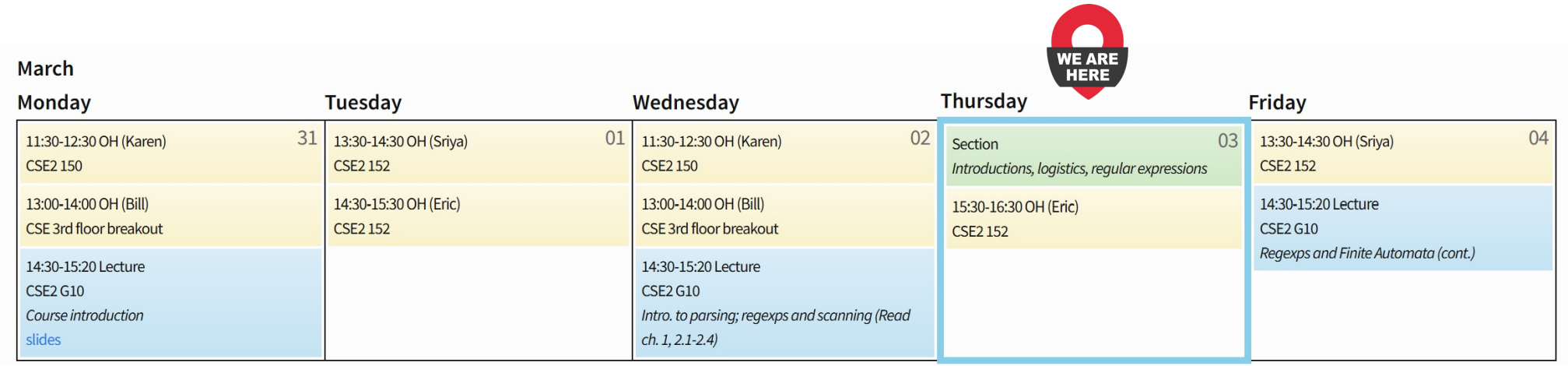
# Section 1: Intro & RegEx

Eric, Karen, Sriya, Bill

Adapted from 24au

# Announcements

- Due early next week (Tuesday 4/8, 11:59pm): Project partner info
  - Check Ed discussion board if you need a partner!



# TA Intro

- TA #1
  - Year
  - Industry/research/etc. experience(s)
  - Class(es) you've TA'd.
- TA #2
  - Year
  - Industry/research/etc. experience(s)
  - Class(es) you've TA'd.

# TAs' Tips

- Get started on project parts early!
  - *Especially* on the parts of the project which we explicitly advise to do so.
- Work *with* your partner on the project so you can both be up-to-speed on everything instead of dividing the tasks up amongst yourselves.
  - The project is entirely cumulative, so making sure that you both know all content helps with debugging the project as it progresses.
  - Project content can (*and likely will*) show up on exams as well, so make sure you know all of it.
- Instead of thrashing when you can't get a solution, come to OH!
- If you are having non-academic issues which are preventing you from meeting any deadlines (ex:  illness), please contact the course staff ASAP so we can try to arrange an accommodation for you.

# Icebreakers

- Find a partner, discuss what you did over Spring Break.
- Once time's up, state the following:
  - Your partner's name
  - Your partner's year
  - What your partner is studying (major/ minor/ field(s) of interest/ research project)
  - What your partner did over Spring Break.

# RegEx Worksheets!

# Answers

# Problem 1i

1) Describe in English the set of strings generated by each of the following regular expressions and give two different strings it can produce:

i) (1 | 0)* 0

# Problem 1i

1) Describe in English the set of strings generated by each of the following regular expressions and give two different strings it can produce:

i) (1 | 0)* 0

Non-empty binary strings ending with 0

0, 10, 111100000, 01010

# Problem 1ii

1) Describe in English the set of strings generated by each of the following regular expressions and give two different strings it can produce:

ii) ([A-Z][a-z]* | [0-9]+)

# Problem 1ii

1) Describe in English the set of strings generated by each of the following regular expressions and give two different strings it can produce:

ii) ([A-Z][a-z]* | [0-9]+)

Non-empty sequence of lower case letters with first letter upper cased or sequence of base 10 digits

A, Aa, Abczzz, 0, 3, 42, 17

# Problem 1iii

1) Describe in English the set of strings generated by each of the following regular expressions and give two different strings it can produce:

iii) ($\varepsilon$ | 4?0+1* X 3+)

# Problem 1iii

1) Describe in English the set of strings generated by each of the following regular expressions and give two different strings it can produce:

iii) ($\varepsilon$ | 4?0+1* X 3+)

This one does not have a simple description, but some strings generated include ε, 0X3, 401X333, 40000111X333

However, an approximate description would be: an empty string, or strings starting with an optional 4 followed by one or more 0's then zero or more 1's, followed by the letter X and one or more 3's.

# Problem 2i

2) Write a regular expression for each of the following specifications:

i) All strings consisting of 0’s and 1’s (binary digits) with an even number of 0s

# Problem 2i

2) Write a regular expression for each of the following specifications:

i) All strings consisting of 0's and 1's (binary digits) with an even number of 0s
1* (0 1* 0 1*)*

(There are other regular expressions that generate the same set of strings)

# Problem 2ii

2) Write a regular expression for each of the following specifications:

ii) camelCased variable name in Java, where the alphabet is upper and lower-cased letters without any numbers or underscores

# Problem 2ii

2) Write a regular expression for each of the following specifications:

ii) camelCased variable name in Java, where the alphabet is upper and lower-cased letters without any numbers or underscores

[a-z]+([A-Z][a-z]*)*

Note: this solution allows multiple upper-case letters to appear adjacent to each other.
Challenge: produce a solution that does not allow adjacent upper-case letters.

# Problem 2iii

2) Write a regular expression for each of the following specifications:

iii) Non-empty binary strings where each 1 directly follows a 0 (challenge: only use symbols in table up until *)

# Problem 2iii

2) Write a regular expression for each of the following specifications:

iii) Non-empty binary strings where each 1 directly follows a 0 (challenge: only use symbols in table up until *)

Challenge 1: (0 | 01) (0 | 01)*

Normal: (0+1?)+