CP2019s Project 1 : Car Simulator ver. 2050

Document changes: (CTRL+F to search for the changes)

- (0421) - IDE restriction in Due Date section.
- (0426) - Additional second move mode example
- (0426) - Case with Energy & Oxygen both depletion
- (0428) - Solar panel description & Report content
- (0429) - Detailed energy description on second move mode example
- (0430) - Case with Energy & Oxygen both depletion with **arrival**
- (0430) - Extra credit print
- (0501) - Exiting program instruction (Section VI)
- (0502) - Final status related comments
- (0503) - TC input change / some error changes
- (0504) - WARNING: **NO Correlation of humidity and energy in ocean environment**
- (0505) - The text graphic placement within the display
- (0506) - About code clarity
- (0507) - Class / Functions / Variables , *example executable file uploaded*

Geneva Motor Show 2017 - Dream come true?

Cp2019s - DCSLAB

I. Overview

The first project on Computer programming is car simulation on various situations. There are three different types of car modes and environment: car driving on a road, airplane flying in a sky, and submarine under the ocean. Your job is to create three different car mode and make those cars to drive on ten different test cases which will be provided. Depending on the car modes, there are different factors to consider while driving, so be aware of constraints and restrictions.

II. Contents

A. Vehicle mode & factors

There are three modes in car simulator. Each mode has attributes that represent current status of vehicle. It is advised to make parent class as ‘Vehicle’ and implements other classes as child of parent class. (NOT MANDATORY)

[Car]

- Speed
- Energy
- Temperature
- Humidity
- Solar panel recharge

Car mode has 4 factors (speed, energy, temperature, humidity) and public function (solar panel recharge). All of the attributes are defined private, so you have to make get/set functions to deal with the situation. When in car mode, the solar charging system is activated to charge energy.

[Airplane]

- Speed
- Energy
- Temperature
- Humidity
- Altitude
- Oxygen rate

- Air density

Airplane mode has 7 factors (speed, energy, temperature, humidity, altitude, oxygen rate, air density). All of the attributes are defined private, so you have to make get/set functions to deal with the situation.

[Submarine]

- Speed
- Energy
- Temperature
- Depth
- Light(Always on)
- Oxygen rate
- Water flow

Submarine mod has 6 factors (speed, energy, temperature, depth, oxygen rate, water flow) and public function(light). All of the factors are defined private, so you have to make get/set functions to deal with the situation.

(0504) As written above, humidity does not affect energy in submarine mode.

B. Environments (Road / Sky / Ocean)

In Road environment, there are 2 external factors (temperature, humidity).
When vehicle faces the road, 2 factors set by initial values of test cases.
Only car mode go through the road environment.
In Sky environment, there are 3 external factors (temperature, humidity, air density).
When vehicle faces the sky, 3 factors set by initial values of test cases.
Only airplane mode go through the sky environment.
In ocean environment, there are 2 external factors (temperature, water flow).
When vehicle faces the ocean, 2 factors set by initial values of test cases.
Only submarine mode go through the ocean environment.

C. Initial Values and the Relationships between variables

- Vehicle initial energy: 1000 / initial oxygen level: 100
- Car default speed: 80km / hr
- Airplane default speed: 900km / hr

- Submarine default speed: 20km / hr

Temperature affects energy. (Celcius)
when 0 < temperature < 40, energy loss 5
temperature >= 40, energy loss 10
temperature = 0, energy loss 8

Humidity affects energy.
when humidity < 50, energy loss 5
humidity >= 50, energy loss 8

(so, in case of car mode, with temp 20 C and humidity 50 -> 13 energy loss every 50km)

> “Unit” is defined as following: **50km for road, 10km for ocean, 1000km for sky**. (See Implementation Section III)

Humidity affects solar panel recharge.
The charging system only operates with a humidity < 50

Solar panel recharge affects energy.
after recharging, energy get 200 (Energy cannot be over 1000) (! Check for constraint)

Air density affects speed.
when air density >= 30, speed loss 200
air density >= 50, speed loss 300
air density >= 70, speed loss 500

When vehicle meet the road, oxygen rate set to 100

Altitude affects oxygen rate.
oxygen rate loss 10 per altitude 1000

Depth affects oxygen rate.
oxygen rate loss 5 per depth 50

Light affects energy.
when in submarine mode, the light is always on, it reduces energy 30 **EVERY 10km movement!**

Water flow affects speed.

when water flow >= 30, speed loss 3

water flow >= 50, speed loss 5

water flow >= 70, speed loss 10

! IMPORTANT
All loss should be “ADDED UP”
In case of speed loss, oxygen loss, energy loss, etc...

D. Test Cases

(0503) TC input will be stored in string data type. (If you want to have different data type other than string, you can change it as long as the data type can hold “[...]...[...]”-like format.)

```
Int main() {

    string TC1 = “[.........],[.....],....,[.....]”;
    string TC2 = “...”
    ...
    string TC10 = “[....],.........,[....]”;

    ….
```

Test case form: R means road, O means ocean, S means sky.

Each test case consists of duration of course, initial values, and if there are any, unexpected situations.

Example: [R50T20H30],[S3000T10H5A2000D20],[X],[O80T0D100W100],[R150T30H50]

The meaning of this test case: The course starts with 50km of road with temperature value 20 and humidity value 30. After 50km, you will face sky with 3000km. Within the sky, the temperature value is 10, humidity(H) is 5, altitude(A) is 2000, air density(D) is 20. After that, you meet unexpected situation(X). You can deal with this unexpected situation or just ignore it. (your choice) After unexpected situation, you will face 80km of ocean.. and so on.

[R50T20H30] : Road (50km) / Temperature (20 C) / Humidity (30)

[S3000T10H5A2000D20] : Sky (3000km) / Temp (10 C) / Humidity (5) / Altitude (2000) / air Density (30)

[X] & [Y] : unexpected situations (if you are not going for the extra credit, **just ignore these signs**)

[O80T0D100W100] : Ocean (80km) / Temp (0 C) / Depth (100m) / Water flow rate (100)

~~E. File I/O(0503)~~

~~Text files containing each test case will be provided. (ex. TC1.txt, TC2.txt …)~~

~~You have to read each files using I/O function(ifstream), and process the input string to properly function.~~

~~Between the environment, there are useful indicators.(ex. ‘,’, ‘[‘, ‘]’, ‘R’, ‘T’, ‘H’ ...)~~

~~Using test case string, you can make full journey that vehicle can go through.~~

~~Read all ten TC files in the start of the program and use them whenever a test case is called.~~

III. Implementation

- From the start to the end, your vehicle will be driving a course (from a test case file). The vehicle should change as it faces different situations.
- The vehicle starts on road and will continue the course until three different endings: when the vehicle successfully arrives at the goal line, when the energy is totally gone, when the oxygen level is zero.
- **Either energy or the oxygen level goes to zero will make the vehicle stop and the journey will be forced to be finished.**

- There are ten test cases (ten different courses to choose) and you have to ask for a test case to run.
    - (Screenshot in Section VI.)
- After choosing the course, ask for two different options to choose: drive until next “unit” or “status change”.
    - (Screenshot in Section VI.)
- “Unit” is defined as following: **50km for road, 10km for ocean, 1000km for sky**.
- “Status change” is defined as following: **the arrival, car mode change(from car to airplane etc), empty energy, zero oxygen level, car stop(* - for extra credit).**

- **First move mode: Drive for next unit**
    - You will drive the vehicle for a single “unit”.
    - Print out the vehicle status after driving a “unit”.
    - (Screenshot in Section VI.)
    - If energy or oxygen level goes below zero, stop the vehicle and print the status.
    - (If there are too little energy or oxygen to go to the next “unit” (but not zero), the vehicle can make to the next “unit” and the remaining energy or oxygen will be gone.)

- **Second move mode: Drive until the next status change**
    - You will drive the vehicle as long as it has no “status change”
    - If there are enough resources(energy/oxygen), you can go as far as the environment does not change.
    - (Suppose you have enough energy) There is a road with 500km. With the first mode, you should print 10 times to drive all 500km. But in this second mode, you can drive the whole 500km without printing the status in between.

    - (0426/0429) Another example, There is a road with 300km remaining. You have energy only to go for 250km(More energy left for 200km and zero energy after 250km). In this case, you should only go until 250km.
    - It means, the car should stop if resources gone empty, so you should stop the car right after the empty resource status.

- After each move, you should print the current status.
    - (Screenshot in Section VI.)
- The current status should be printed DIFFERENTLY with the vehicle mode.
    - Car does not has oxygen level status.
    - Airplane does not has depth status.
    - Submarine only has depth status. Etc….
    - The point is that the output should show correct and only related current status (Screenshot in Section VI. look for the difference between car/airplane/submarine)

- Journey will end in three different cases: the arrival, oxygen failure, energy failure.
- Print the final status on Distance / Energy / Oxygen values (Exact format is on the screenshot in Section VI.)
- After the whole journey, you should print “Blackbox” which records all data of [“car mode”,”energy”,”oxygen”,”speed”]
    - For car mode, record oxygen level as 100 (Always)
    - This blackbox records only when status changes. (from car to airplane etc…)

IV. Constraints

Code constraints

- The journey starts with the “car mode”. (TC also always starts with Road.)
- On car mode, oxygen level is **ALWAYS 100.**
- From airplane & submarine to car mode, oxygen level goes up to 100.
- From airplane to submarine, oxygen level continues (NOT go up to 100). *vise versa
- **Speed does NOT affect the move “unit”. (50km per move regardless of car speed)**
- No change of temperature / humidity / altitude / air density / depth / water flow as long as the environment does not change. (It’s possible to have a different value at the next environment change)
- Solar panel recharge function ONLY activates (0428) **INSTANTLY RIGHT after** the vehicle is changed to car mode.
    - (Ex. one activation right after the vehicle changes from Airplane/Submarine to Car)
    - (Ex2. Airplane -> Car (function activates) -> Submarine -> Car(function again activates) -> … ->)

Program constraints

- Compile with **C++11**
- **All inputs (menu selections, mode selections) are all valid. NO Exception handling required.**

V. For Extra credit

A. Unexpected factor will occur in between the courses (in test case)

At the first of the program, you should ask for the handle mode whether you will consider unexpected situations or not.

```
Mode Select(1 for EXTRA, 2 for NORMAL) :
```

There are two different unexpected events: X, Y.

X : For road, you will face reindeer, eagle for sky, and shark for ocean. This event occur with [X] mark on test cases. This event has “20%” chance of complete stop of the vehicle. If the vehicle survives the event, 100 energy is decreased (and will continue).

Y : The vehicle will get damaged without a reason. This event occur with [Y] mark on test cases. This event has “35%” chance of complete stop of the vehicle. If the vehicle survives the event, following options will occur on 50% chance. For car on the road, solar panel will be destroyed. For airplane on sky and submarine under the ocean, 30 oxygen level will be decreased.

(0430) - When car stops by X or Y, you may print “Successfully moved to next 0 km” and print final status right after OR you can print nothing and print final status.

B. Text graphic of the progress

- Print the graphic after each session report. (After each move mode)
- For the vehicle, use “@”.
- For road, use “=“. Each character represents 50km.
- For ocean, use “~”. Each character represents 10km.
- For sky, use “^”. Each character represents 1000km.
- **(0505) Print the graphic right after session print. (current status or the final status in case of current status: print the graphic right after the status and before Next move message)**

Cp2019s - DCSLAB

VI. Result examples

Initial State

```
PJ1.홍길동.2018-00000
Choose the number of the test case (1~10) :
```

Choosing the test case (print the initial status)

(0501) - Input 0 → Exit program. (No need to print anything just return 0 on main class)

TC5 : [R500T20H20],[S3000T10H5A2000D30],[O80T0D100W100] **(not same with actual TC5)**

```
Choose the number of the test case (1~10) : 5
Test case #5.

Current Status: Car
Distance: 0 km
Speed: 0 km/hr
Energy: 1000
Temperature: 20 C
Humidity: 20
Next Move? (1,2)
CP-2018-00000>
```

Select the first move mode :

Select the second move mode : (Drive all the way until the next status change)

```
CP-2018-00000>2
Successfully moved to next 450 km
Current Status: Car
Distance: 500 km
Speed: 80 km/hr
Energy: 900
Temperature: 20 C
Humidity: 20
Next Move? (1,2)
CP-2018-00000>
```

Select the first move mode: (Car -> Airplane) (Notice that there are **more features** to show)

```
CP-2018-00000>1
Successfully moved to next 1000 km
Current Status: Airplane
Distance: 1500 km
Speed: 700 km/hr
Energy: 890
Oxygen Level: 80
Temperature: 10 C
Humidity: 5
Altitude: 2000 m
Air Density: 30
Next Move? (1,2)
CP-2018-00000>
```

Select the second move mode:

```
CP-2018-00000>2
Successfully moved to next 2000 km
Current Status: Airplane
Distance: 3500 km
Speed: 700 km/hr
Energy: 870
Oxygen Level: 40
Temperature: 10 C
Humidity: 5
Altitude: 2000 m
Air Density: 30
Next Move? (1,2)
CP-2018-00000>
```

Select the first move mode: (Airplane -> Submarine)

```
CP-2018-00000>1
Successfully moved to next 10 km
Current Status: Submarine
Distance: 3510 km
Speed: 10 km/hr
Energy: 832
Oxygen Level: 30
Temperature: 0 C
Depth: 100 m
Water Flow: 100
Next Move? (1,2)
CP-2018-00000>
```

Select the second move mode: (not enough oxygen → DIE)

```
CP-2018-00000>2
Successfully moved to next 30 km
Final Status:
Distance: 3540 km
Energy: 718
Oxygen Level: 0

!FINISHED : Oxygen failure
Blackbox:
Mode: Car > Airplane > Submarine
Energy Level: 900 > 870 > 718
Oxygen Level: 100 > 40 > 0
Speed: 80 > 700 > 10
--------------------
```

!FINISHED : Oxygen failure //(when no oxygen)
!FINISHED : Energy failure //(when no energy)
!FINISHED : Arrived //(when finished the full course)
(0426) - When both energy and oxygen gone in the same time, **print energy failure**.
(0430) - When all energy and oxygen gone with the vehicle arrival, **print Arrived.**
!FINISHED : Vehicle stop // * for extra credit
(0502) When in times with the vehicle can no longer move, you may print "successfully moved to next 0 km" or nothing.

After Printing "----------", go back to the option to choose the after test case.
(use loop for the whole process)

VII. Report

1. Development/Implementation Environment
2. Specific code description (detailed enough to know about the functions and mechanism of whole process)
3. Troubleshooting points while implementing your code (trial and error, things to be careful of, etc)
4. Screenshots of the interactive console (From the start to the end of whole course, add different screenshots in between status changes and various events)
5. **There is NO report length limitation.**
6. **(0428) If you completed extra credit part, please mention it on the first page.**

VIII. Due date

- **Due on May 10th 23:59.**
- **Submit the whole file zipped into one single zip file(your whole source code and the report) to ETL**
- **CAUTION : 20% deduction per day after the submit due date.**
- **NO submit possible after two days (Must submit before May 13th 00:00)**
- **Compile error : ZERO POINT**
- (0421) Code file type : **ONLY** *.cpp are allowed. (NO *.sln, CMake etc)
- (0421) IDE restriction : no restriction, but submit your code into ONE .cpp format. **(Merge all your files into one cpp file)**
- (0421) Code format : “project_1_2019_00000.cpp”
- (0421) Report file type : **ONLY** docx, hwp, pdf are accepted.
- (0421) Report format : “project_1_2019_00000_홍길동.xxx”

IX. Evaluation

Classification	Criteria	Points
General	Program Execution I/O Format	15
	Code clarity	15
Test case	Total 10 test cases (each - partial point exist)	50 (5)
Report	Clarity	5
	Code / Screenshot Description	15
Error	**Compile Error**	**-100**
Total		**100**
Extra Credit	Unexpected Situation handling	20
	Graphic progress	**10**

(0506) - For code clarity section, we will focus on the flow and “logical coordination” between classes/functions/structs/variables. It is important to organize your code so that your friends can understand easily. Place and name your classes/functions/variables decisively.

Ex) If you decide to make some class functions, you should use them otherwise there is no reason to put them there.

Ex) If the variable names are so vague that you cannot follow the code without looking at the declaration part again and again, the code is not clear at all.

In sum, ORGANIZE your code from top to bottom.

(0507) - It is decided NOT TO LIMIT in any implementation on vehicle classes and environment. EX) If you want to use humidity in vehicle class even though the submarine subclass doesn’t need one, you can use it. 어느정도 융통성을 두고 채점하도록 하겠습니다.

(0506) - In case of compile error, **please double check** before you submit your code. NO mistakes are allowed this time.

X. FAQ

All questions from email will be answered here. Since this document is on-line editable, please check this FAQ before mailing.

1. In the Extra credit section 2, the car character should be in between roads. It means:

|===^^~~| (Original environment map)

|@===^^~~| (START)

|=@==^^~~| (After one move) **(It does NOT take up the space of environment)**

|===^^~~@| (FINISH)

2. Error on cout in Korean language within VS code environment.
 Use English initials for the error. Ex) 홍길동 → HGD

3. For solar panel recharge, is there any case with consecutive road environment? **NO**

4. For move mode, there is a condition stated that it is assumed the vehicle moved to the next unit without sufficient energy, is that condition also applied to the second move mode? **NO – check for the update (0426)**

5. When both energy and oxygen deplete on the same time, **you SHOULD print "energy failure"** as the reason vehicle stopped.

6. Is there any off-unit input related to altitude & depth (such as 2500 in altitude)? **NO**.

7. For text graphic, should we include final status in the session report? **YES**. Print the last position of the vehicle in the final status print.

8. For Unexpected case X, Y, is there any symbols to put for the text graphic? **NO**. Two extras are independent.

9. For unexpected case Y, when the vehicle survives from the event, the effect takes place corresponding to the vehicle state "**RIGHT BEFORE THE EVENT**".

10. For unexpected case X, Y, the events takes place "**ONLY IN BETWEEN**" R/S/O environment status changes.

11. I defined Vehicle as the base class and Car class inherited from Vehicle. In this case, is it possible to inherit some member variables from the base class such as speed and energy member variable from Vehicle? **YES**.

Also, is it possible to state member variables protected instead of private? **NO.**

12. (This question is on hold - considering since different OS has different file manage system) Test case files “might” be stored inside the code since file I/O is all different for different systems. (NOT SURE YET)

~~13. **TC1 ~ TC10 will not be opened until May 5th**.~~

14. All numbers we use on this project can be handled through **integer data type** (as far as I can think of - if not, email me).

15. TC1 ~ TC10 file length is **no more than 2000 characters**.

16. Put the text graphic **right before** “Next move? (1, 2)”.

17. For the value changes when status change, **ONLY one change** occurs with speed. It means, when you fly on sky with 1000m altitude, the speed gets 200km/h lower for the first time, **NO MORE** decrease afterwards. Vehicle changes to the airplane -> 900km/h initial -> altitude 1000m -> fly all the way in 900-200km/h **= 700km/h no accumulation (not 500km/h after one unit and not 300km/h after two units and so on)**.

18. In case of [X], is it right to decrease 100 energy of the vehicle for 80% chance? **YES**

If then, should I print 100 energy lower when printing the last finished status? **NO** - Record the blackbox when the environment is over. Assume the lost energy is **dealt on the next** environment.

19. When the vehicle right **BEFORE** the event is car, solar panel goes dead on total 0.65*0.5 chance, and for other modes, oxygen level changes.

And, yes, solar panel **will not work at all in all cases** after solar panel goes dead.

20. (0503) In case of [S…][X][O…], when X makes the vehicle stop by “depleting the energy”, final status should be **AIRPLANE** & !Finished Energy failure.

When X makes the vehicle COMPLETE STOP by 20%, final status is still **AIRPLANE**, but the !Finished message should be “Vehicle stop”. (Sorry for the mistake T^T…)

21. Obviously, humidity inside the ocean is **100**. - No effect on submarine.

22. There is no need to print additional message when encountering event X or Y.

(No need to print “Vehicle met reindeer” and so on) & (No need to print “Energy 100 decreased”)

23. When vehicle stops right after X or Y, you don’t print Next Move(1,2)?

The vehicle don’t go any further. **Take care of X or Y** without having next move action.

(설명이 모호할 수 있어 한글로 추가설명합니다. [S][X][O] 환경을 예시로, S 후에 Next Move(1,2)? 를 쓰기 전에 X를 만나면 바로 처리를 하기 바랍니다. 그 후 vehicle이 멈추면(확률때문에 멈추거나 에너지/산소 등이 떨어져서 멈추는 경우 모두) final status 를 출력하기 바랍니다. 즉, final status 에서는 **successfully moved to next 0 km(QNA 30번참고)** 인 셈입니다.)

+추가설명: 어떠한 이벤트가 발동된 상태에서 어떠한 이유로든 자동차가 멈추게 된다면 다음 환경은 무시한 채로 현재 상황까지를 블랙박스에 저장하여 출력하기 바랍니다.

24. Example based QNA: case on **2nd move mode**. 6 units are left to go and 53 energy left where you need 13 energy for every unit. You are supposed to go 5 units (since 13 * 4 = 52, 1 energy left for one more move).

25. Priority Message : Arrived > Energy depletion > Oxygen depletion

(도착과 동시에 에너지와 산소 모두 바닥났다면, 도착한 것으로 판정합니다.)

26. Blackbox **does not** record any changes from the unexpected events.

27. When Y happens and oxygen gets all depleted, **you are FINISHED** even though the next environment is road which you can refill the oxygen.

28. You **only get one input for extra credit mode when your program starts**.

29. NO test case with the same environment coming right after that environment. (No S → S, O → O…)

30. In cases when you can no longer move (with various reasons) you may print” succ… moved to next 0 km” or nothing.

31. For extra credit mode selection, the setting is fixed all the way until the end of the program.

32. In a case with NOT enough (but not zero) energy for next unit, it is POSSIBLE to move for the next unit, but **NOT POSSIBLE TO MOVE AFTERWARDS EVEN THOUGH ENVIRONMENT CHANGE.**

33. (0507) Check for the cp2019 course page for example file. I didn’t take care of unexpected events there, only the text graphics. Take a look. (The exact numbers might not be so accurate on the program. Just for a vague guide.) ***The TCs on the file are just some examples***.

(will be added after e-mail questions…)