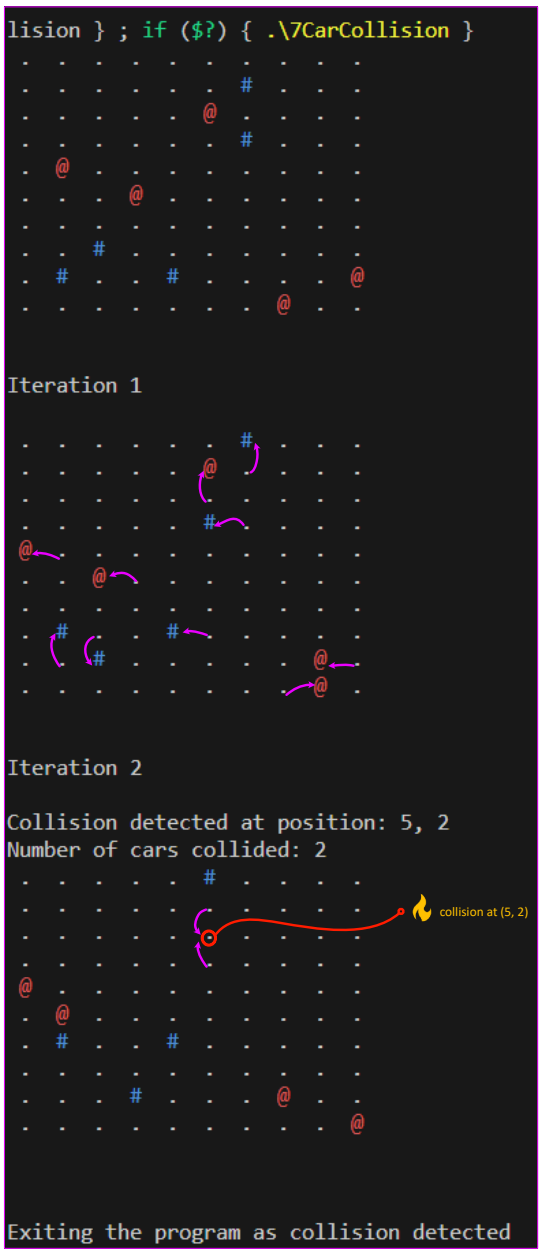
```cpp
void checkCollision() {
    for (int i = 0; i < carList.size(); i++) {
        for (int j = i + 1; j < carList.size(); j++) {//
            if (carList[i].position == carList[j].position) {
                cout << "Collision detected at position: " << carList[i].position.xCord << ", " << carList[i].position.yCord << endl;
                cout << "Number of cars collided: 2" << endl;
                carList.erase(carList.begin() + j);
                carList.erase(carList.begin() + i);

                displayRemaining();

                cout<<"\nExiting the program as collision detected\n";

                exit(0);
                return; // Exit after detecting collision
            }
        }
    }
}
```

```cpp

void randomizeCars() {
    for (int i = 0; i < 100; i++) {
        for (int j = 0; j < carList.size(); j++) {
            carList[j].changeDirection(rand() % 4 + 1);
        }
        checkCollision();
        displayRemaining();
        cout<<"Iteration "<<i+1<<"\n\n"; // Add some space between each iteration
    }
}
```



```cpp
int main() {
    createPoints();
    createCars();
    checkCollision();
    randomizeCars();
    return 0;
}

```

## OUTPUT 6:




*** drawings were used for illustration purposes only and are not part of the actual output**

# The End.

**Submitted to : Engr. Sohail**

**Date: 23/9/2023**

**22MDSWE197**

**Class No. 04**

---

**ABDUL HASEEB**