Question A - Fizz Buzz Cipher

It's Denis' 21st birthday today and he has received a mysterious birthday card with an encrypted message.

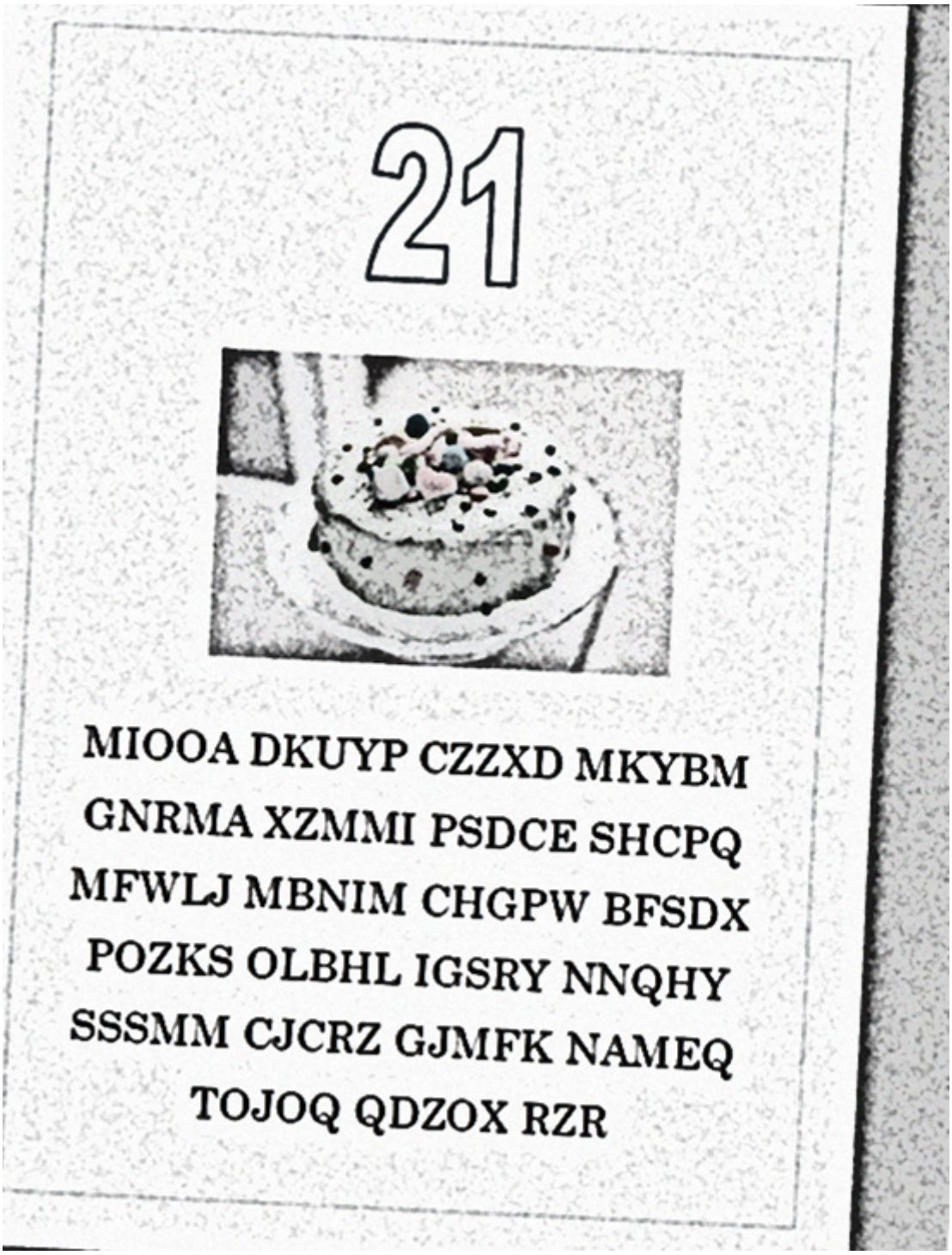
Denis loves cryptography, and quickly identified that the message had been encrypted using the 'Fizz Buzz Cipher' - a creation of his own! Soon after making this discovery, Denis jumped to his feet and left to make a phone call.

A scan of the card has been included below. What does the encrypted text say?

The Fizz Buzz Cipher

To encrypt a message using the fizz buzz cipher, the following steps are taken:

1. A positive integer is chosen to be used as a 'key'.
2. The original message plaintext is written out in capital letters, omitting any numerals, spaces & punctuation.
3. A 'fizz buzz keystream' is written out to match the length (in characters) of the plaintext. To write a fizz buzz keystream, write a series of consecutive integers as would appear on a number line starting with the chosen 'key' integer, with the following exceptions:
    - Replace any integers divisible by 3 (and not 5) with "FIZZ"
    - Replace any integers divisible by 5 (and not 3) with "BUZZ"
    - Replace any integers divisible by 15 with "FIZZBUZZ"
4. Each character of the plaintext is incremented by the 'value' of the corresponding character of the keystream, to produce the ciphertext (encrypted message).
    - Digits have the same value they represent, i.e. 1 = 1
    - Letters have the value matching their 0-indexed position in the alphabet, i.e. A = 0, B = 1 ... Z = 25



Having decrypted the message, paste the plaintext into the hex-grid tool with **Question A** selected & present your pattern for verification.

For example, `HELLOWORLD` would be an incorrect answer, given in the correct format.

Question B: Funny Cat Slideshow

Ellie has a long bus journey in the countryside. To pass the time, she tries watching a funny cat compilation on her phone. Having only a weak connection during the journey, the video playback is frequently interrupted.

Ellie's favourite content platform 'DougaBeam' uses the following logic for streaming video:

- Each video frame has an associated frame ID, indicating its position in the overall video
- Any video stream is treated as a series of 'buffer blocks' containing 12 consecutive frames each, though the final 'block' within a video may contain fewer frames.
- A buffer block is considered 'unhealthy' until all anticipated frames have arrived, at which point the block becomes 'healthy'.
- Video frames are played at a rate of 20 frames per second.
- If the video is playing and would enter an 'unhealthy' block of video frames, playback instead stops after playing the last frame from the previous healthy block.
- When the video isn't playing, the application waits until either of the following conditions are met before resuming (or starting) playback:
    - A) The next three un-played blocks of the video are healthy
    - B) All remaining blocks of the video are healthy
- DougaBeam evaluates whether to begin or resume playing at 200ms intervals using an internal timer system. Importantly, this check ignores any video frames that arrive at the same time as the check is being performed.

After a series of interruptions, Ellie's video finally finishes playing after over 16 minutes. Given `FunnyCatStream.csv`, which details the arrival of video frames relative to DougaBeam's internal timer, how many times did playback stop to wait for more video frames to arrive?

Enter your answer as an integer (e.g. 10) into the hex-grid tool under **Question B** and generate a pattern to present for validation.

Question C: Transposed Pixels

Katie is very suspicious of cloud storage and gently obfuscates all of her images with custom software.

The software is simple and shifts the pixel values for each colour channel across the canvas to scramble images. To support this, the image canvas is treated as being joined at both sets of opposite edges or 'wrapped'. If a pixel value would be shifted outside of the original canvas, it instead continues moving from the opposite side.

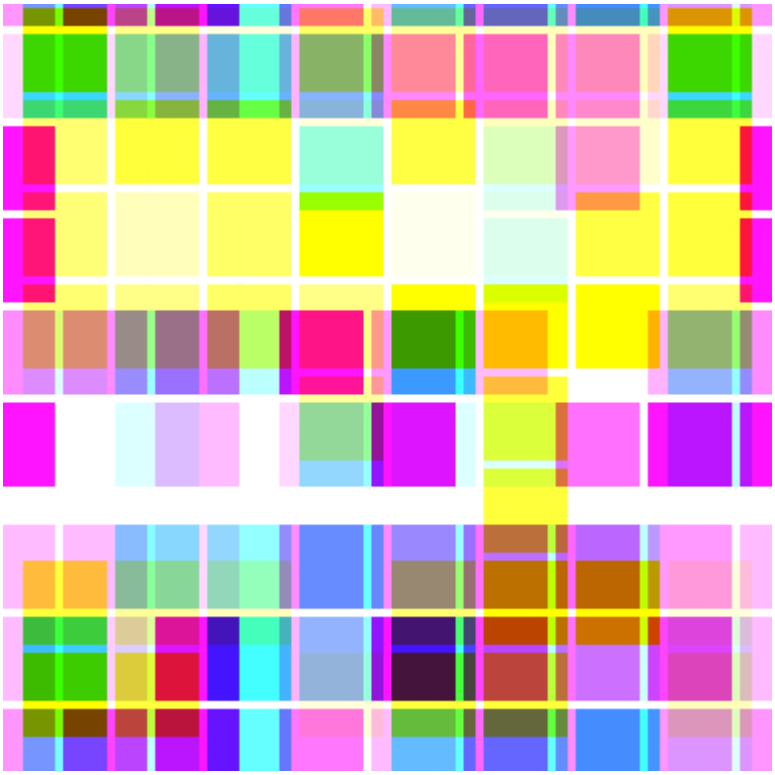
There are lots of different modes available, but Katie likes to use one called "The London" and applies it for **n** many iterations.

'The London' performs the following manipulation:

- All red channel values are moved up 2 places.
- All green channel values are moved up 2 places, then to the left 1 place.
- All blue channel values are moved up 3 places, then to the right 3 places.

`image.png` has been obfuscated by Katie, but **n** remains unknown.

Recover the original image and present it for validation.

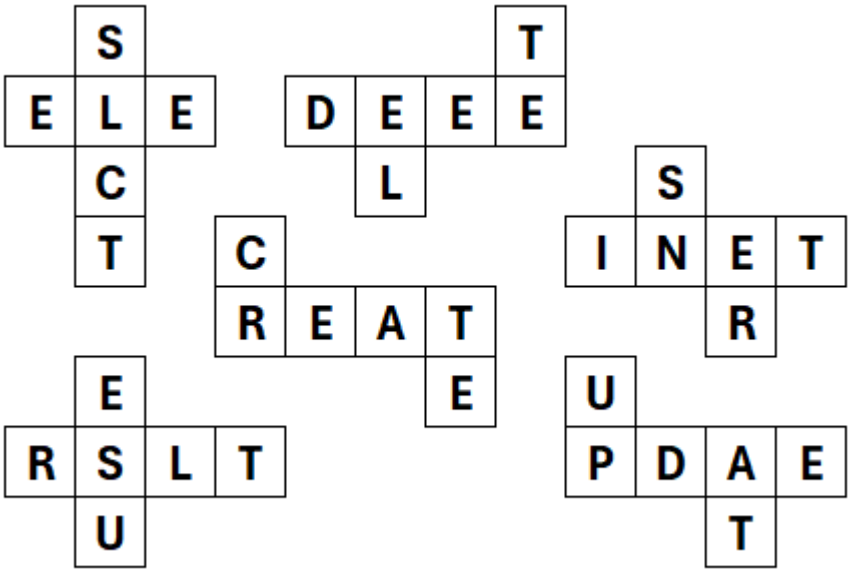


Question D: Dice Game I

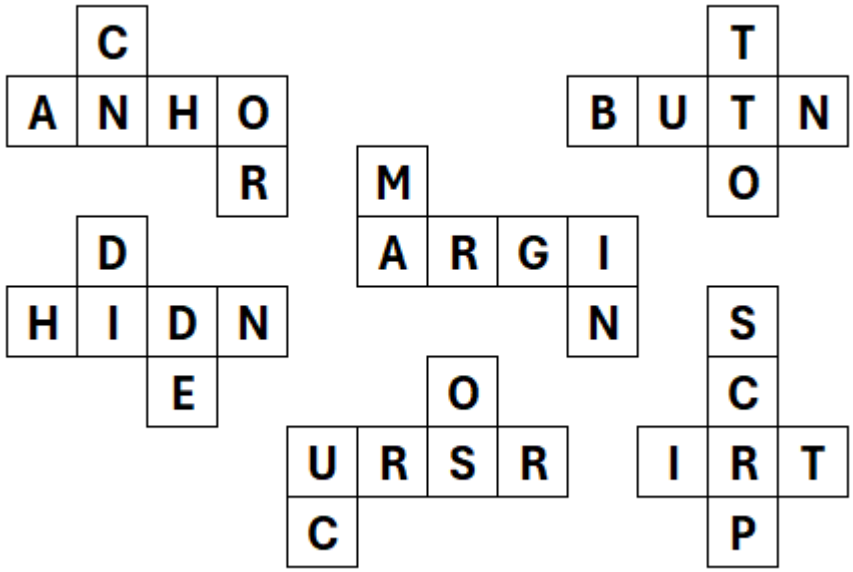
One afternoon, James took it upon himself to invent a word game. His game sees two players taking turns to roll their respective sets of six dice and score words using the twelve letter faces available on the table.

As hours passed, James found himself utterly enamoured with his invention, yet at a loss as to how to assign letters to each of the dice faces. Ultimately, he resigned to simply assigning a 6-letter word to each of the dice. Surely, dice composed of English words themselves would allow a diverse range of words to be spelled. James' initial set of dice maps are pictured below.

> Words used: SELECT CREATE DELETE INSERT UPDATE RESULT



> Words used: ANCHOR BUTTON CURSOR HIDDEN MARGIN SCRIPT



After a few turns of rolling the dice, James noted that many of his favourite words were entirely impossible to construct within the game. He found himself particularly dismayed at his inability to spell "JOIN" with the dice.

Given James' custom dictionary of words: `CustomWordDictionary.txt`, find the number of words which cannot be spelt by using each of the 12 dice at most once (to contribute a single letter).

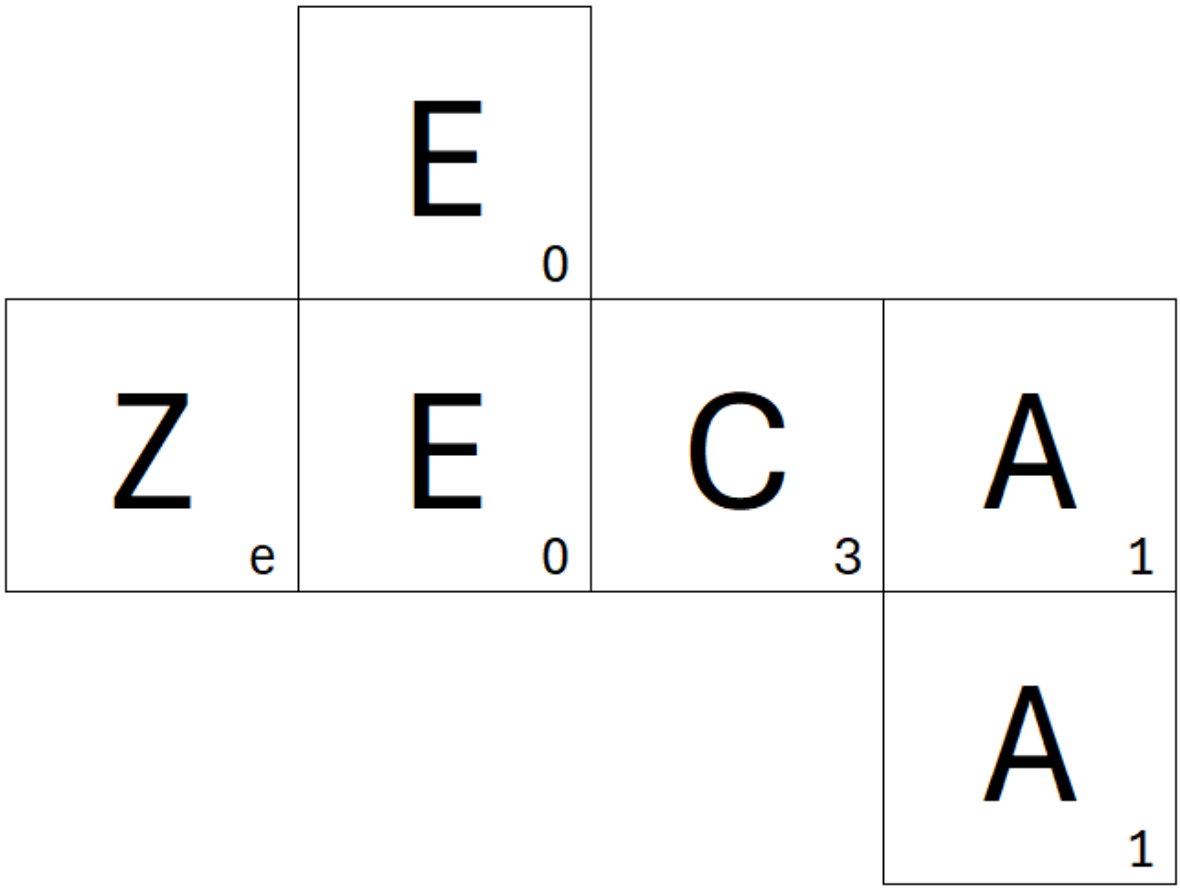
Enter your answer as an integer (e.g. 56) into the hex-grid tool under **Question D** & present your pattern for verification.

Question E: Dice Game II

Ashamed of the poor quality of his initial set of dice maps, James produced a new set of dice, using a complex algorithm to find a selection of mappings that yielded a much more comprehensive coverage of the English language.

For his revised set of dice James introduced a custom scoring scheme, commissioning special dice sets with single-digit letter scores on each face - in hexadecimal of course! Throughout each game, players score points for the words they find according to the sum of the letter scores used. The player with the most points at the end of the game wins.

Below is a glimpse of one of James' revised dice maps. Notably, E is seen to be worth 0 points.



Given this knowledge, propose a scoring scheme consistent with the dice map seen above, and both scoresheets from James & Emily's most recent game: `Scoresheet_Emily.pdf` & `Scoresheet_James.pdf`.

Enter your answer as length-26 string - the scores (in hexadecimal) for each letter in alphabetical order. Paste this answer into the hex-grid tool under **Question E** & present your pattern for verification. As an example, `a0a0a0b1b1b1b1b1c2c2c2c2c2` is an incorrect answer given in the correct format.

Question F: Harmonic Forensics

After many weekends of tinkering, Grace has finally finished her homemade synthesizer: a chaotic mess of wires, LEDs, speakers and chips spread across a series of breadboards.

Elated, she reaches out and plays 4 notes at t=0 for a sustained period of 0.8 seconds. In addition to startling the entire household, the resulting digital signal was logged to a .csv file, sampled at intervals of 0.1 ms.

Grace's synthesizer produces pure sine waves and can play each of the 25 notes in the range C5 to C7. However, to simplify the implementation process, Grace rounded the note-specific frequencies to be integers. This is shown in the reference table below.

Using this knowledge and the `HomemadeSoundwave.csv` log file, recover each of the 4 distinct note-frequencies of the chord Grace played.

Synthesizer Reference Table

Note	Frequencies (Hz)		
C	523	1047	2093
C#/Db	554	1109	
D	587	1175	
Eb/D#	622	1245	
E	659	1319	
F	698	1397	
F#/Gb	740	1480	
G	784	1568	
Ab/G#	831	1661	
A	880	1760	
Bb/A#	932	1865	
B	988	1976	

Answer Format

Once you are confident in your answer:

1. Sort the 4 frequencies into ascending order
2. Format the integers as comma separated values, without spacing
3. Paste the resulting string into the hex-grid tool under **Question F**
4. Present your hex grid for verification

For example, `523,554,587,2093` is an incorrect answer given in the correct format.

Question G: DJ O'clock

After years of fumbling his DJ performances, Oli devised his own system for predicting whether two tracks will have a smooth transition into each other. Before performing, he carefully analyses each of the tracks in his library and assigns them an 'hour' - such as 3 o'clock in the morning.

Oli finds that using his system, he can achieve a harmonious transition by:

- Mixing between two tracks of the same hour - e.g. 7:00 AM into 7:00 AM
- Mixing between two tracks one hour apart - e.g. 7:00 AM into 6:00 AM or 8:00 AM
- Mixing between two tracks 12 hours apart - e.g. 7:00 AM into 7:00 PM

Oli swears by this system, asserting that transitions that follow these guidelines 'always sound good'. In fact, he has since subjected himself to further restrictions:

- Oli never plays any track more than once within a set
- Oli never transitions into a track more than 2 BPM slower than the one playing
- Oli never transitions into a track more than 5 BPM higher than the one playing

Oli carefully draws a map of his track library by hand in advance of his performances and refers to it throughout each set to ensure all of his transitions are fluid and harmonious. However, on this occasion Oli's hand-drawn guide got soaked in a haphazard water spill.

Oli needs you to save the day. Given his library of tracks: `DJTrackLibrary.csv`, produce a setlist of exactly **33 tracks** to be played in sequence that abide by Oli's self-imposed restrictions.

Give your answer in the form of the track ids, delimited by commas. Paste your answer into the hex-grid tool under **Question G** and present your pattern for verification.

As an example, `1,2,3,4,5` is an incorrect answer given in the correct format.

Question H: Acetate 720

Solve the puzzle presented in `Scans.pdf`.

Once you are confident in your answer, type it into the hex-grid tool under **Question H** and present your pattern for verification.