# CS 35L: Week 3 Solutions

## Introduction to Regular Expressions

**Regular Expressions**

For the following exercises, quickly verify your answers using grep or sed as you go. In addition, assume the environment locale variable is set to LC_ALL=C (i.e. letters are ASCII a-zA-Z only).

1. Write a **basic** regular expression that causes grep to match lines ending in the characters ‘dog’.

   It should match:
   - ‘bulldog’
   - ‘smol dog’
   - ‘hotdog’

   It should not match:
   - ‘doge’
   - ‘dogecoin’
   - ‘doggo’
   - ‘woof dogs’
   - ‘sinnoh remakes’

   **dog$**

2. Write a basic regular expression that checks whether or not a vowel is present.
   **[aeiou]\|y**
   a. Write an extended regular expression that checks if a vowel is present.
   **[aeiou]|y**
   b. What is the difference between grep and grep -E? egrep?
   **Extended regex operators need to be escaped in basic regex**

   It should match:
   - ‘abcdef’
   - ‘ayylmao’
   - ‘foobar’

   It should not match:
   - ‘bbc’
   - ‘nnn’
   - ‘ttyl’

   ~~It should sometimes match:~~
   - ~~why?~~
   - ~~Y~~
   - ~~y~~

3. Write a **basic** regular expression that causes grep to match lines that contain the *subsequence* ‘abc’. A subsequence of a string is a new string which is formed from the original string by deleting some (can be none) of the characters without disturbing the relative positions of the remaining characters. Example: ‘bce’ is a subsequence of ‘abcde’ while ‘bec’ is not.

   It should match:
   - ‘**a**Q**b**Q**c**Q’
   - ‘**ab**a**c**us’
   - ‘QQ**a**aQ**b**QQQ**c**QQQ’
   - ‘**abc**’

   It should not match:
   - ‘bac’
   - ‘QQQacbQQQabbbb’
   - ‘blockade’

   **a.*b.*c**

4. Write a sed-based command that replaces every line that contains the subsequence ‘abc’ to ‘BOOM’. As an example, the command should make the following transformation:

| | | |
|---|---|---|
| **abc** | → | BOOM |
| bac | → | bac |
| blockade | → | blockade |
| Q**a**Q**b**Q**c**Q | → | BOOM |
| **ab**a**c**us | → | BOOM |
| QQ**a**aQ**b**QQQ**c**QQQ | → | BOOM |
| QQQacbQQQabbbb | → | QQQacbQQQabbbb |
| BOOM | → | BOOM |

   **sed 's/.*a.*b.*c.*/BOOM/g'**
   **Notice how we need to prepend ‘.*’ and append ‘.*’ to the regex from question 2 in order to replace the entire line.**

5. Write a **basic** regular expression that causes grep to match lines solely consisting of alphabetical characters (A-Za-z) and digit characters (0-9), where no alphabetical characters appear before digit characters.

   It should match:
   - '0123456789abc'
   - '42WallabyWay'
   - ''
   - 'Meow'
   - '1234'

   It should not match:
   - 'abc123'
   - '1$7 apples'
   - '1a2b3c'

   **^[0-9]*[a-zA-Z]*$**
   **or**
   **^[[:digit:]]*[[:alpha:]]*$**

6. Write an **extended** regular expression that causes grep -E to match lines that either contain at least one digit or the string "flower". It should be as short as possible. The optimal solution is 12 characters long. (If you like this sort of thing, check out code golfing!)

   **[0-9]|flower**

7. What do the following **extended** regular expression operators do? (These are operators, not full regular expressions themselves.)
   a. '+' **matches the preceding item one or more times**
   b. '?' **makes the preceding item optional**
   c. '|' **preceding or following item must be present**
   d. '(' and ')' **groups enclosed elements into a subexpression**
   e. '{i}' **matches the preceding element exactly i times**
   f. '{i,j}' **matches the preceding element between i and j times**