3 (100 PTS.) Regular.

For each of the following languages, give a regular expression that accepts that language, and briefly argue why your expression is correct. Below, $\#_0(x)$ denotes the number of 0s in x.

You do not need to provide the shortest [or even short] regular expression that works – instead, try to provide a systematic solution explaining how you reached your answer. Please provide an explicit and full regular expression.

3.A. (25 PTS.) All strings in $\{0, 1\}^*$ that do not contain 01010 as a subsequence.

The longest sequence of alternating letters that does not contain the subsequence 01010 is "10101". As such, the set of all strings that do not contain 01010 as a subsequence is:

1*0*1*0*1*

3.B. (25 PTS.) All strings in $\{0, 1\}^*$ such that the symbols at even positions are alternating. For example: the string $0\underline{0}0\underline{1}1\underline{0}0\underline{1}1\underline{0}1$ is in the language because the underlined characters alternate between 0 and 1. While $1\underline{0}1\underline{1}1\underline{0}0\underline{0}1\underline{0}1$ is not in the language.

(Hint: Start with a regular expression for all strings that all their bits are alternating, and then extend it to the desired expression.)

All strings with alternating bits:
(e+1)(01)*(e+0)

All strings with alternating bits at even positions:
e + (11 + 01) + (0+1) (e + 1(0+1)) (0(0+1)1(0+1))* (e + 0 + 0(0+1))

Brute forced the cases for "", 0, 1, 00, 01, 10, and 11.

(0+1) (e + 1(0+1)) covers the first digit, as well as even digits that start with 1.

(0(0+1)1(0+1))* covers all even digits that start with 0 and end with 1.

(e + 0 + 0(0+1)) covers the last even digits that end with 0, as well as the last odd digit, if it exists.

3.C. (25 PTS.) All strings $x \in \{0, 1\}^*$, such that x does not begin with 010 and $\#_0(x)$ is even.

(Hint: First come up with a regular expression for all strings with even (and separately odd) number of 0s. Then create a regular expression for all the strings in the language starting with 1, etc.)

```
Strings with even number of 0's:
1*(01*0)*1*

Strings that don't begin with 010:
1(1*(01*0)*1*)
00(1*(01*0)*1*)
011(1*0(01*0)*1*)
011(1*(01*0)0*1*)

Strings with length at most 3 that don't equal 010:
0, 1, 00, 01, 10, 11, 000, 001, 011, 100, 101, 110, 111
```

All strings such that x does not begin with 010 and $\#_0$(x) is even:

```
(0+1+00+01+10+11+000+001+011+100+101+110+111) + 1(1*(01*0)*1*) +
00(1*(01*0)*1*) + 011(1*0(01*0)*1*) + 011(1*(01*0)0*1*)
```

3.D. (25 PTS.) All strings in $\{0, 1\}^*$ that do not contain 010 as a substring.

(Hint: Generate a regular expression for all strings in this language that starts with a 0 and ends with a 0. Once you have this regular expression, getting the answer is shockingly easy.)

```
All strings that start with 0 and end with 0:
0*1*0*

All strings that do not contain 010 as a substring:
1*(0*111*0*)*1*
```

4 (100 PTS.) Divisible by something.

In the following, you need to explain (shortly) why your solution works (a formal proof is not necessary).

4.A. (50 PTS.) Let $\Sigma = \{0, 1\}$. For a string $w \in \Sigma^*$, let w_2 be the integer value if we interpret w as a number written in base 2. Thus, $1010_2 = 1 \cdot 2^3 + 1 \cdot 2^1 = 10$.

Describe *formally* a DFA that accepts the language L of all strings $w \in \Sigma^*$, such that $(w^R)_2$ is divisible by 13. For example, $001011 \in L$, since $(001011^R)_2 = 110100_2 = 13 \cdot 4$, which is divisible by 13, as is $10111011 \in L$. But $001 \notin L$, since $100_2 = 4$, which is not (yet) divisible by 13.

(Hint: Think about the DFA as giving you the input from right to left.)

States: M0, M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, M11, M12
Alphabet: {0, 1}
Transition Function:

- Let MX be a state, with X being the corresponding state's number from 0 to 12.
- If given a 0, MX points to M((X * 2) % 13).
- If given a 1, MX points to M((X * 2 + 1) % 13).

Initial State: M0
Accepting State: M0

Since we are looking for numbers divisible by 13, the DFA must remember all possible integer remainders for when a number is divided by 13.
The alphabet is {0, 1}, since we are working in modulo 2.
Since we are counting this in modulo 2, increasing the digit by 1 would multiply the pre-existing digits by 2. Thus, any given digit will multiply the total by 2, and if the digit is 1, it will increase the current substring's sum by an additional 1 unit.
While the actual transition function is too complicated to outright draw, we know that 0 is divisible by 13. Out of the 13 possible states, it is the only state MX where X % 13 = 0. Thus, it is the only accepting state.
It is also the initial state, as the empty string and all initial 0's have a sum of 0.

4.B. (50 PTS.) A string $w \in \Sigma^*$ is a k-palindrome, for a prespecified integer $k > 1$, if k divides w_2, and k also divides $(w^R)_2$. Describe *formally* a DFA that accepts all strings $w \in \Sigma^*$ that are 13-palindrome.

DFA for w_2:

```
Let P = [1, 2, 4, 8, 3, 6, 12, 11, 9, 5, 10, 7], where the number
at index i is equal to 2^i % 13.

Let R = 0, I = 0.
For every incoming digit X:
      If X = 0, R = R.
      If X = 1, R = (R + P[I]) % 13
      I = (I + 1) % 13
```

States: 13 * 13 * 13 = 2197, with M_0_0_0 being the starting state, which represents an empty string.

Alphabet: {0, 1}

Transition Function:

- See notes.

Initial State: M_0_0_0

Accepting State: M_0_0_0, M_0_0_1, M_0_0_2, M_0_0_3, M_0_0_4, M_0_0_5, M_0_0_6, M_0_0_7, M_0_0_8, M_0_0_9, M_0_0_10, M_0_0_11, M_0_0_12

Explanation:

To make a DFA that accepts binary strings divisible by 13, we need to keep track of both the remainder, which can be 1 of 13 values, as well as the current index, which can be 1 of 13 values. We cross multiply these variables to create the "DFA for w_2". This requires 13 * 13 = 169 states.

Additionally, the binary string must also be divisible by 13 when reversed. We have created a DFA for that in part A which requires 13 states. We cross multiply our w_2 DFA's states with the $(w^R)_2$ DFA from the previous question, giving us a total of 169 * 13 = 2197 possible states.

For convenience, our states can be called M_X_R_I, where X keeps track of the remainder of our $(w^R)_2$ DFA, R is the remainder of our w_2 DFA, and I is the current index of the number, modulo 13.

We will assume that the empty string is equal to 0, and set M_0_0_0 as the initial state. Since we are looking for 13-palindrome numbers, we will only accept states where X and R are both equal to 0, signifying that the string, when both read from the left and from the right, is divisible by 13.