



## Programming Exercise

This exercise is here to allow you to do what you do best - write great code. It's your opportunity to shine and show us what you can do. And yes, you get to tell us all about it in the follow-up review session.

There is one mandatory problem for this exercise. There's also a bonus problem which you can choose to complete at your leisure.

When ready, simply share your project with us on [github](#).

Happy coding!

The Sypht team

# Mandatory Problem

This one is **a mandatory problem** and it needs to be completed in JavaScript

We would love to see what you can do with the Syphth API. Sign-up for a free account (<https://syphth.com>) and use one or more of the provided fieldsets to create a demo which shows off your skills.

## Available fieldsets

- `syphth.document`
  - Returns basic summary information from various document types
  - E.g. document date, reference number, supplier ABN, etc..
- `syphth.invoice`
  - Extracts key fields for invoices, receipts and bills
  - E.g. subtotal, tax, total due, due dates, etc..
- `syphth.generic`
  - Extracts every label-value pair present on a document
  - Can address fields not covered by other fieldsets and recover higher-level page structures such as tables

Check <https://docs.syphth.com> for the full set. If there's something outside the free tier you'd like to use, just shoot us a message. Also see <https://github.com/syphth-team> for API client libraries and other demo applications which might help you get started.

## Requirements

- Build a simple web app frontend for our REST API which lets a user upload files and view the results
- Do something on top of data extracted by Syphth, e.g:
  - Analyse or plot bill size distribution over time from a corpus of receipts
  - Integrate with a third party API to verify the Supplier ABN on an invoice
  - Cluster similar fields by label from data returned by the `generic` fieldset
- Your application needs to be written in JavaScript

# Bonus Problems

For the Bonus problems there are no prescribed languages to choose from - simply read the problem description and pick a language one that works for you.

## Bonus Problem #1

### Date Calculator

You have joined a science project where a series of experiments are run for which you need to calculate the number of full days elapsed in between two events.

The first and the last day are considered partial days and never counted. Following this logic, the distance between two related events on 03/08/2018 and 04/08/2018 is 0, since there are no fully elapsed days contained in between those, and 01/01/2000 to 03/01/2000 should return 1.

The solution needs to cater for all valid dates between 01/01/1901 and 31/12/2999.

#### Test cases

- 1) 02/06/1983 - 22/06/1983 = 19 days
- 2) 04/07/1984 - 25/12/1984 = 173 days
- 3) 03/01/1989 - 03/08/1983 = 1979 days

(Please note these dates are formatted DD/MM/YYYY)

#### Instructions

- Write a command-line based program that accepts date input from the console.
- You **should not** use any existing date libraries for your implementation.
- You may however use date libraries to **test your solution** (we encourage it!)
- Consider other potential input sources & how your app might fit into a bigger system.

## Bonus Problem #2

### Progress Pie

Progress bars are common in software; some accurately give you an idea of how an application is progressing, other hang for what seems like an eternity, sometimes the application just shows a spinner or an hourglass to say 'something' is happening...

Sometimes the progress bar isn't a bar or an hourglass, it is a circle.

Imagine you are using an app that has a circular progress bar, or if you prefer, a progress pie. On the screen is a square area with its bottom corner at (0,0) and its upper-right corner at (100,100). Every pixel in this square is either white or black. Initially, the progress is at 0% and all pixels are white. When the progress percentage,  $P$ , is greater than 0% a sector of angle  $(P\% * 360)$  degrees is coloured black, starting from the line segment from the centre of the square (50, 50) to the centre of the top side (50, 100) and proceeding clockwise.



As you wait for the progress pie to fill in, you find yourself wondering whether certain points would be white or black at different amounts of progress.

#### Input

The input file begins with an integer  $T$ , the number of points you're curious about. For each point, there is a line containing three space-separated integers,  $P$ , the amount of progress as a percentage, and  $X$  and  $Y$ , the coordinates of the point.

#### Output

For the  $i$ th point, print a line containing "Case #i: " followed by the color of the point, either "black" or "white".

## Constraints

- $1 \leq T \leq 1,000$
- $0 \leq P, X, Y \leq 100$
- Whenever a point (X, Y) is queried, it's guaranteed that all points within a distance of  $10^{-6}$  of (X, Y) are the same color as (X, Y).

## Sample

Input	Output
5 0 55 55 12 55 55 13 55 55 99 99 99 87 20 40	Case #1: white Case #2: white Case #3: black Case #4: white Case #5: black

In the first case all of the points are white, so the point at (55, 55) is of course white.

In the second case, (55, 55) is close to the filled-in sector of the circle, but it's still White.

In the third case, the filled-in sector of the circle now covers (55, 55), coloring it black.

## Instructions

- Write a program that solves the problem above for any number of valid coordinates.
- We encourage you to test your solution and make those tests reproducible by us.
- Consider other potential input sources & how your app might fit into a bigger system.