Terminal

A Humbling Tale of Attempted Redemption

Lessons from previous projects

Portfolio

Find the purpose. Make it personal. Use Ruby. Make it repeatable.

Workbook

Use sub-goals. Work consistently.

Partner Programming

Talk through issues. Pseudocode. Get input. Help others with problems even if I haven’t encountered them yet.

A project from the past

Game on the train to school

Carriage number 1234 => 1 + 2 + 3 + 4 = 10, 1000 => 10 + 0 + 0 = 10

Tried to program it in Year 10, but it was very inelegant - regretted ever since

if n1 + n2 + n3 + n4 = 10 then print n1 + n2 + n3 + n4 =10

Perfect for loops

Recursion useful for future projects

First plans required too much computation

Trying to brute force every combination of operations would take too long with 13 slots and 8 operations including brackets and spaces. Needed so many to allow for occasions with 3 operations in a row e.g. 2)**(2

Found more optimised ways to solve including specific operations that are possible for each part of an equation e.g. an equation can never start with) or **

Began to solve for first solution instead of every solution, and used strategies like subgoals and investigating binary trees and preferential arguments e.g. if subgoal is 9 then it is not worth trying 9 ** 9 as no square root functions could bring total back down, but this requires too much code and is equally inelegant.

New plan to divide the tasks

Take advantage of objects in Ruby to avoid some operations

One method to perform calculations - V1 specifically for 4 digits, later versions to be more flexible

One method to check if the total equals 10 when no digits remain

One method to create patterns of parentheses e.g. a + b * c + d vs (a + b) *(c +d)

One method to create double digits

Capable of solving for all solutions not just first

Implement recursion if possible - lessons from solutions to Countdown game

Other features

Welcome message. Main heading displayed with ASCII art

Instructions - User can choose to view instructions

Ability for user to input numbers and receive solutions

Future

Ability for user to receive a random valid number, but only if it has a solution. User can make attempts to find valid solutions.

Time Management

Monday to Thursday - Research. Taking long walks and thinking about infinity

Friday - Built Calculator v1 - Expect v2 is within reach

Saturday - Built every other feature and tested the permanent ones

Sunday - Foetal position in shower + gain 2kg

Monday - Built calculator v2, abandoned recursion

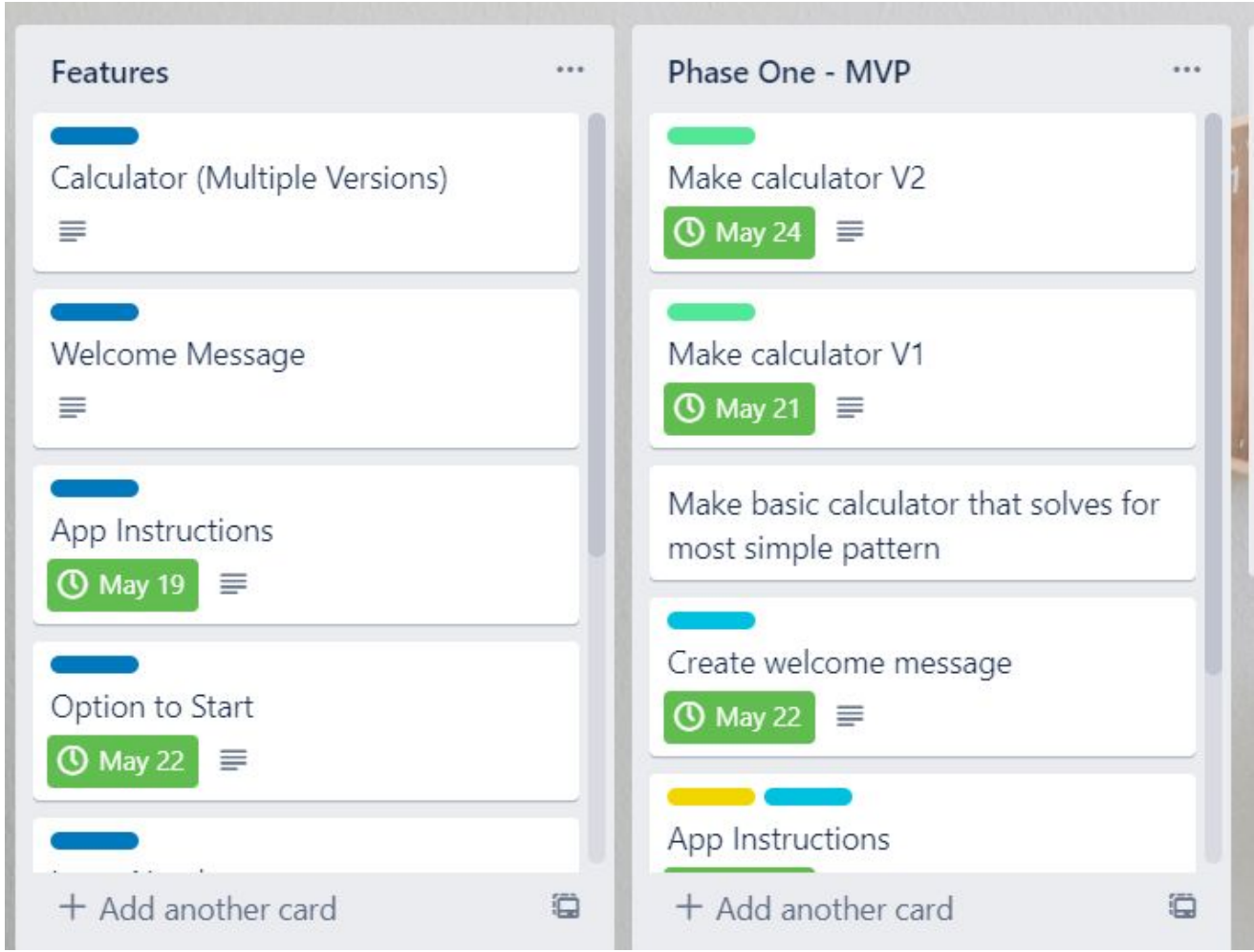
Tuesday - Present on MVP, Implement more gems, rework flow of features

Wednesday to Thursday - More testing and error handling, command line bash scripts and complete documentation

Phase One - Trello

Still learning how to make best use of Trello

Broke into 3 phases, but phase one is MVP for this project



Calculator v1

Hardcoded. While loops that solve for all operations on 4 digits. Requires 3 different indexes for 3 different arrays. Running total and next digit calculated

```
while ab_index < 5

    first2 = opps_ab[ab_index]
    fc_index = 0

        while fc_index < 5
            opps_fc = [op.add(first2, c), op.subt(first2, c), op.div(first2, c), op.mult(first2, c), op.exp
            (first2, c)]
            second2 = opps_fc[fc_index]
            sd_index = 0

                while sd_index < 5
                    opps_sd = [op.add(second2,d), op.subt(second2,d), op.div(second2,d), op.mult(second2,d)
                    , op.exp(second2,d)]
                    total = opps_sd[sd_index]
                    # sd_index += 1
                    if total == 10
                    # puts "#{a.to_i} a#{ab_index.to_i} #{b.to_i} b#{fc_index.to_i} #{c.to_i} c#{sd_index.
                    to_i} #{d.to_i} = 10"
                    puts "#{a.to_i} #{printarray[ab_index]} #{b.to_i} #{printarray[fc_index]} #{c.to_i} #
                    {printarray[sd_index]} #{d.to_i} = 10"

                    end
                    sd_index += 1
                end
            fc_index += 1
            end
```

Calculator v2

Still iteration but single index starting from -1

Five calculations done at once for each running total, creating new array of results.

Each operation and digit pushed to string

Running total starts from 0 so first operation removed from string e.g + 1 + 2 + 3 + 4 = 10

Check result separate method

```
def calculate(run_total, index, digits, returnString)

    # Until final digit
    if index < 3
        # Call allopps to calculate all possibile totals using the v
        current digit (index starts at -1 to enable to then start at
        array = @op.allopp(run_total, digits[(index+1)])
        array.each_with_index do |x, i|
            # create a new array to store
            newerString = returnString.dup
            # create a variable to store the operation that took pla
            symb = (@printArray[i].to_s)
            # push the symbol
            newerString.push(symb)
            # push the digit
            newerString.push((digits[(index+1)]).to_i)
            # call this method again, with running total now element
            than last time
            calculate(x, (index+1), digits, newerString)
        end
        else
            # Once final digit, send to check if running total = 10
            check(run_total, returnString)
    end
end
```

Calculator v2.5

Still iteration but single index starting from first digit

Five calculations done at once for each running total, creating new array of results.

Each operation and digit pushed to string, but first digit pushed separately at the end

Now starting from first digit not 0.

Check result separate method

```
def calculate(run_total, index, digits, return_string)
    print_array = ["+", "-", "/", "*", "**"]
    # Until final digit
    if index < 3
        # Call allopps to calculate all possibile totals
        current digit (index starts at -1 to enable to th
        array = allopp(run_total, digits[(index+1)])
        # if index = -1 array.each_with_index
        array.each_with_index do |x, i|
            # create a new array to store
            newer_string = return_string.dup
            # create a variable to store the operation th
            symb = (print_array[i].to_s)
            # push the symbol
            newer_string.push(symb)
            # push the digit
            newer_string.push((digits[(index+1)]).to_i)
            # call this method again, with running total
            higher than last time
            calculate(x, (index+1), digits, newer_string)
        end
        else
```

Calculator v2.5 cont.

Still iteration but single index starting from first digit

Five calculations done at once for each running total, creating new array of results.

Each operation and digit pushed to string, but first digit pushed separately at the end

Now starting from first digit not 0.

Check result separate method

```
            else
                # Once final digit, send to check if running total = 10
                check(run_total, return_string, digits)
            end
        end

        def check(result, new_string, digits)
            if result == 10

                new_string.unshift(digits[0].to_i)
                rows = []
                rows << [new_string.join(' '), " = #{result.to_i}"]
                table = Terminal::Table.new :rows => rows
                puts table

            else

            end
        end
end
```

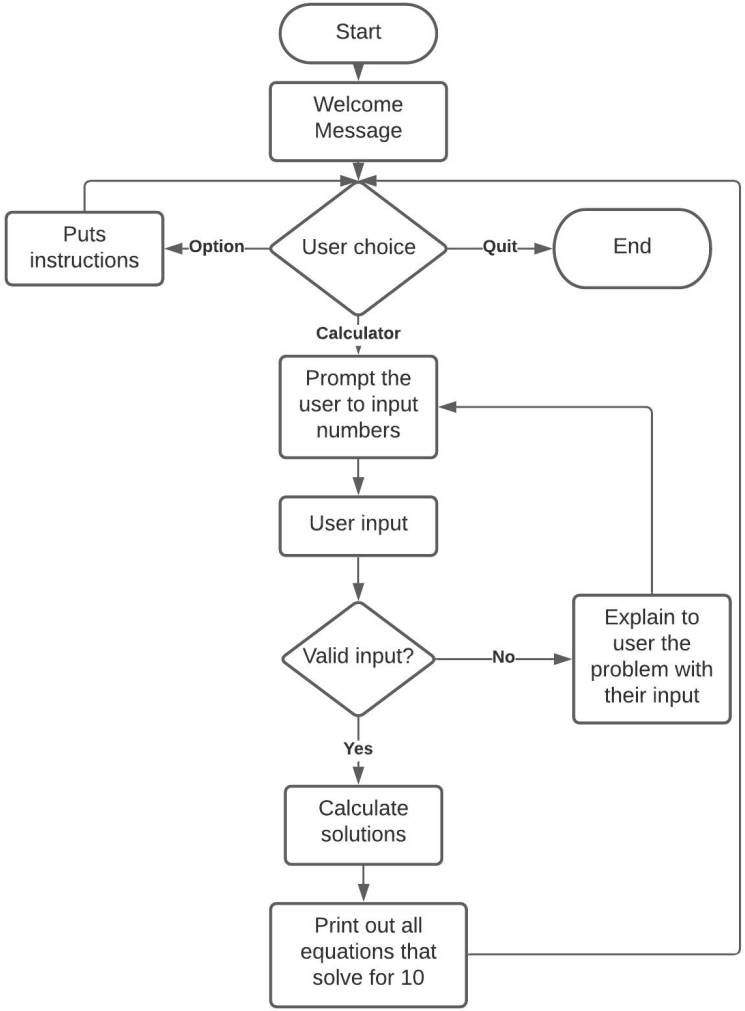
User flowchart

Welcome message

Choice: Instructions / Quit / Calculator

User inputs 4 numbers

Print solutions



Gems

TTY Prompt - Functionality - Avoids most errors by controlling user input options

Colorize - Aesthetics : Used for headings on every page

Terminal-table - Aesthetics : Used for displaying calculator results

Artii - Aesthetics: Main heading on Welcome

Future

Save results to JSON - Useful for future game that requires pre-knowledge of solutions to rank difficulty of different numbers

Testing & Error Handling

Implemented

Welcome message and instructions tested and documented

Basic calculation methods tested and documented e.g. add, subtract etc

Many errors handled by using TTY Prompt to limit user input.

Manual error handling to control for users inputting negative numbers. Tells them the issue and gives the chance to input again.

No issues with calculator dividing by zero as it just moves on to next number

2 Attempts, and still more required

Still need to solve for parentheses and two digit numbers by creating a method that can create new patterns out of the four digits

Automatically solve all patterns between 0000 and 9999 and output all solutions to JSON

For each pattern, sum the possible solutions. For patterns with solutions > 0 assign them as “Possible”

Generate random patterns that are “Possible”, and other game features

Keep log of how many solutions are undiscovered for each pattern

Lessons / Ethics

We should all strive to collaborate more

What's the difference between searching Stack Overflow and asking a fellow student? Stack Overflow doesn't get smarter by answering.

Help avoid more students dropping out. We want their ideas in 8 months even if not their abilities today.

Realistic expectations

It has been important for me to learn to manage my expectations. I shouldn't feel pride or shame after 1 month of ruby. Every new level will always make previous levels look simple. Live to fight another day. Next time choose something I understand better. Attrition on rubric.

Master the basics but also never fall behind on class work

Focusing on workbook meant a sacrifice in relevant class work. Still not understanding some basics. Assignment designed to make us feel this way and adapt. If they wanted our best we would have been given more than 7 days.

Thank you.

Someone choose a number that will pass!