CSC 404 - Homework 2 - Name: Chris Glanzer

Problem 1 (Odd Digits - Wee!)**.** Consider the language of all numbers that contain an odd number of digits.

a. Construct a finite-state automaton that recognizes this language.

b. Represent this language with a regular expression.

```
/(-?\d(\d\d)*)/g
```

-?

c. Implement the regular expression and test it against the sets

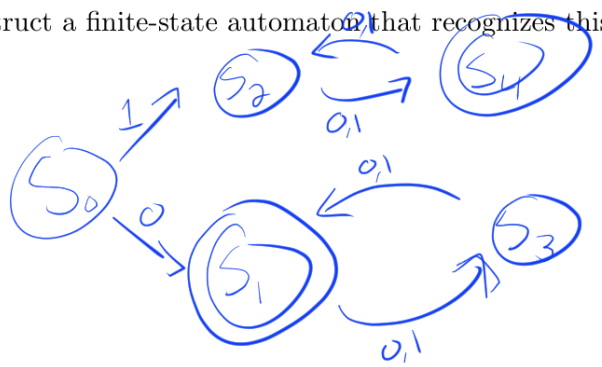
```
/^(-?\d(\d\d)*)$/gm
Text  Tests NEW
8675309
12345
111

1337
1234
12345678
```

Accepted: `8675309, 12345, 111` Rejected: `1337, 1234, 12345678`

Problem 2 (0 = Odd and 1 = Even?)**.** Consider the language of all bit strings that start with 0 and have odd length or start with 1 and have even length.

a. Construct a finite-state automaton that recognizes this language.



b. Represent this language with a regular expression.

```
/^((0(\d\d)*|1\d(\d\d)*))$/gm
Text  Tests NEW
101010
11011
1111
000

10101
011110
11011
```

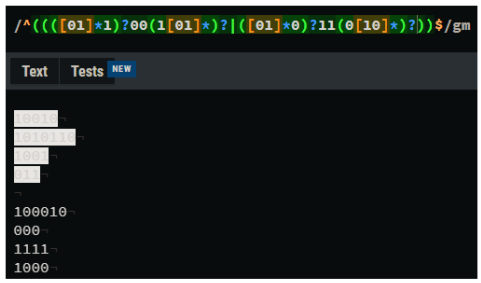
c. Implement the regular expression and test it against the sets

Accepted: `101010,01010,1111,000` Rejected: `10101,011110,11011`

Problem 3 (Yay double letters, but only once)**.** Consider the language of all bit strings that contains exactly one double letter. For example 10010 has exactly one double letter, but 100010 has two double letters.

a. Represent this language with a regular expression.
Hint: You should see this fits into two bins - one that has the double `00` and one that has the double `11`. Do not forget about `0?` and `1?` as convenient ways to insert/append possible `0`s or `1`s to strings.

b. Implement the regular expression and test it against the sets



Accepted: `10010, 1010110, 1001, 011` Rejected: `100010, 000, 1111, 1000`

Problem 4 (Passwords!)**.** Define a regular expression that matches passwords with the following rules (of your choice):

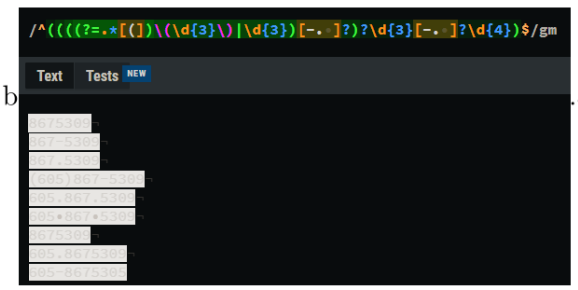
i. 8-16 characters
ii. 1+ capital letters
iii. 1+ special characters
iv. 1+ numbers
v. (More?)

Then, implement the regular expression and test it against passwords of your choice. (Be creative/try something new!)

Problem 5 (Phone Numbers)**.** Give a regular expression to describe American phone numbers in all the various forms you can think of. Note - phone numbers is a rabbit hole that you can go down pretty far (e.g., various ways to write them/international codes,...). At a minimum, your regular expression should accept the following phone numbers (Jenny in South Dakota):

`8675309`
`867-5309`
`867.5309`
`(605)867-5309`
`605.867.5309`

Tip: If it makes it easier, we will b .867-5309. Do not forget that you may need to escape the (and).



Problem 6 (Addresses!)**.** Give a regular expression to describe addresses. Then, implement the regular expression and test it against address of your choice. Note - the world of address is a very deep rabbit hole that you can venture down (i.e., rules and requirements), so go as deep as you want. At a minimum, your regular expression should accept the following address:

`820 N Washington Ave, Madison, SD 57042`
`1600 Pennsylvania Avenue NW, Washington, DC 20500`
`4059 Mt Lee Dr, Hollywood, CA 90068`
`350 Fifth Avenue, New York, NY 10118`

Tip: To account for the white space you can put spaces in the Regular Expression or make use of $\backslash s$.