**C# OOP Basics Exam Preparation I – Need For Speed**

**Overview**

You have a task to write a software program, which will create a virtual model of the cars, races and their development.

1. **Structure**

The main structure of the program should include the following elements:

**Cars**

A basic car has the following properties: brand (string), model (string), yearOfProduction (int), horsepower (int), acceleration (int), suspension (int), and durability (int). Each different type of car adds to those properties. Here are the types:

PerformanceCar – a car made for racing. Might be a little ugly, but it is a rocket inside.

Has addOns (Collection of strings). (by default – empty)

Increases its given horsepower by 50%.

Decreases its given suspension by 25%.

ShowCar – a car made for showing off. Looking cool out there, bro.

Has stars (int). (by default – 0)

**Races**

The basic race has the following properties: length (int), route (string), a prizePool (int), and participants (Collection of Cars),

CasualRace – just a normal race. Several beasts’ warfare, spreading their roars throughout the roads.

DragRace – a drag race. An engine fray. The ideal gear shifting will be the winner in this.

DriftRace – a drift race. Don’t you wish your girlfriend was drifty like me.

**Garage**

Garage – The Garage is that place where all the cars stay, when they are not racing. The Garage also provides the ability to modify parked car

Has parkedCars (Collection of Cars).

**Constructors**

Implement all class constructors, with the parameters in the EXACT given order and the EXACT given types.

String Representation

Implement ToString() methods for every Car class. You can see the requirements in the Output Section below.

1. **Business Logic**

The Controller Class

The business logic of the program should be concentrated around several commands. Implement a class called CarManager, which will hold the main functionality, represented by these methods:

void Register(int id, string type, string brand, string model, int yearOfProduction, int horsepower, int acceleration, int suspension, int durability)

string Check(int id)

void Open(int id, string type, int length, string route, int prizePool)

void Participate(int carId, int raceId)

string Start(int id)

void Park(int id)

void Unpark(int id)

void Tune(int tuneIndex, string addOn)

**Commands**

The commands in the CarManager class should represent the functionality to the input commands of the user. Here are the input commands you need to accept from the user input.

register {id} {type} {brand} {model} {year} {horsepower} {acceleration} {suspension} {durability}

REGISTERS a car of the given type, with the given id, and the given stats. The car type will be either “Performance” or “Show”.

check {id}

CHECKS details about the car with the given id. RETURNS a string representation of the car.

open {id} {type} {length} {route} {prizePool}

OPENS a race of the given type, with the given id, and stats. The race type will be either “Casual”, “Drag” or “Drift”.

participate {carId} {raceId}

ADDS a car as a participant in the given race. Once added, a car CANNOT turn down a race or be REMOVED from it.

start {raceId}

INITIATES the race with the given id. RETURNS detailed information about the race result.

park {carId}

PARKS a car by a given id in the garage.

unpark {carId}

UNPARKS the car with the given id from the garage.

tune {tuneIndex} {tuneAddOn}

Tunes the currently parked CARS with the given index and the given add-on. You should increase a car’s horsepower by the given index, and the suspension, by HALF of the given index. 150 tuneIndex = 150 increase in the horsepower and 75 increase in suspension. If the car is a ShowCar it should increase its stars by the given tuneIndex. Upon tuning, a PerformanceCar adds the given add-on to its collection of add-ons.

**Functionality**

Cars and Races are the main entities in the program’s functionality. They have no suitable way to be ACCESSED, which is why, upon registration, they are given an Id. The Id will be a simple integer. There is NO need for Cars and Races to know their Ids. The CarManager is the one that controls the main logic, which is why it is the only class which needs to know of every car and race’s id.

When you register a car, you store it in such a way, so that you can access it by id. You can then make the car participate in a race, or select it in the garage. There are several RULES that you must follow:

Once a car has been ADDED as a participant in a race, it CANNOT be PARKED in the garage, UNTIL the race is OVER.

IGNORE any attempt to park a racer car.

A car, which has been PARKED in the garage, CANNOT participate in a race.

IGNORE any attempt to include a parked car in a race.

IGNORE any attempt to TUNE cars, when there are NO PARKED cars in the garage.

SINGLE car CAN participate in MANY races.

A race CANNOT start without ANY participants.

A race CAN start with LESS than three participants.

Performance points (PP) determine every race’s winners. PP are either Overall Performance, Engine Performance or Suspension Performance. Here are the different formulas:

A CasualRace determines its winners based on their Overall Performance (OP) (in DESCENDING order). Overall Performance, of EACH CAR, is calculated by the following formula:  
(horsepower / acceleration) + (suspension + durability)

A DragRace determines its winners based on their Engine Performance (EP) (in DESCENDING order). Engine Performance, of EACH CAR, is calculated by the following formula:  
(horsepower / acceleration)

A DriftRace determines its winners based on their Suspension Performance (SP) (in DESCENDING order). Suspension Performance, of EACH CAR, is calculated by the following formula:  
(suspension + durability)

Depending on the different TYPE of RACE, different type of POINTS are calculated for the racers. In the end all points are presented as Performance Points (in the OUTPUT).

When you OPEN a race, you register it – this provides the functionality to add participants to it.   
When you START a race, the winners are calculated immediately, PRINTED as output, and the race becomes CLOSED (you CANNOT add any more participants in it, and you CANNOT start it again).

If TWO cars have the SAME result, participant registered before the other comes FIRST.

The 1st place winner takes 50 % of the race’s prize pool.

The 2nd place winner takes 30 % of the race’s prize pool.

The 3rd place winner takes 20 % of the race’s prize pool.

You need to take in account ONLY the FIRST 3 players, AFTER you’ve ordered them in descending order, by the corresponding criteria.

In case a race has LESS than 3 participants, you should print only them, as winners. The prizes remain the SAME.

In case a race has NO participants, you should print “Cannot start the race with zero participants.”, and IGNORE the command.

1. **I / O (Input / Output)**

**Input**

The input will come in the form of commands, in the format specified above.

The input sequence ends when you receive the command “Cops Are Here”.

**Output**

Two elements generate output in the program’s functionality:

The “check” command should RETURN a String representation of the CAR with the GIVEN ID:

“{brand} {model} {yearOfProduction}

{horsepower} HP, 100 m/h in {acceleration} s

{suspension} Suspension force, {durability} Durability”

If the car is a PerformanceCar, you must print “Add-ons: {add-ons}”, on the last line – each add-on separated by a comma and a space “, “. In case there are NO add-ons, print “None”.

If the car is a ShowCar, you must print “{stars} \*”, on the last line.

The “start” command should RETURN a String representation the RACE with the GIVEN ID:

“{route} - {length}

1. {brand} {model} {performancePoints}PP - ${moneyWon}

2. {brand} {model} {performancePoints}PP - ${moneyWon}

3. {brand} {model} {performancePoints}PP - ${moneyWon}”

1, 2 and 3 – being the 1st, 2nd and 3rd participants (the winners).

If there are LESS than 3 participants, print as much as there are.

In case there are NO participants, print “Cannot start the race with zero participants.”, and IGNORE the command.

Constrains

All integers in the input will be in range [0, 100000].

All strings in the input may consist of any ASCII character, except SPACE

So that the input is easily processed.

There will be NO invalid input lines, or invalid (non-existent) Ids.

Note that throughout the program, you are working ONLY with INTEGERS.

Each mathematical or logical action performed on numeric data, should be performed between INTEGERS.

Note: 50% of X is EQUAL to (X \* 50) / 100.

Note: Decrease means DECREASE… 100 decreased by 25% = 100 – (100 \* 25) / 100 = 100 – 25 = 75.

**Examples**

|  |  |
| --- | --- |
| Input | Output |
| register 1 Performance BMV M92 2013 300 4 150 75  register 2 Show Maserati Levante 2015 400 6 250 100  register 3 Performance Nissan GT-R 2017 550 4 300 100  register 4 Performance McLaren P1 2016 650 2 400 200  register 5 Performance Trabant 601 1988 2000 1 10000 1000  open 1 Drag 10 BeverlyHills 50000  open 3 Casual 20 NewYork 100000  participate 1 1  participate 2 1  participate 3 1  participate 4 1  participate 5 1  participate 1 3  participate 2 3  participate 3 3  participate 4 3  participate 5 3  check 5  start 1  start 3  Cops Are Here | Trabant 601 1988  3000 HP, 100 m/h in 1 s  7500 Suspension force, 1000 Durability  Add-ons: None  BeverlyHills - 10  1. Trabant 601 3000PP - $25000  2. McLaren P1 487PP - $15000  3. Nissan GT-R 206PP - $10000  NewYork - 20  1. Trabant 601 11500PP - $50000  2. McLaren P1 987PP - $30000  3. Nissan GT-R 531PP - $20000 |
| register 3 Show Porsche Carrera 2017 550 4 300 100  register 4 Performance McLaren P1 2016 650 2 400 200  register 5 Performance Trabant 601 1988 2000 1 10000 1000  open 1 Casual 20 Manhattan 100000  open 2 Drag 14 Washington 100000  participate 5 1  participate 5 2  park 3  park 4  park 5  start 2  tune 150 Turbo  tune 100 Nitrous  tune 50 Tires  participate 3 1  check 3  check 4  unpark 4  participate 4 1  start 1  Cops Are Here | Washington - 14  1. Trabant 601 3000PP - $50000  Porsche Carrera 2017  850 HP, 100 m/h in 4 s  450 Suspension force, 100 Durability  300 \*  McLaren P1 2016  1275 HP, 100 m/h in 2 s  450 Suspension force, 200 Durability  Add-ons: Turbo, Nitrous, Tires  Manhattan - 20  1. Trabant 601 11500PP - $50000  2. McLaren P1 1287PP - $30000 |
| register 1 Performance BMV M92 2013 300 4 150 75  register 2 Show Maserati Levante 2015 400 6 250 100  register 3 Performance Nissan GT-R 2017 550 4 300 100  register 4 Performance McLaren P1 2016 650 2 400 200  register 5 Performance Trabant 601 1988 2000 1 10000 1000  open 1 Drag 10 BeverlyHills 50000  open 3 Casual 20 NewYork 100000  Cops Are Here |  |

1. **Bonus**

The modern racers like different types of races. If you are really good at writing software, then your employers would like to hire you for some more work.

Your task is to implement classes for 2 extra SPECIAL races:

TimeLimitRace

Is INITIALIZED with an EXTRA PARAMETER – goldTime (int).

CircuitRace

Is INITIALIZED with an EXTRA PARAMETER – laps (int).

Both races, have an extra parameter, aside from the normal races. The parameter is received, from the user input as last parameter, when OPENING one of these types of races.

Logic

The TimeLimitRace can only have 1 participant. ANY attempt to add more participants to it should be IGNORED.  
The participant has a Time Performance (TP), which is calculated by the following formula:

raceLength \* ((participantHorsepower / 100) \* participantAcceleration)

Depending on the Time Performance, the player earns “Gold”, “Silver” or “Bronze” time:

TP <= raceGoldTime – Racer has earned Gold Time and earns 100% of the prizePool.

TP <= raceGoldTime + 15 – Racer has earned Silver Time and earns 50% of the prizePool.

TP > raceGoldTime + 15 – Racer has earned Bronze Time and earns 30% of the prizePool.

The String representation of the TimeLimitRace is in the following format:

“{route} – {length}

{participantBrand} {participantModel} – {participantTimePerformance} s.

{participantEarnedTime} Time, ${wonPrize}.”

The CircuitRace is almost like a normal race, with the difference that it has laps and 4 winners in total. The winners are determined by Overall Performance (OP) like in CasualRace.

1st place earns 40% of the prizePool.

2nd place earns 30% of the prizePool.

3rd place earns 20% of the prizePool.

4th place earns 10% of the prizePool.

The special thing about this race is … That EVERY lap DECREASES the DURABILITY of EACH participant by (length \* length). The String representation of the CircuitRace is in the following format:

“{route} - {length \* laps}

1. {brand} {model} {performancePoints}PP - ${moneyWon}

2. {brand} {model} {performancePoints}PP - ${moneyWon}

3. {brand} {model} {performancePoints}PP - ${moneyWon}

4. {brand} {model} {performancePoints}PP - ${moneyWon}”

**Examples**

|  |  |
| --- | --- |
| Input | Output |
| register 1 Performance Mitsubishi Lancer-Evo 2010 400 5 200 100  register 2 Performance Nissan Z370 2012 500 4 300 200  register 3 Show BMW i8-Spyder 2016 600 3 400 300  register 4 Performance Lamborghini Aventador 2017 1000 2 500 300  register 5 Show Ford Mustang-Shelby 1970 400 5 700 200  open 1 Circuit 10 SofiaStreets 100000 5  open 2 Circuit 2 SofiaAirport 10000 2  participate 1 1  participate 2 1  participate 3 1  participate 4 1  participate 5 1  start 1  check 1  check 2  check 3  check 4  check 5  Cops Are Here | SofiaStreets - 50  1. Lamborghini Aventador 925PP - $40000  2. Ford Mustang-Shelby 480PP - $30000  3. BMW i8-Spyder 400PP - $20000  4. Nissan Z370 112PP - $10000  Mitsubishi Lancer-Evo 2010  600 HP, 100 m/h in 5 s  150 Suspension force, -400 Durability  Add-ons: None  Nissan Z370 2012  750 HP, 100 m/h in 4 s  225 Suspension force, -300 Durability  Add-ons: None  BMW i8-Spyder 2016  600 HP, 100 m/h in 3 s  400 Suspension force, -200 Durability  0 \*  Lamborghini Aventador 2017  1500 HP, 100 m/h in 2 s  375 Suspension force, -200 Durability  Add-ons: None  Ford Mustang-Shelby 1970  400 HP, 100 m/h in 5 s  700 Suspension force, -300 Durability  0 \* |
| register 1 Performance Mitsubishi Lancer-Evo 2010 400 5 200 100  register 4 Performance Lamborghini Aventador 2017 1000 2 500 300  park 4  tune 1000 Turbo  unpark 4  open 1 TimeLimit 5 SofiaAirport 100000 260  open 2 TimeLimit 5 Malibu 10000 240  start 1  participate 4 1  participate 1 1  participate 4 2  start 1  start 2  Cops Are Here | Cannot start the race with zero participants.  SofiaAirport - 5  Lamborghini Aventador - 250 s.  Gold Time, $100000.  Malibu - 5  Lamborghini Aventador - 250 s.  Silver Time, $5000. |