**A: DOCUMENTATION**

**Name and Title : EV App Portal**

A C++ exception is a response to an exceptional circumstance that arises while a program is running, such as an attempt to divide a number by zero. Exception Handling in C++ is a process to handle runtime errors. We perform exception handling so the normal flow of the application can be maintained even after runtime errors.

**Requirements**: Create an EV App portal that displays details of Electric vehicles such as charging percentage, tyre pressure & available range to its owners. The portal should also provide a few additional functionalities which are described in the "Member Methods" section below.

To complete this task, create the following in your program:

1. **Enumerations:**
   1. enum HEALTH : It contains three literal values - GOOD,AVERAGE & POOR.
   2. enum CONFIDENCE: It contains three literal values - HIGH,MEDIUM & LOW.
2. **Class**
   1. Create a class named EvApp which has the following members:

|  |  |  |
| --- | --- | --- |
| **Data Member Name** | **Data Type** | **Description** |
| evRegNumber | string | Stores the registration number of the electric vehicle. |
| evChargePct | int | Stores the battery charge percentage of the EV vehicle. |
| evTyrePressureLevel | int | Stores a value between 1 to 100 to indicate what is the level of tyre pressure in the vehicle. (1 means lowest level possible, 100 means perfect value for tyre pressure as per manufacturer's guidelines) |
| isEvServiceDue | bool | Indicates whether the electric vehicle requires a service to be scheduled. |
| evAvailableRange | int | Indicates the available range(in Kms). It will always be a value rounded off to the nearest integer & will always be greater than 0 |
| evBatteryHealth | enum | Indicates the health of the battery pack of the EV and represents its operational condition. It must be a value from the HEALTH enum defined above. |

Complete all the below member functions of the EvApp class as per the given instructions:

|  |  |  |  |
| --- | --- | --- | --- |
| **Method Name** | **Arguments** | **Return Type** | **Description** |
| Parameterized Constructor | String,int,int,  bool,int,enum HEALTH | none | This method is used to initialize a new EvApp object |
| Destructor | none | none | This method is used to destroy the object at the end of scope. |
| getEvRegNumber | none | string | Returns the *evRegNumber* of the vehicle |
| getEvCharge | none | string | Returns the *evChargePct* of the vehicle |
| getEvTyrePressure | none | int | Returns total *evTyrePressureLevel* of the vehicle. |
| getEvServiceDueStatus | none | none | Returns the isEvServiceDue value for the current vehicle, |
| getEvAvailableRange | none | char | Returns the *evAvailableRange* value for the current vehicle, |
| isEvReady | none | bool | Refer to the section 'EV Ready' below. |
| tripConfidence | float | Enum CONFIDENCE | Refer to the section 'Trip Confidence' below |
| findEvIssue | none | string | Return a string value based on conditions explained in the "EV Issue" table. |

**EV Ready**

Return **true** if the following conditions are satisfied

* The *evChargePct* is greater than or equal to 30.
* The *evTyrePressureLevel* is greater than or equal to 70.
* The *isEvServiceDue* property of the of the object is set to 'false'

If any of the above three conditions is not satisfied, an exception of *const char\** type must be thrown with a message as defined below

|  |  |  |  |
| --- | --- | --- | --- |
| **Sr. No** | **Type of Exception** | **Exception throw in Condition** | **Corresponding Message** |
| 1 | Low battery charge | evCharPct value is less than 30 | Battery percentage is too low |
| 2 | Low tyre Pressure | evTyrePressureLevel value is less than 70 | Tyre pressure is not optimum |
| 3 | Service due | isEvServiceDue value is true | Car requires immediate servicing |

**Note**: Write conditions in isEvReady function to check for exceptions in the same sequence (1, 2 and 3) i.e check for Low Battery charge exception first.

**Trip Confidence**

To decide the trip confidence, calculate a value 'factor' by using float value *distance* passed to the tripConfidence function as follows:

factor = distance / evAvailableRange

If factor is greater than 1.0, throw a *const char\** type exception with message "Distance is beyond of available range". For other values of factor, trip confidence is determined as per the following table

|  |  |  |  |
| --- | --- | --- | --- |
| **evChargePct** | **factor** | | |
| **Less than 0.5** | **0.5 to 0.74** | **0.75 to 1** |
| **Greater than 80** | HIGH | MEDIUM | LOW |
| **61 to 80** | MEDIUM | MEDIUM | LOW |
| **30 to 60** | MEDIUM | LOW | LOW |
| **Below 30** | Low battery Charge Exception (Refer to EV Ready  Section for the corresponding message) | | |

**Note**: Write conditions in tripConfidence function to check for Distance beyond Range exception first. 'Low Battery Charge' exception should be thrown only when *factor* is less than or equal to 1.0 and *evChargePct* value is below 30.

**EV Issue**

The findEvIssue function should call isEvReady function on the current EvApp instance and must return a *const char\** valye based on following conditions

* If *isEvReady* function does NOT throw an exception, return value is "No EV issue detected".
* If *isEvReady* function throws a 'Low tyre Pressure' exception, return value is "Possible integrity issue. Please get tyres checked".
* If *isEvReady* function throws a 'Service Due' exception, return value is "Servicing for car is due".
* If *isEvReady* function throws a 'Low battery charge' exception AND the *evBatteryHealth* for current ***EvApp*** instance is POOR function, return value is "Battery low. If battery is not charging, get battery replaced".

**Test Cases:**

The above-mentioned requirements have been considered for creating test cases. Below Table

gives details of test cases to be used for validating written code.

|  |  |  |  |
| --- | --- | --- | --- |
| **Test**  **Case**  **No.** | **Test Case Description** | **Input Values** | **Expected Output** |
| 1. | To validate parameterized constructor functionality of *EvApp* class. | EvApp ev1("LH1623",27,80,false,250,HEALTH::GOOD); | LH1623,27,80,false,250,  HEALTH::GOOD |
| 2 | To validate *isEvReady* functionality of *EvApp* class | EvApp ev1("LH1623",56,80,true,250,HEALTH::AVERAGE);  ev1.isEvReady() | Exception with the message  Battery percentage is too low |
| 3. | To validate isEvReady functionality of EvApp class | EvApp ev1("LH1623",56,80,true,250,HEALTH::AVERAGE);  ev1.isEvReady() | Exception with message  Car requires immediate servicing |
| 4 | To validate *tripConfidence* functionality of EvApp class. | EvApp ev1("LH1623",96,80,false,250,HEALTH::AVERAGE);  ev1.tripConfidence(25) | HIGH |
| 5 | To validate *tripConfidence* functionality of EvApp class | EvApp ev1("LH1623",96,80,false,250,HEALTH::AVERAGE);  ev1.tripConfidence(260.0) | Exception with message  Distance is beyond of available range |
| 6 | To validate *findEvIssue* functionality of EvApp class | EvApp ev1("LH1623",34,66,false,250,HEALTH::AVERAGE);  ev1.findEvIssue(); | Possible integrity issue. Please get tyres checked |