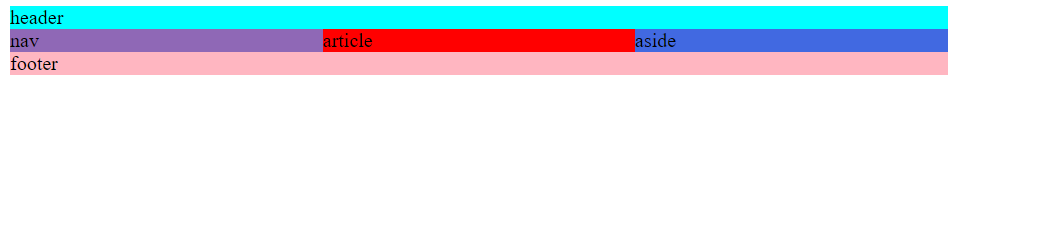
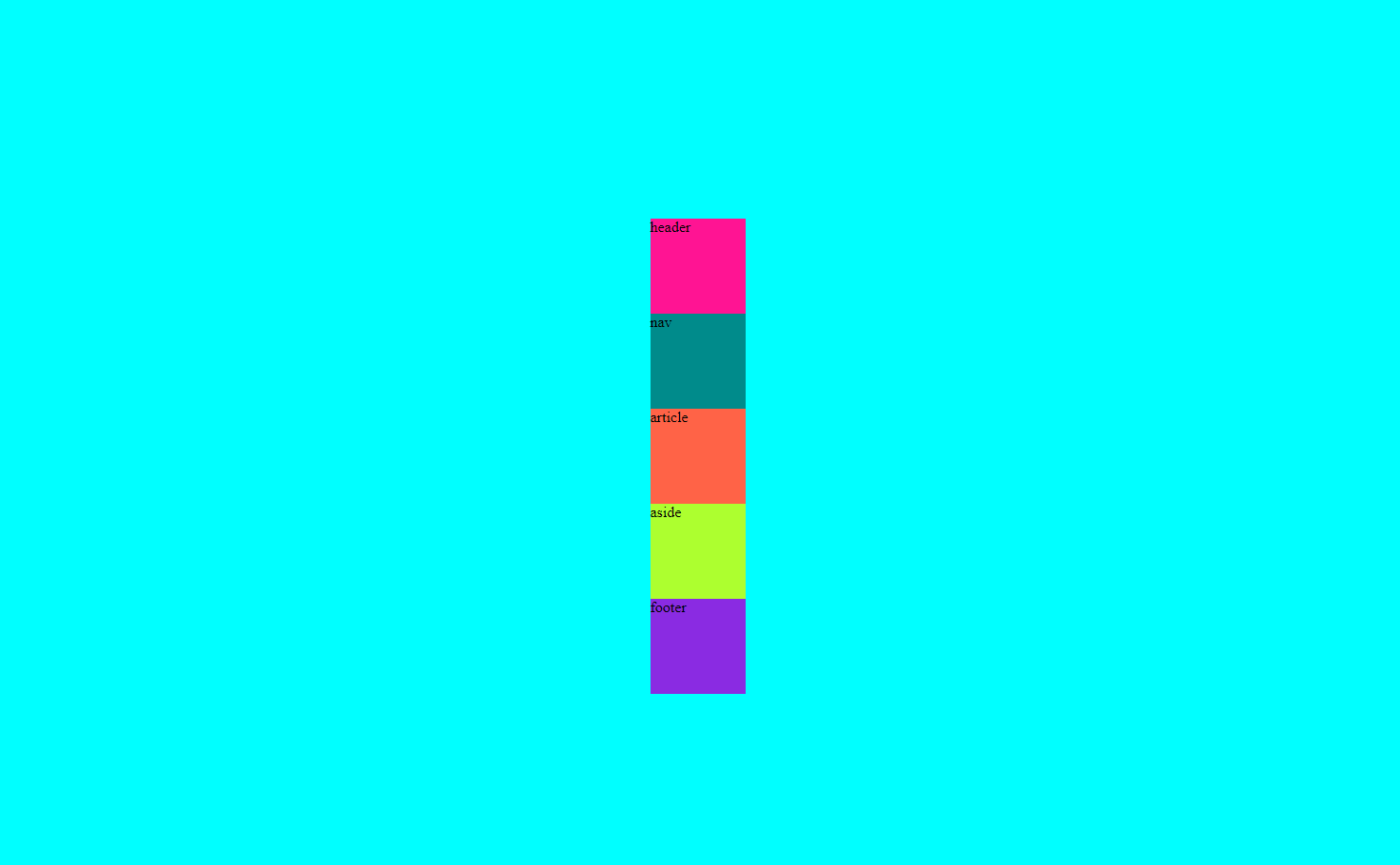
**Section 1 HTML:**

1. Flex-direction allows you to do what? html\_1.txt
2. How to create container using flex? give example. htmlQ2.html style2.css
3. Define a grid container with 3 columns and 3 rows with the same width and height htmlQ3.html style3.css  
   
4. Center a button in the screen htmlQ4.html style4.css
5. Create a html table with user information such as (id, first name, last name, email)
6. htmlQ5.html style5.css
7. Add design CSS to the table from point (5) htmlQ5.html style5.css
8. Add 3 images in the same line to the middle of the screen htmlQ6.html style6.css
9. Add 3 images in the same column in the middle of screen htmlQ6.html style6.css

Center items in column example:



Center items in row example:

Shape

Description automatically generated

**JavaScript:**

**Q0**

Define class of Person.js with first name and last name and age

Define class of Student.js with properties of first name and last name and grade and age

Add method of print name

Add method of print age

Inherit details from Person class

*Person and Student should be in separate JavaScript files.  
 (*Hint*)*

**Text

Description automatically generated**

**Q1**

A phrase is a **palindrome** if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.

Given a string s, return true*if it is a****palindrome****, or*false*otherwise*.

**Example 1:**

**Input:** s = "A man, a plan, a canal: Panama"

**Output:** true

**Explanation:** "amanaplanacanalpanama" is a palindrome.

**Example 2:**

**Input:** s = "race a car"

**Output:** false

**Explanation:** "raceacar" is not a palindrome.

**Example 3:**

**Input:** s = " "

**Output:** true

**Explanation:** s is an empty string "" after removing non-alphanumeric characters.

Since an empty string reads the same forward and backward, it is a palindrome.

**Q2**

Given a **non-empty** array of integers nums, every element appears twice except for one. Find that single one.

You must implement a solution with a linear runtime complexity and use only constant extra space.

**Example 1:**

**Input:** nums = [2,2,1]

**Output:** 1

**Example 2:**

**Input:** nums = [4,1,2,1,2]

**Output:** 4

**Example 3:**

**Input:** nums = [1]

**Output:** 1

**Q3**  
  
Implement the myAtoi(string s) function, which converts a string to a 32-bit signed integer (similar to C/C++'s atoi function).

The algorithm for myAtoi(string s) is as follows:

1. Read in and ignore any leading whitespace.
2. Check if the next character (if not already at the end of the string) is '-' or '+'. Read this character in if it is either. This determines if the final result is negative or positive respectively. Assume the result is positive if neither is present.
3. Read in next the characters until the next non-digit character or the end of the input is reached. The rest of the string is ignored.
4. Convert these digits into an integer (i.e. "123" -> 123, "0032" -> 32). If no digits were read, then the integer is 0. Change the sign as necessary (from step 2).
5. If the integer is out of the 32-bit signed integer range [-231, 231 - 1], then clamp the integer so that it remains in the range. Specifically, integers less than -231 should be clamped to -231, and integers greater than 231 - 1 should be clamped to 231 - 1.
6. Return the integer as the final result.

**Note:**

* Only the space character ' ' is considered a whitespace character.
* **Do not ignore** any characters other than the leading whitespace or the rest of the string after the digits.

**Example 1:**

**Input:** s = "42"

**Output:** 42

**Explanation:** The underlined characters are what is read in, the caret is the current reader position.

Step 1: "42" (no characters read because there is no leading whitespace)

^

Step 2: "42" (no characters read because there is neither a '-' nor '+')

^

Step 3: "42" ("42" is read in)

^

The parsed integer is 42.

Since 42 is in the range [-231, 231 - 1], the final result is 42.

**Example 2:**

**Input:** s = " -42"

**Output:** -42

**Explanation:**

Step 1: " -42" (leading whitespace is read and ignored)

^

Step 2: " -42" ('-' is read, so the result should be negative)

^

Step 3: " -42" ("42" is read in)

^

The parsed integer is -42.

Since -42 is in the range [-231, 231 - 1], the final result is -42.