This is CS50

CS50's Introduction to Computer Science

OpenCourseWare

Donate (https://cs50.harvard.edu/donate)

David J. Malan (https://cs.harvard.edu/malan/)
malan@harvard.edu
(https://www.clubhouse.com/@davidjmalan) (https://www.facebook.com/dmalan) (https://github.com/dmalan) (https://www.instagram.com/davidjmalan/) (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://www.tiktok.com/@davidjmalan) (https://davidjmalan.t.me/) (https://twitter.com/davidjmalan)

Recursive `atoi`

Learning Goals

- Deepen an understanding of strings
- Practice creating recursive functions

Background

Image that you travel back in time to the 1970's, when the `C` programming language was first created. You are hired as a programmer to come up with a way to convert `string`s to `int`s. (You may have used a function just like this in Week 2, called `atoi` (https://manual.cs50.io/3/atoi)). You want to be thorough in your development process and plan to try several approaches before deciding on the most efficient.

In this problem, you will start with a simple implementation of `atoi` that handles positive `int`s using loops. You want to rework this into an implementation that uses recursion. Recusive functions can be memory intensive and are not always the best solution, but there

are some problems in which using recursion can provide a simpler and more elegant solution.

(Scroll to the bottom of this page to see what an implementation of `atoi` might actually look like.)

⊞ Hints

Demo

Getting Started

1. Log into code.cs50.io (https://code.cs50.io/) using your GitHub account.
2. Click inside the terminal window and execute `cd`.
3. Execute `wget https://cdn.cs50.net/2022/fall/labs/3/atoi.zip` followed by Enter in order to download a zip called `atoi.zip` in your codespace. Take care not to overlook the space between `wget` and the following URL, or any other character for that matter!
4. Now execute `unzip atoi.zip` to create a folder called `atoi`.
5. You no longer need the ZIP file, so you can execute `rm atoi.zip` and respond with "y" followed by Enter at the prompt.

Implementation Details

In the recursive version of `convert`, start with the last `char` and convert it into an integer value. Then shorten the `string`, removing the last `char`, and then recursively call

`convert` using the shortened string as input, where the next `char` will be processed.

Thought Question

Why do you need a base case whenever you create a recursive function?

How to Test Your Code

Your program should behave per the examples below.

```
atoi/ $ ./atoi
Enter a positive integer: 3432
3432
```

```
atoi/ $ ./atoi
Enter a positive integer: 98765
98765
```

No `check50` for this one!

To evaluate that the style of your code, type in the following at the `$` prompt.

```
style50 atoi.c
```

How to Submit

No need to submit! This is an optional practice problem.

A More Thorough Implementation

The actual version of `atoi` must handle negative numbers, as well as leading spaces and non-numeric characters. It might look something like this:

```c
#include <stdio.h>

// Iterative function to implement `atoi()` function in C
long atoi(const char S[])
{
    long num = 0;
    int i = 0, sign = 1;

    // skip white space characters
    while (S[i] == ' ' || S[i] == '\n' || S[i] == '\t') {
        i++;
    }
```

```
    // note sign of the number
    if (S[i] == '+' || S[i] == '-')
    {
        if (S[i] == '-') {
            sign = -1;
        }
        i++;
    }

    // run till the end of the string is reached, or the
    // current character is non-numeric
    while (S[i] && (S[i] >= '0' && S[i] <= '9'))
    {
        num = num * 10 + (S[i] - '0');
        i++;
    }

    return sign * num;
}

// Implement `atoi()` function in C
int main(void)
{
    char S[] = " -1234567890";

    printf("%ld ", atoi(S));

    return 0;
}
```

From techiedelight.com/implement-atoi-function-c-iterative-recursive (https://www.techiedelight.com/implement-atoi-function-c-iterative-recursive/).