Program Of

“Car Management and Race Simulation Using C++”

```
#include <iostream>
#include <string>
#include <queue>
#include <cstdlib>
#include <ctime>
using namespace std;

// ---------- Car Class ----------
class Car {
public:
   string name;  // Car's name
   int speed;    // Car's speed
   int price;    // Car's price

// Constructor with default values
   Car(string n = "", int s = 0, int p = 0)
      : name(n), speed(s), price(p) {}

// Compare two cars by speed (used in sorting)
   bool operator<(const Car &c) const {
      return speed < c.speed;
   }
// Compare two cars by name (used in search)
   bool operator==(const Car &c) const {
```

```
    return name == c.name;   // Compare by name
  }
};

// ---------- Linked List Node ----------
struct Node {
  Car car;
  Node *next;
  Node(Car c) : car(c), next(nullptr) {}
};

// ---------- Garage Class ----------
// Supports adding, deleting, searching, upgrading, selling, sorting, etc.
class Garage {
  Node *head;

public:
  Garage() : head(nullptr) {}

// Destructor - deletes all nodes to free memory
  ~Garage() {
    while (head) {
      Node *temp = head;
      head = head->next;
      delete temp;
    }
  }
```

// Add a car at the beginning (linked list)

```
void addCar(Car c) {
    Node *newNode = new Node(c);
    newNode->next = head;
    head = newNode;
    cout << "Car added to garage.\n";
}
```

// Display all cars

```
void displayCars() {
    Node *temp = head;
    if (!temp) {
        cout << "Garage is empty.\n";
        return;
    }
    cout << "\n--- Cars in Garage ---\n";
    while (temp) {
        cout << "Name: " << temp->car.name
             << ", Speed: " << temp->car.speed
             << ", Price: " << temp->car.price << endl;
        temp = temp->next;
    }
}
```

// Search car by name

```
Car* searchCar(string n) {
    Node *temp = head;
    while (temp) {
        if (temp->car.name == n)
```

```
            return &(temp->car); // if found return pointer
        temp = temp->next;
    }
    return nullptr; // not found
}

// Sort cars by speed using bubble sort
void sortCars() {
    if (!head || !head->next)
        return;

    bool swapped;
    do {
        swapped = false;
        Node *temp = head;
        while (temp->next) {
            if (temp->car < temp->next->car) {
                swap(temp->car, temp->next->car);
                swapped = true;
            }
            temp = temp->next;
        }
    } while (swapped);
    cout << "Cars sorted by speed.\n";
}

// Add all cars from garage into a race queue
void addAllCarsToQueue(queue<Car> &raceQueue) {
```

```
    Node *temp = head;
    if (!temp) {
      cout << "Garage is empty. No cars to add.\n";
      return;
    }
    while (temp) {
      raceQueue.push(temp->car);
      temp = temp->next;
    }
    cout << "All cars added to race queue.\n";
  }

// Delete a car by name
  bool deleteCar(string n, queue<Car> &raceQueue) {
    if (!head) return false;

    bool deleted = false;

// Case 1: Car to delete at the head
    if (head->car.name == n) {
      Node *toDelete = head;
      head = head->next;
      delete toDelete;
      deleted = true;
    }
// Case 2: Car is somewhere else in the list
else {
      Node *temp = head;
```

```
        while (temp->next) {
            if (temp->next->car.name == n) {
                Node *toDelete = temp->next;
                temp->next = temp->next->next;
                delete toDelete;
                deleted = true;
                break;
            }
            temp = temp->next;
        }
    }
    if (!deleted) return false;

// Also remove car from race queue
    queue<Car> newQueue;
    while (!raceQueue.empty()) {
        Car c = raceQueue.front();
        raceQueue.pop();
        if (c.name != n) {
            newQueue.push(c);
        }
    }
    raceQueue = newQueue;

    return true;
}
// Upgrade a car's speed by increment++
bool upgradeCar(string n, int increment) {
```

```
    Car* c = searchCar(n);
    if (c) {
      c->speed += increment;
      cout << "Car " << c->name << " upgraded! New speed: "
          << c->speed << " km/h\n";
      return true;
    }
    return false;
  }
// Sell a car
  int sellCar(string n, queue<Car> &raceQueue) {
    if (!head) return 0;

    int price = 0;
// Case 1: Car is at the head
    if (head->car.name == n) {
      price = head->car.price;
      Node *toDelete = head;
      head = head->next;
      delete toDelete;
    }
// Case 2: Car is elsewhere in list
 else {
      Node *temp = head;
      while (temp->next) {
        if (temp->next->car.name == n) {
          price = temp->next->car.price;
          Node *toDelete = temp->next;
```

```
            temp->next = temp->next->next;
            delete toDelete;
            break;
          }
          temp = temp->next;
        }
      }

      if (price == 0) return 0;
// Remove car from race queue
      queue<Car> newQueue;
      while (!raceQueue.empty()) {
        Car c = raceQueue.front();
        raceQueue.pop();
        if (c.name != n) {
          newQueue.push(c);
        }
      }
      raceQueue = newQueue;

      return price;
    }
};

// ---------- Random Car Name Generator ----------
// Generate car name from a predefined list + random number
string randomName() {
    string names[] = {"Falcon", "Storm", "Viper", "Raptor", "Shadow", "Blaze"};
```

```
    int idx = rand() % 6;
    return names[idx] + to_string(rand() % 100);
}

// ---------- Main Program ----------
int main() {
    Garage g;                       // Garage object (linked list of cars)
    queue<Car> raceQueue;           // Queue for race participants
    int choice;                     // Menu choice
    int balance = 100000;           // Player's money

    srand(time(0));

// Main Menu
    do {
        cout << "\n------ Car Game Menu ------\n";
        cout << "1. Add Car to Garage\n";
        cout << "2. Add Multiple Cars to Garage\n";
        cout << "3. Generate Random Car\n";  // moved here
        cout << "4. Display All Cars\n";
        cout << "5. Search Car by Name\n";
        cout << "6. Sort Cars by Speed\n";
        cout << "7. Add All Cars to Race Queue\n";
        cout << "8. Exit Car from Race Queue\n";
        cout << "9. Upgrade Car (Increase Speed)\n";
        cout << "10. Delete Car by Name\n";
        cout << "11. Start Race\n";
        cout << "12. Sell Car for Money\n";
```

```
        cout << "13. Show Balance\n";
        cout << "14. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

// ---------- MENU OPERATIONS ----------
        if (choice == 1) {
// Add one car manually
            string name;
            int speed, price;
            cout << "Enter Car Name: ";
            cin >> name;
            cout << "Enter Speed: ";
            cin >> speed;
            cout << "Enter Price: ";
            cin >> price;
            g.addCar(Car(name, speed, price));
        }
        else if (choice == 2) {
// Add multiple cars
            int n;
            cout << "How many cars do you want to add? ";
            cin >> n;
            for (int i = 0; i < n; i++) {
                string name;
                int speed, price;
                cout << "\nEnter Car " << (i+1) << " Name: ";
                cin >> name;
```

```
            cout << "Enter Speed: ";
            cin >> speed;
            cout << "Enter Price: ";
            cin >> price;
            g.addCar(Car(name, speed, price));
        }
    }
    else if (choice == 3) {
// Generate and add a random car
        string name = randomName();
        int speed = 100 + rand() % 201;
        int price = 5000 + rand() % 50001;
        g.addCar(Car(name, speed, price));
        cout << "Random Car Generated: " << name
             << " | Speed: " << speed
             << " | Price: " << price << endl;
    }
    else if (choice == 4) {
// Display all cars
        g.displayCars();
    }
    else if (choice == 5) {
// Search car by name
        string name;
        cout << "Enter Car Name to Search: ";
        cin >> name;
        Car *c = g.searchCar(name);
        if (c)
```

```
            cout << "Found: " << c->name << " | Speed: " << c->speed
                 << " | Price: " << c->price << endl;
        else
            cout << "Car not found.\n";
    }
    else if (choice == 6) {
// Sort cars by speed
        g.sortCars();
    }
    else if (choice == 7) {
// Add all cars to race queue
        g.addAllCarsToQueue(raceQueue);
    }
    else if (choice == 8) {
// Remove the front car from race queue
        if (raceQueue.empty()) {
            cout << "Race queue is empty. Nothing to exit.\n";
        } else {
            Car exited = raceQueue.front();
            raceQueue.pop();
            cout << "Car " << exited.name << " exited from race queue.\n";
        }
    }
    else if (choice == 9) {
// Upgrade car by increasing speed
        string name;
        int inc;
        cout << "Enter Car Name to Upgrade: ";
```

```
        cin >> name;
        cout << "Enter Speed Increment: ";
        cin >> inc;
        if (!g.upgradeCar(name, inc))
            cout << "Car not found.\n";
    }
    else if (choice == 10) {
// Delete car by name
        string name;
        cout << "Enter Car Name to Delete: ";
        cin >> name;
        if (g.deleteCar(name, raceQueue))
            cout << "Car deleted from garage.\n";
        else
            cout << "Car not found in garage.\n";
    }
    else if (choice == 11) {
// Start the race
        if (raceQueue.empty()) {
            cout << "Race queue is empty!\n";
        } else {
            cout << "\n  Race Started!\n";
            Car winner;
            bool first = true;

// Go through all cars in race queue
            while (!raceQueue.empty()) {
                Car c = raceQueue.front();
```

```
            raceQueue.pop();
            cout << "Car " << c.name << " running at "
                 << c.speed << " km/h!\n";

// Update winner if this car is faster
            if (first || c.speed > winner.speed) {
               winner = c;
               first = false;
            }
          }
          // Announce race winner
          cout << " Race Finished!\n";
          cout << " Winner: " << winner.name
               << " with speed " << winner.speed << " km/h!\n";
        }
      }
      else if (choice == 12) {
// Sell car and add money to balance
        string name;
        cout << "Enter Car Name to Sell: ";
        cin >> name;
        int money = g.sellCar(name, raceQueue);
        if (money > 0) {
          balance += money;
          cout << "Car sold for " << money << "! New Balance: " << balance << endl;
        } else {
          cout << "Car not found in garage.\n";
        }
```

```
        }
        else if (choice == 13) {
// Show current balance
            cout << "Current Balance: " << balance << endl;
        }
}
 while (choice != 14); // Exit loop


    return 0;
}
```