This is CS50x

OpenCourseWare

Donate (https://cs50.harvard.edu/donate) ☑ (https://community.alumni.harvard.edu/give/59206872)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
f (https://www.facebook.com/dmalan) ◯ (https://github.com/dmalan) ◎ (https://www.instagram.com/davidjmalan/) in
(https://www.linkedin.com/in/malan/) ⓘ (https://orcid.org/0000-0001-5338-2522) Ⓠ (https://www.quora.com/profile/David-J-Malan) ⓡ (https://www.reddit.com/user/davidjmalan) ⿻ (https://twitter.com/davidjmalan)

Lab 1: Population Growth

> You are welcome to collaborate with one or two classmates on this lab, though it is expected that every student in any such group contribute equally to the lab.

Determine how long it takes for a population to reach a particular size.

```
$ ./population
Start size: 100
End size: 200
Years: 9
```

Background

Say we have a population of `n` llamas. Each year, `n / 3` new llamas are born, and `n / 4` llamas pass away.

For example, if we were to start with `n = 1200` llamas, then in the first year, `1200 / 3 = 400` new llamas would be born and `1200 / 4 = 300` llamas would pass away. At the end of that year, we would have `1200 + 400 - 300 = 1300` llamas.

To try another example, if we were to start with `n = 1000` llamas, at the end of the year, we would have `1000 / 3 = 333.33` new llamas. We can't have a decimal portion of a llama, though, so we'll truncate the decimal to get `333` new llamas born. `1000 / 4 = 250` llamas will pass away, so we'll end up with a total of `1000 + 333 - 250 = 1083` llamas at the end of the year.

Getting Started

- Copy the "distribution code" (i.e., starter code) from cdn.cs50.net/2020/fall/labs/1/population.c (https://cdn.cs50.net/2020/fall/labs/1/population.c) into a new file in your IDE called `population.c`.
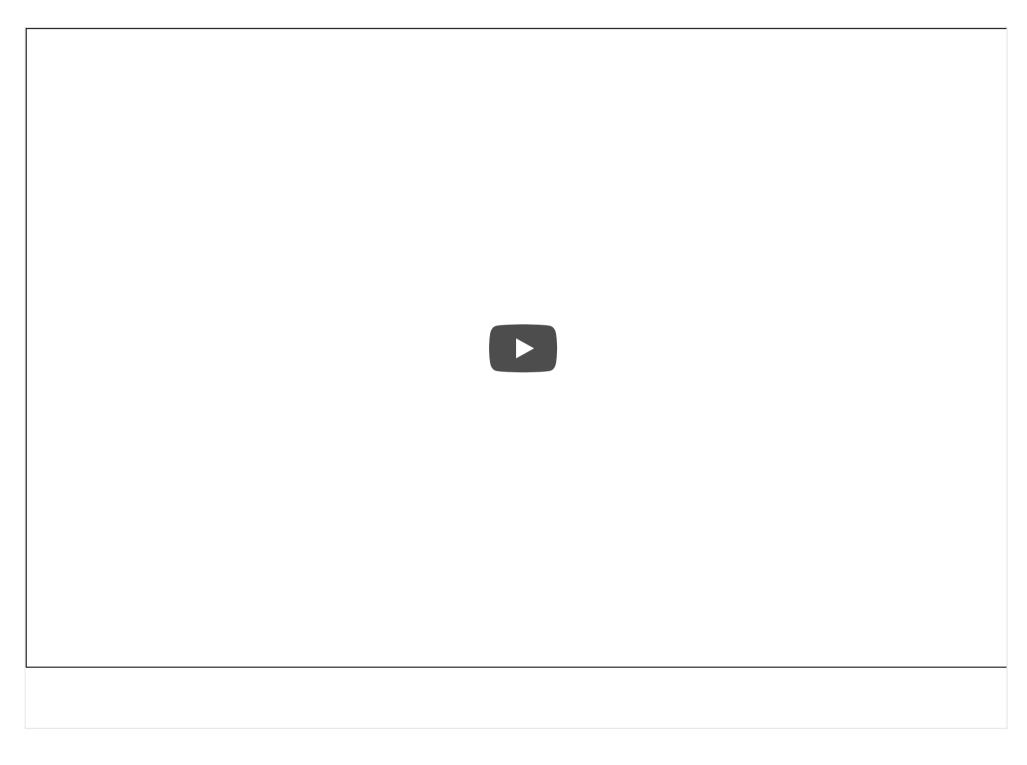
Implementation Details

Complete the implementation of `population.c`, such that it calculates the number of years required for the population to grow from the start size to the end size.

- Your program should first prompt the user for a starting population size.
  - If the user enters a number less than 9 (the minimum allowed population size), the user should be re-prompted to enter a starting population size until they enter a number that is greater than or equal to 9. (If we start with fewer than 9 llamas, the population of llamas will quickly become stagnant!)
- Your program should then prompt the user for an ending population size.
  - If the user enters a number less than the starting population size, the user should be re-prompted to enter an ending population size until they enter a number that is greater than or equal to the starting population size. (After all, we want the population of llamas to grow!)
- Your program should then calculate the (integer) number of years required for the population to reach at least the size of the end value.
- Finally, your program should print the number of years required for the llama population to reach that end size, as by printing to the

terminal `Years: n`, where `n` is the number of years.

Walkthrough

Hints

- If you want to repeatedly re-prompt the user for the value of a variable until some condition is met, you might want to use a `do ... while` loop. For example, recall the following code from lecture, which prompts the user repeatedly until they enter a positive integer.

```
int n;
do
{
    n = get_int("Positive Integer: ");
}
while (n < 1);
```

  How might you adapt this code to ensure a start size of at least 9, and an end size of at least the start size?

- To declare a new variable, be sure to specify its data type, a name for the variable, and (optionally) what its initial value should be.
  - For example, you might want to create a variable to keep track of how many years have passed.

- To calculate how many years it will take for the population to reach the end size, another loop might be helpful! Inside the loop, you'll likely want to update the population size according to the formula in the Background, and update the number of years that have passed.

- To print an integer `n` to the terminal, recall that you can use a line of code like

```
    printf("The number is %i\n", n);
```

to specify that the variable `n` should fill in for the placeholder `%i`.

How to Test Your Code

Your program should behave per the examples below.

```
$ ./population
Start size: 1200
End size: 1300
Years: 1
```

```
$ ./population
Start size: -5
Start size: 3
Start size: 9
End size: 5
End size: 18
Years: 8
```

```
$ ./population
Start size: 20
End size: 1
End size: 10
End size: 100
Years: 20
```

```
$ ./population
Start size: 100
End size: 1000000
Years: 115
```

▶ **Not sure how to solve?**

Execute the below to evaluate the correctness of your code using `check50`. But be sure to compile and test it yourself as well!

```
check50 cs50/labs/2021/x/population
```

Execute the below to evaluate the style of your code using `style50`.

```
style50 population.c
```

How to Submit

Execute the below, logging in with your GitHub username and password when prompted. For security, you'll see asterisks (`*`) instead of the actual characters in your password.

```
submit50 cs50/labs/2021/x/population
```