



Lab 4 - Used-Car Dappreciation

Due on 2/12/2024 at 11:59PM

Introduction:

The US Federal Reserve Bank has spent the better part of the last year attempting to rein in **consumer** [↗](https://www.bls.gov/news.release/cpi.nr0.htm) **price** [↗](https://www.bls.gov/opub/ted/2022/food-prices-up-10-8-percent-for-year-ended-april-2022-largest-12-month-increase-since-november-1980.htm) **inflation** [↗](https://www.nber.org/papers/w30116). Their job may almost be done. What caused the rampant explosion in inflation? Well, the jury is still out, but one cause is clearly disruptions to the supply chain as a result of countries freezing their manufacturing sectors to protect the population from the terrible health impact of the Covid-19 pandemic. The cost of everything from toothbrushes to grain to ads on Facebook has increased wildly. The price of used cars has been particularly affected by inflationary forces. From the beginning of the pandemic to August 2023, **the average price of a used car has increased by 30%** [↗](https://www.wsj.com/personal-finance/car-prices-might-be-unsustainable-for-buyers-18d7b395?st=9hkjsij0rtkfhly&reflink=desktopwebshare_permalink). **In fact, during the height of pandemic lockdowns, some used cars today were worth more than they were when they were new** [↗](https://www.wsj.com/articles/why-the-car-in-your-garage-may-be-gaining-value-11644575402?st=c78ycnr57p6itd5&reflink=desktopwebshare_permalink). The average (retail) price of a used car is now **\$31,030** [↗](https://www.copilotsearch.com/index-report-12-2023/) (or \$26,091 **according to a different estimate** [↗](https://www.cnbc.com/2024/01/08/used-car-prices-high-but-expected-to-be-stable-in-2024.html)). Although an index of wholesale prices for used cars has declined sequentially month-over-month for the past two years, **it is still 33% higher than it was in December 2019** [↗](https://www.prnewswire.com/news-releases/manheim-index-used-vehicle-values-fall-further-in-2023-with-values-now-down-21-from-all-time-high-reached-in-2021-302028646.html).

Although AI has recently taken center stage, **decentralized** [↗](https://www.wsj.com/articles/defi-is-helping-to-fuel-the-crypto-market-boomand-its-recent-volatility-11622712602?st=m1oc5g758dbo80o&reflink=desktopwebshare_permalink) **finance** [↗](https://www.wsj.com/articles/defi-is-cryptos-wall-street-without-a-safety-net-11631611945?st=xdog70knhklpu3q&reflink=desktopwebshare_permalink) continues to gain its fair share of headlines. The proponents of decentralized finance (**DeFi** [↗](https://www.bloomberg.com/news/articles/2020-08-26/why-defi-utopia-would-be-finance-without-financiers-quicktake?sref=bx4jJ3RK)) have promised that the technology will make it easier and cheaper for people to access the financial system and lower the cost of essential economic tools like loans and remittances. DeFi exists as decentralized applications (dapps) on the blockchain. They are programmed according to smart contracts and allow people to interact with one another and arrange economic transactions without a human intermediary.

It was the confluence of increasing inflation and a surge in used-car prices that led the Trusted Ternary Advisors group to launch a DeFi marketplace to facilitate used-car sales and purchases. In the marketplace, sellers can list their car for a price determined by the application and users can buy the car for the pre-determined price without any haggling. It's like **Carvana**  (<https://www.carvana.com/>) meets **Buy It Now**  (<https://www.ebay.com/help/selling/listings/selling-buy-now?id=4109>) (from eBay). The buyers and sellers interact with the marketplace through the *dappreciation* application.


From the seller's perspective, *dappreciation* will help determine whether their car is eligible for listing on the platform. And, if it is eligible, *dappreciation* will calculate the car's non-negotiable resale value.

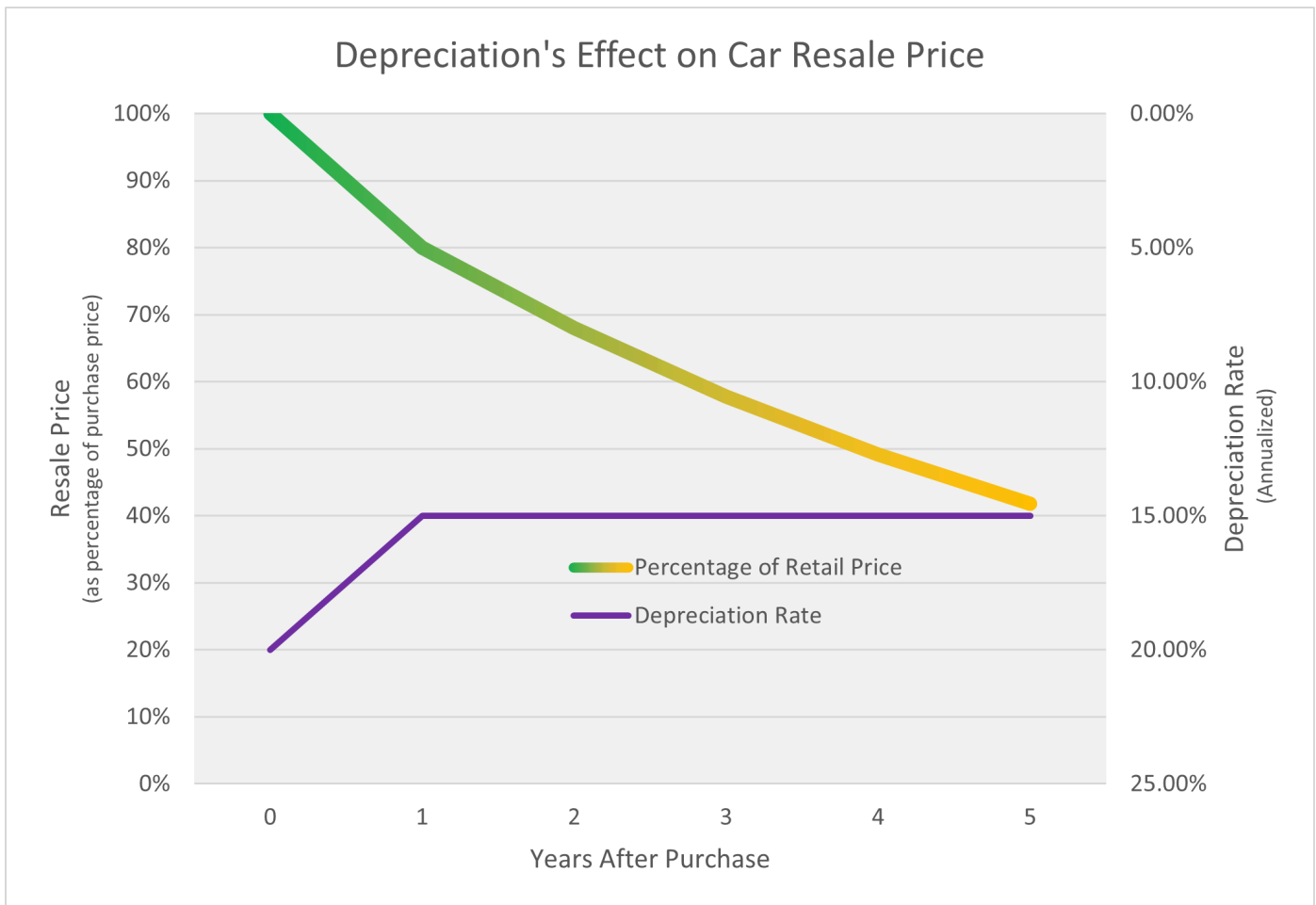
A car's eligibility for listing on the marketplace is based on two factors:

1. It's age; and
2. How many time's its been wrecked.

If the car is (strictly) less than 11 years old *and* has been in three or fewer wrecks, the car is eligible! If the car does not meet those conditions, it cannot be listed. Once *dappreciation* has determined the car's eligibility, it will calculate its resale value. The resale value of the car is based on several factors:

- Its age;
- The number of times it has been wrecked; and
- Whether it has premium options (e.g., leather seats, heated steering wheel, Harmon Kardon audio system, etc.)

The car's age plays the single biggest role in determining resale value. If the car is less than 5 years old, it is worth 85% of its manufacturer's suggested retail price (MSRP). If the car is between 5 and 8 years old (inclusive), it is worth 63% of MSRP. If the car is 9 or 10 years old, it is worth 45% of MSRP. We'll call this percentage the car's *baseline resale percentage*. (All data here is from actual depreciation rates calculated by **CarEdge**  (<https://caredge.com/compare/depreciation#linear>).



The number of accidents to which a car has been party has a non-linear impact on the resale price. We will call this the *wreck penalty percentage*. If the car has been in a single accident, the resale price drops by *an additional 2%* off the baseline resale percentage. If it has been involved in two (2) accidents, the price drops by 10% points from the baseline resale percentage. Finally, if it has been in three (3) accidents, the price drops by 20% points from the baseline resale percentage.

If the car was purchased new with additional premium options, then the resale value *increases*. The presence of premium options adds five (5) percentage points to the car's resale value. We will call this the *premium percentage*.

Subtracting the *wreck penalty percentage* from the *baseline resale percentage* and adding any applicable *premium percentage* will give us a *final resale percentage*. The actual resale price is simply the car's original MSRP multiplied by the *final resale percentage*.

After doing its calculations, *dappreciation* will output a message telling the user either a) that their vehicle is ineligible for the platform, or b) the resale price that they can expect by conducting a sale on the marketplace.

Program Design Task:

As my dad always said, "If I cut down a tree in the forest at least I can hear it fall." Before you start writing code, please write the pseudocode or draw a flow chart for your implementation of the *dappreciation* application.

Program Design Requirements:

Your pseudocode or flow chart must describe the entirety of the solution. You may choose to write the flow chart or the pseudocode at any level of detail but remember that this is a tool for you!

Your pseudocode or flow chart *must* include a description of how you plan to determine

1. Whether the car is eligible for resale on the platform, based on its age and accident history; and
2. The car's resale value based on its age, accident history and MSRP.

Programming Task:

Your programming task is to implement the *dappreciation* application and document the implementation of a single supporting function.

Your program will prompt the user for the following information (in *this order*):

1. The model year of the car up for resale
2. The number of accidents the car has been in
3. Whether the car has any premium options
4. The car's original MSRP

The user must be able to enter the car's model year as a whole number, number of accidents as a whole number, and original MSRP as a floating-point number. The user must be able to enter whether the car has premium options or not by typing or , respectively. In your program you may assume that the user always enters valid inputs.

Given the user's input about the car they intend to list for sale, the application will calculate a) whether the car is eligible for listing on the platform and, if eligible, b) the no-haggle resale price.

Calculating the resale price must be done according to the parameters defined above and repeated below (in outline format):

Baseline Resale Percentage

Age of vehicle	
Between 0 and 4 years old (inclusive)	85%
Between 5 and 8 years old (inclusive)	63%
9 or 10 years old	45%

Wreck Penalty Percentage

Number of Accidents	
0 accidents	0%
1 accident	2%
2 accidents	10%
3 accidents	20%

Premium Percentage

Premium Options?	
Yes	5%
No	0%

Example Calculations and dappreciation Output:

2019 Toyota Tundra

85%	Vehicle age
- 10%	2 accidents
+ <input type="text" value="5%"/>	Premium options
80%	Final Resale Percentage

\$33,265 ➞ <https://www.caranddriver.com/toyota/tundra-2019> * .80 = \$26612.00

dappreciation will list your car for \$26612.00.

Note: If you are in the market for an SUV, be the first to bring this note to my attention and you're in luck! I have a used SUV that I'm trying to get rid of. Seriously.

2013 Porsche 911 Carrera Cabriolet

45%	Vehicle age
- 20%	3 accidents
+ <input type="text" value="0%"/>	Premium options
25%	Final Resale Percentage

\$94,100 ➞ <https://www.kbb.com/porsche/911/2013/> * .25 = \$23525.00

dappreciation will list your car for \$23525.00.

1965 Shelby Cobra

N/A	Vehicle age
- 0	0 accidents
+ <input type="text" value="5%"/>	Premium options
N/A	Final Resale Percentage

Unfortunately your car is ineligible for the dappreciation platform.

Note: Your program will be tested against other test cases. Your program must compute properly in all cases in order to receive full points! Check the output of the autograder to make sure that your program behaves as expected. Read the autograder's output carefully!

Programming Requirements:

If you are a Windows-based C++ developer, start with this [skeleton](https://uc.instructure.com/courses/1657742/files/176806159?wrap=1)

(<https://uc.instructure.com/courses/1657742/files/176806159?wrap=1>)_ ↓

(https://uc.instructure.com/courses/1657742/files/176806159/download?download_frd=1) . If you are a macOS-based C++ developer, use this [skeleton](https://uc.instructure.com/courses/1657742/files/176806375?wrap=1)

(<https://uc.instructure.com/courses/1657742/files/176806375?wrap=1>)_ ↓

(https://uc.instructure.com/courses/1657742/files/176806375/download?download_frd=1) . If you are a Linux-based C++ developer, use this [skeleton](https://uc.instructure.com/courses/1657742/files/176806705?wrap=1)

(<https://uc.instructure.com/courses/1657742/files/176806705?wrap=1>)_ ↓

(https://uc.instructure.com/courses/1657742/files/176806705/download?download_frd=1) . These skeletons provide functions you may use to successfully complete this lab.

Using the skeleton provided skeleton code will make it much easier to implement the *dappreciation* application than it would otherwise. The skeleton code provides functions that will

Gather Input From the User:

- `car_model_year`: This function will prompt the user to enter the car's model year and return their input as an integer (`int`).
- `car_accidents`: This function will prompt the user to enter the number of accidents the car has been in and return their response as an integer (`int`).
- `car_msrp`: This function will prompt the user to enter the car's MSRP and return their response as a double (`double`).
- `car_has_premium_options`: This function will prompt the user to enter whether the car has premium options and return their response as a Boolean (`bool`); `true` indicates that the car has premium options and `false` indicates that there are no premium options.

Display Output To the User:

- `print_eligible_message`: This function takes a single parameter (the car's resale price) and prints it to the screen along with a nice message.
- `print_ineligible_message`: This function takes no parameters and prints a message letting the user know the disappointing news that their car is not suitable for the marketplace.

Besides generating the appropriate output depending on the input, a complete submission for this lab also contain comments explaining the operation of the implementation of the

`car_has_premium_options` function. Again, the `car_has_premium_options` function

1. Prompts the user to enter `yes` or `no` depending on whether the car has premium options;
2. Generates ("returns") `true` if the user entered `yes`; or
3. Generates `false` if the user entered `no`.

You must use a multi-line/block comment above `car_has_premium_options` to state *what* the function does and describe its return value (the meaning of the value *and* the type of the value).

You must use single-line comments throughout the function to explain its operation.

Your code must have the following characteristics:

1. All numeric values defined in this lab document and used in your code must be manipulated as properly typed constants. For example, the only time that the floating-point number `63` should appear in your code is during an assignment to a constant variable whose type is a `double`, as in (for example) `const double TIER_TWO_BASELINE_RESALE_PERCENTAGE{63};`
2. Blocks of code associated with if or else conditions must be enclosed in `{}`s.

Assumptions:

1. Alternatively, you may assume that the user enters valid input *or* that the functions provided to gather input from the user work properly.
2. You may assume that the current model year is 2023.
3. You may assume that output generated from the functions provided in the skeleton produce properly formatted output.

Getting Started

If I were completing this lab, the first thing that I would do is make sure that I understood how to use the provided functions to gather input from the user. Because the car's age plays an outsize role in determining whether or not the car is eligible for resale on the *dappreciation* platform, let's ask the user for the car's model year first (it also helps that the lab specification mandates that is the first bit of input we collect from the user!).

The provided `car_model_year` function is like the `get_int` function from the previous lab, except better! Whereas `get_int` did *not* prompt the user for their input (leaving that burden at your feet), `car_model_year` *does* prompt the user. What exactly does that mean? Well, let's find out together. Inside the main function of the skeleton, let's add a call to the function and see what happens:

```
int main() {  
    car_model_year();  
    return 0;  
}
```

When we run the code, we see output that prompts us to enter the car's model year:

```
What is the model year of the car?
```

Wonderful! As it stands, our code simply ignores the *return value* of the call to that function and exits. But, we will need to use the user's response in order to perform calculations so we should probably store it in a variable. Let's use our auto power and declare a variable (named, say, `user_model_year`) and store the model year that the user entered:

```
int main() {  
    auto user_model_year{car_model_year()};  
    return 0;  
}
```


If we execute our program now, the behavior will look the same but we know that it's a significant improvement from before. To guarantee that we properly captured the user's input, let's use the debugging trick from the previous lab and print out the contents of the variable:

```
int main() {  
    auto user_model_year{car_model_year()};  
    std::cout << "The user entered: " << user_model_year << "\n";  
    return 0;  
}
```

Run the program and make sure that it does what you think it should!

Great work!

The other functions provided for gathering input from the user behave similarly. From this base of code you can follow your pseudocode and complete the lab!

Critical Thinking Task:

Matryoshka dolls are the technical term for what are commonly referred to as stacking dolls.

According to Wikipedia [↗\(https://en.wikipedia.org/wiki/Matryoshka_doll\)](https://en.wikipedia.org/wiki/Matryoshka_doll), these dolls were first made in the 1890s. An incredibly small, wooden doll is placed inside a series of larger and larger wooden dolls -- together these dolls form a set. Every doll in the set, no matter the size, is painted to look the same. This property of self-similarity makes them like a real-world **fractal** [↗\(https://en.wikipedia.org/wiki/Fractal\)](https://en.wikipedia.org/wiki/Fractal)! Here are some pictures to give you a sense for what they look like:



The Matryoshka dolls are also referred to as nesting dolls. Smaller dolls are nested inside larger dolls. This past week we learned about a concept in C++ that relies on nesting. In this critical thinking component, you will describe how a teacher could use Matryoshka dolls to teach that concept.

Critical Thinking Requirement:

There are at least two different aspects of C++ that include concepts of nesting. You must write a *brief* (less than 200 words) lesson plan for how you would use Matryoshka in a classroom to teach *one* of those concepts to students learning the C++ language. Your lesson plan must include a

technical description of how the nesting operates in the C++ language and how it relates (physically) to the dolls.

Question Task:

For this lab you deployed your knowledge of using functions, benefited from abstraction but started to peer behind the curtain for the first time and employed selective execution. Was there anything surprising? Something that didn't make sense? Please feel free to ask anything related to anything that we've learned so far (or will learn)!

Question Requirement:

Submit a file named `question.pdf` which contains your question. Take particular notice of the fact that the question does not have to be subtle, or deep or philosophical for you to receive full points. The goal of the question is for me to be able to get a sense whether there are common questions about the material which would indicate where I failed to teach the material appropriately. You may *not* say that you understood all the material and have no questions. You *must* submit a question.

Deliverables:

- 1. The pseudocode describing the algorithm of your *dappreciation* program in PDF format (named `design.pdf`).
- 2. The C++ source code for your *dappreciation* application (named `dappreciation.cpp`).
- 3. A written response to the Critical Thinking Task prompt in PDF format (named `dolls.pdf`).
- 4. A written question about the material covered so far in this class in PDF format (name the file `question.pdf`).

Rubric:

Points	Task	Criteria
15	Design	Pseudocode uses algorithmic thinking to express a high-level description of how to implement the <i>dappreciation</i> application and meets the specified criteria.
15	Critical Thinking	Lesson plan includes a technical description of how the nesting operates in the C++ language and how it relates (physically) to the dolls.
10	Programming	Comments on the implementation of the <code>car_has_premium_options</code> function are thorough (follows the specifications above).

10	Programming	Program uses properly typed constants for each of the numeric values defined in this document for calculating the car's resale value on the marketplace.
5	Programming	Program uses <code>{ }</code> s to enclose blocks associated with if and else conditions.
20	Programming	Program correctly calculates its outputs for all test cases.
20	Programming	Program displays output according to the specified format for all test cases.
3	Programming	<code>main</code> function's purpose is described by a grammatically correct multi-line comment.
2	Question	Submission demonstrates a thoughtful attempt to ask a clarifying question about the class material. You may <i>not</i> say that you understood all the material and have no questions. You <i>must</i> submit a question.

Related Learning Objectives

1. Writing boolean expressions using relational and logical operators
2. Using if-statements to implement selective program execution
3. Recognizing self-similarity/nesting in parts of C++
4. Comparing strings and numbers using relational operations
5. Using pre-defined functionality through function calls