Supplementary Material for *Unicode Analogies: An Anti-Objectivist Visual Reasoning Challenge*

Steven Spratley Kris Ehinger Tim Miller
School of Computing and Information Systems, The University of Melbourne
https://github.com/SvenShade/UnicodeAnalogies

1. Unicode blocks and fonts used

Our software generates PMPs from an annotated pool of Unicode characters. To select this pool, we looked at blocks/sections in Unicode that contained characters we expected to be amenable to the assembly of novel problems. That is, blocks depicting characters that are symbolic, geometric, not overly complex, and able to express multiple concepts from our schema. We include the full lists of blocks and the fonts used to render them in Tab. 1 and Tab. 2. We also include licensing details and websites for all fonts in Tab. 3.

2. Sample problems

To further aid researchers in exploring the dataset and its current conceptual schema, in this document we provide just under a hundred PMPs, each with a description of the depicted rule and concept, and the correct answer frame. Answers are numbered from 1 to 4, starting top left and continuing left to right, top to bottom. All PMPs shown here are generated without context shift, to more easily communicate the base rule. As discussed in the main paper, these problems are generated with a similar philosophy to Bongard problems, where ideally the most elegant solution will be the intended one.

Like Bongard, we wish to point out the polysemy of characters used to assemble our problems. Take the problem shown in Fig. 5, for example. Without context, the second frame of the top row could depict two black objects. Within this problem, and to most humans, it is perceived as a square due to gestalt closure. In this dataset, it is also used in problems that play with low and high ink level, in problems that involve triangles, in problems that explore diagonally-arranged objects, the number of solid components, an equal aspect ratio, a centred mass, two base contacts, flat base contacts... and the list goes on. Settling on a perceptual take is therefore a question of context and of usefulness, and not of assigning an objective label.

3. Ethics approvals

In keeping with expectations for ethical human research, all facets of our human baseline experiment, including design, methodology, and plain language statement and consent forms, were approved by the relevant research ethics board (to be disclosed after blind review). Our dataset contains no personally identifiable information, with participant IDs being used for the sole purpose of ensuring a single set of fifteen solved problems per participant, before being deleted.

4. Defining custom schemas

Unicode Analogies is intended to be a framework that facilitates the use of PMPs in research across multiple fields. As such, there is the opportunity for researchers to define their own schemas and generate new problems. Encoding a designed schema is straight-forward, and involves typing out a `classes` dictionary consisting of concept keys and applicable rule values, in Python script. The software loads this dictionary and, given parameters to define the split (e.g. sampling diversity), will begin generating problems by loading images directly from their annotation folders on disk. This also makes the annotation process simple, as it requires no additional software or metadata; images need to either be generated and saved to individual folders (or placed there during manual annotation, as was the case with the Unicode characters used in this paper), and the software will load them and begin assembly.

5. Further details on training

To promote reproducibility, all data and code, along with scripts to automate all experiments, is released on our project repository. In addition to this resource, we wish to mention further details regarding the training process. MRNet was trained with both cross-entropy and multi-head losses. Blind, ResNet, RelBase, and MRNet models were all trained with dropout on both spatial and fully-connected layers, set to 0.1 and 0.5 respectively. For pragmatic rea-

sons, SCL models were trained using a higher batch size, permitted by relatively fewer trainable parameters. We limited training MRNet to 20 epochs due to overfitting; we expect this is due to it possessing an order of magnitude more trainable parameters than other models. In Tab. 4, we tabulate further information. The average times per epoch are reported on a system using PyTorch's `DataParallel` feature to train models across twin NVIDIA 2080Ti GPUs.

All dataset splits, with the exception of the split used for establishing a human baseline (and subsequent *Challenge* split), possess a train-validation-test ratio of 70-3-27. This can be easily changed in software to facilitate different experiments. Note that this split ratio simultaneously defines both the ratio of characters used to form problems in each partition, as well as the ratio of problems formed. In *Extrapolation* / *Extrapolation-plus* splits, it also defines the ratio of held-out problem tuples / class types.

The number of problems available to each split is highly dependent on the split parameters used. If there are weak restrictions governing character hold-out and sampling diversity (for instance, if no hold-out is requested, or if the system is permitted to re-instantiate a problem tuple with the same answer), then the full number of problems requested (default = 10000) will likely be generated. Given that character hold-out and sampling diversity are key design considerations and both greatly contribute to this dataset's challenging nature, some splits will not make up the full 10000 problems. While most splits remained between 8000-10000 as reported in the main paper, the *Union* split is only able to generate on average, 2300 problems per fold. This is due to answers to union problems being more scarce (they need to depict the right intersection of concepts), as well as there being a limited number of concepts in the schema applicable to union problems. The training partitions of all dataset splits contain both context-shifted and non-shifted problems, regardless of whether validation and testing is requested to contain shifted problems, as this enables the generation of far more problems (and assumedly, training more capable models) given limited annotations.

Unicode block	Name	Rendering font
16A0|16FF	Runic	Alphabetum
10C80|10CFF	Old Hungarian	”
11000|1107F	Brahmi	”
10100|1013F	Aegean Numbers	”
10300|1032F	Old Italic	”
103A0|103DF	Old Persian	”
10800|1083F	Cypriot Syllabary	”
12400|1247F	Cuneiform Numbers and Punctuation	”
A6A0|A6FF	Bamum	Google Noto Bamum
16800|16A3F	Bamum Supplement	”
A000|A48F	Yi Syllables	Google Noto Yi
A490|A4CF	Yi Radicals	”
2D30|2D7F	Tifinagh	Google Noto Tifinagh
10600|1077F	Linear A	CTAN Linear A
1E800|1E8DF	Mende Kikakui	Mende Kikakui
1BC00|1BC9F	Duployan	Duployan
16F00|16F9F	Miao	Miao Unicode
1D800|1DAAF	Sutton SignWriting	Sutton SignWriting
10380|1039F	Ugaritic	Quivira
680|169F	Ogham	”
1400|167F	Unified Canadian Aboriginal Syllabics	”
18B0|18FF	Unified Canadian Aboriginal Syllabics Extended	”
A700|A71F	Modifier Tone Letters	”
0000|007F	Basic Latin	Symbola 10.24
0080|00FF	Latin-1 Supplement	”
0100|017F	Latin Extended-A	”
0180|024F	Latin Extended-B	”
0250|02AF	IPA Extensions	”
02B0|02FF	Spacing Modifier Letters	”
0300|036F	Combining Diacritical Marks	”
0370|03FF	Greek and Coptic	”
0400|04FF	Cyrillic	”
0500|052F	Cyrillic Supplement	”
2000|206F	General Punctuation	”
2070|209F	Superscripts and Subscripts	”
20A0|20CF	Currency Symbols	”
20D0|20FF	Combining Diacritical Marks for Symbols	”
2100|214F	Letterlike Symbols	”

Table 1. Unicode blocks used, with the fonts used to render them.

Unicode block (continued)	Name	Rendering font
2150|218F	Number Forms	Symbola 10.24
2190|21FF	Arrows	”
2200|22FF	Mathematical Operators	”
2300|23FF	Miscellaneous Technical	”
2460|24FF	Enclosed Alphanumerics	”
2500|257F	Box Drawing	”
2580|259F	Block Elements	”
25A0|25FF	Geometric Shapes	”
2600|26FF	Miscellaneous Symbols	”
2700|27BF	Dingbats	”
27F0|27FF	Supplemental Arrows-A	”
1D000|1D0FF	Byzantine Musical Symbols	”
1D100|1D1FF	Musical Symbols	”
1D00|1D7F	Phonetic Extensions	”
2440|245F	Optical Character Recognition	”
27C0|27EF	Miscellaneous Mathematical Symbols-A	”
2800|28FF	Braille Patterns	”
2900|297F	Supplemental Arrows-B	”
2980|29FF	Miscellaneous Mathematical Symbols-B	”
2A00|2AFF	Supplemental Mathematical Operators	”
2B00|2BFF	Miscellaneous Symbols and Arrows	”
2E00|2E7F	Supplemental Punctuation	”
4DC0|4DFF	Yijing Hexagram Symbols	”
FE20|FE2F	Combining Half Marks	”
1D200|1D24F	Ancient Greek Musical Notation	”
1D300|1D35F	Tai Xuan Jing Symbols	”
1D360|1D37F	Counting Rod Numerals	”
1D400|1D7FF	Mathematical Alphanumeric Symbols	”
1F100|1F1FF	Enclosed Alphanumeric Supplement	”
1F300|1F5FF	Miscellaneous Symbols and Pictographs	”
1F700|1F77F	Alchemical Symbols	”
1F780|1F7FF	Geometric Shapes Extended	”
1F030|1F09F	Domino Tiles	”
1D2E0|1D2FF	Mayan Numerals	Babelstone Han
3000|303FCJK	Symbols and Punctuation	”
FE30|FE4FCJK	Compatibility Forms	”
1FB00|1FBFF	Symbols for Legacy Computing	Legacy Computing Font

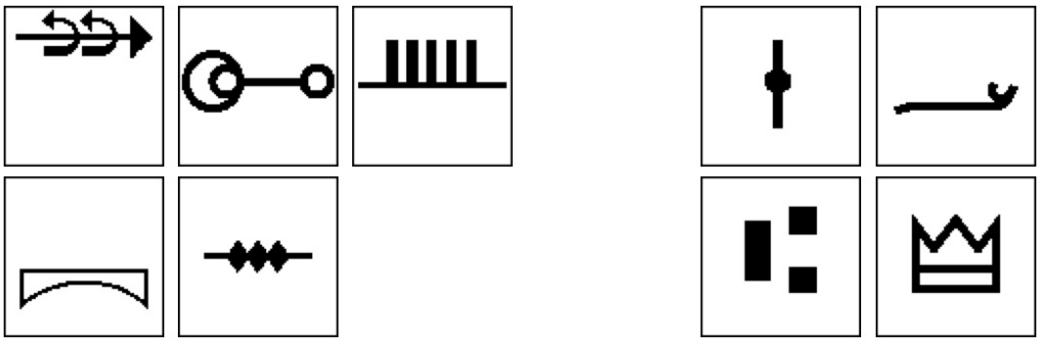
Table 2. Unicode blocks used, with the fonts used to render them (continued).

Rendering font	Licence	Website
Alphabetum	Paid publishing licence	http://guindo.pntic.mec.es/ jmag0042/alphaeng.html
Google Noto Bamum	OFL 1.1	https://fonts.google.com/
Google Noto Yi	”	”
Google Noto Tifinagh	”	”
CTAN Linear A	The LATEX Project Public Licence	https://ctan.org/pkg/lineara
Mende Kikakui	OFL 1.1	https://athinkra.github.io/mende-kikakui/
Duployan	”	https://github.com/dscorbett/duployan-font
Miao Unicode	”	https://github.com/phjamr/MiaoUnicode
Sutton SignWriting	”	https://slevinski.github.io/SuttonSignWriting/
Quivira	Public domain / unrestricted	http://www.quivira-font.com/
Symbola 10.24	Freeware ($\leq$ Symbola 10.24)	https://packages.fedoraproject.org/pkgs/gdouros-symbola-fonts
Babelstone Han	OFL 1.1	https://www.babelstone.co.uk/Fonts/Han.html
Legacy Computing Font	”	https://github.com/dokutan/legacy_computing-font/

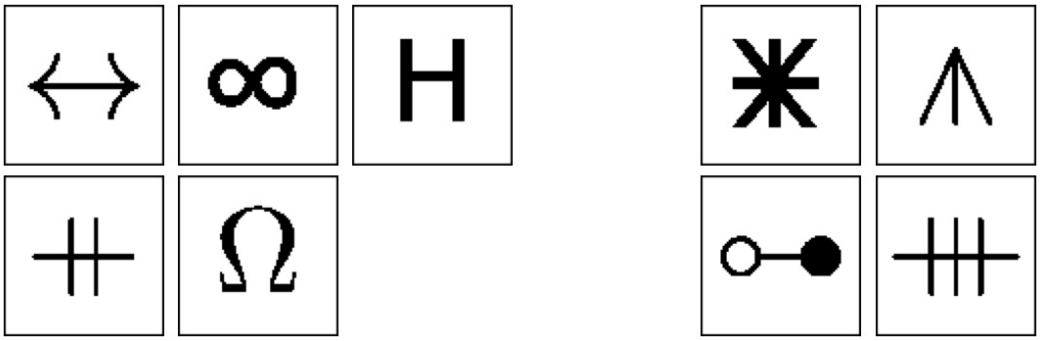
Table 3. Fonts used, with their licences and website details.

Model architecture	Learning rate	Batch size	Max. epochs	Avg. time per epoch (s)
Context-blind	3e-4	32	60	2.45
ResNet	3e-4	32	60	3.73
RelBase	3e-4	32	60	9.97
MRNet	1e-3	32	20	19.15
SCL	1e-3	128	60	3.64

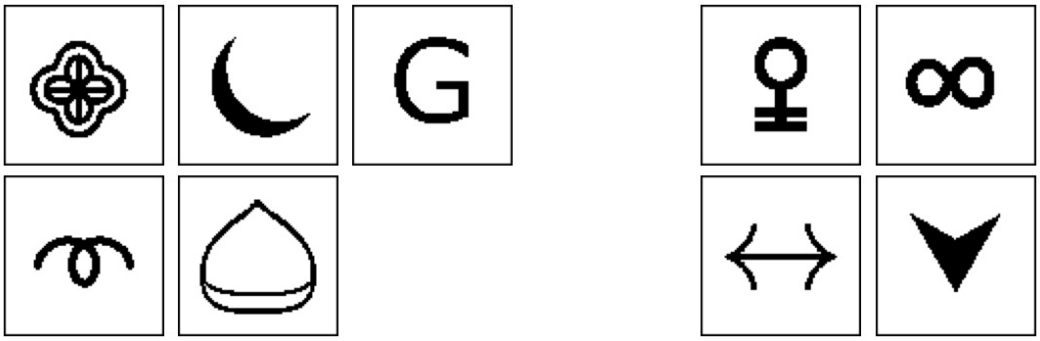
Table 4. Further information on hyperparameters used in training models.



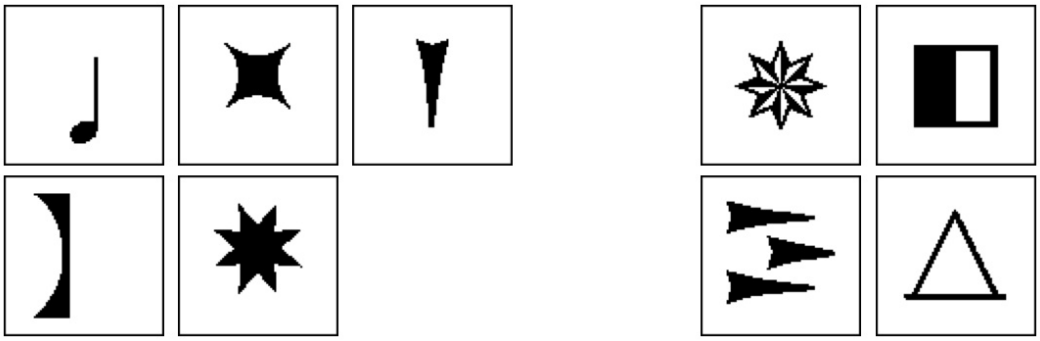
Problem 1. Constant aspect ratio (wide). Answer=2.



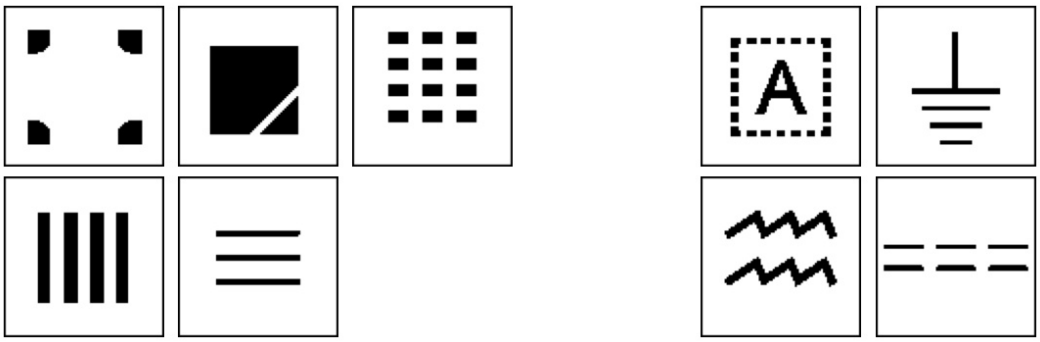
Problem 2. Constant number of base contacts (two). Answer=3.



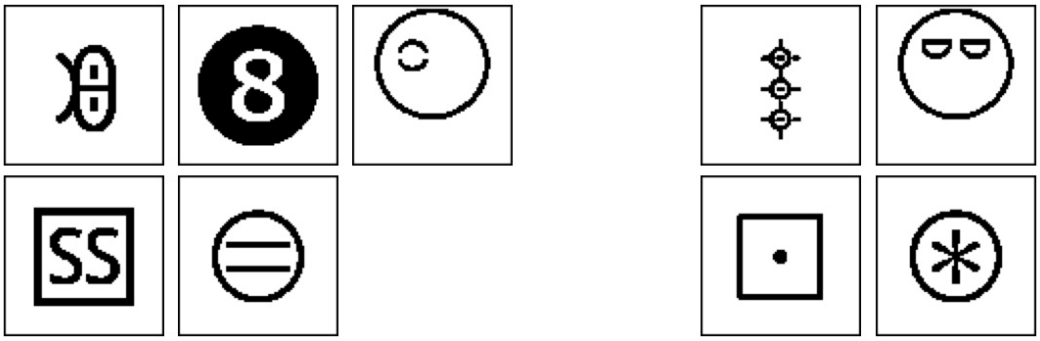
Problem 3. Constant base style (curved base). Answer=2.



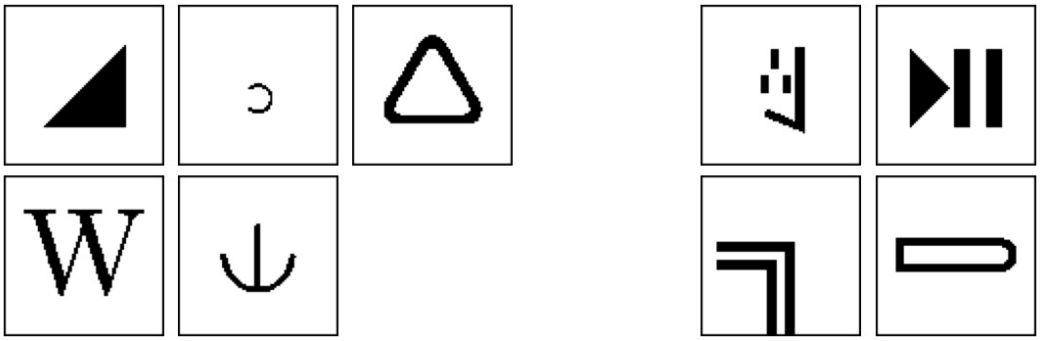
Problem 4. Constant closed fill (full). Answer=3



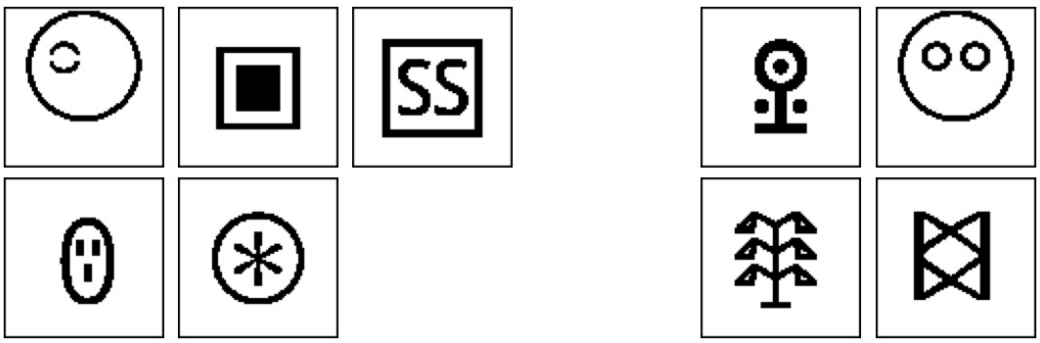
Problem 5. Constant closure pattern (square). Answer=1.



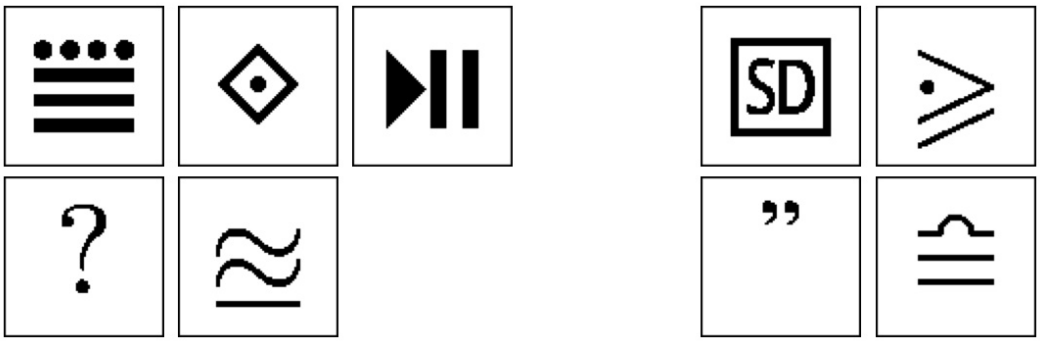
Problem 6. Constant number of internal solid components (two). Answer=2.



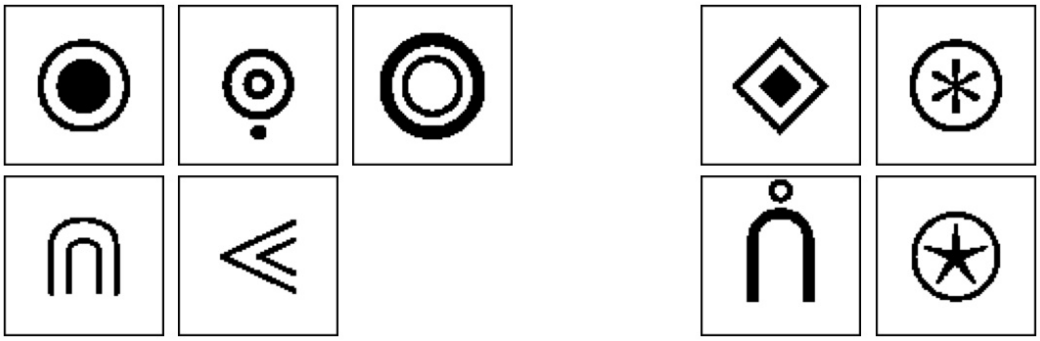
Problem 7. Constant number of total solid components (one). Answer=4.



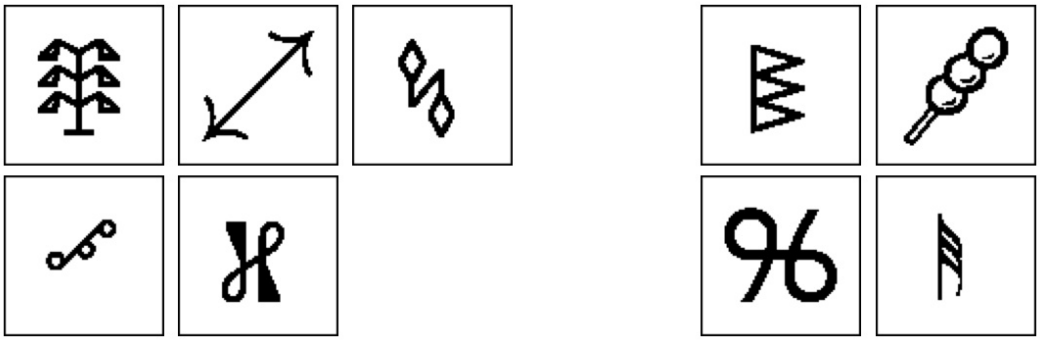
Problem 8. Constant number of internal spaces (one). Answer=1.



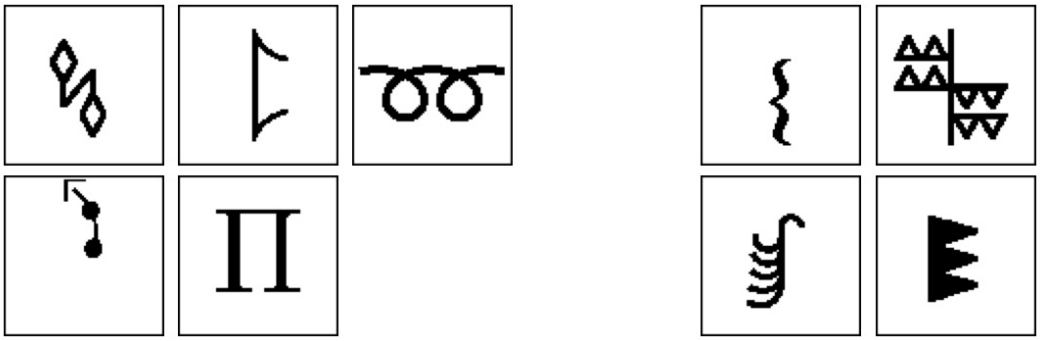
Problem 9. Constant number of unique solid components (two). Answer=4.



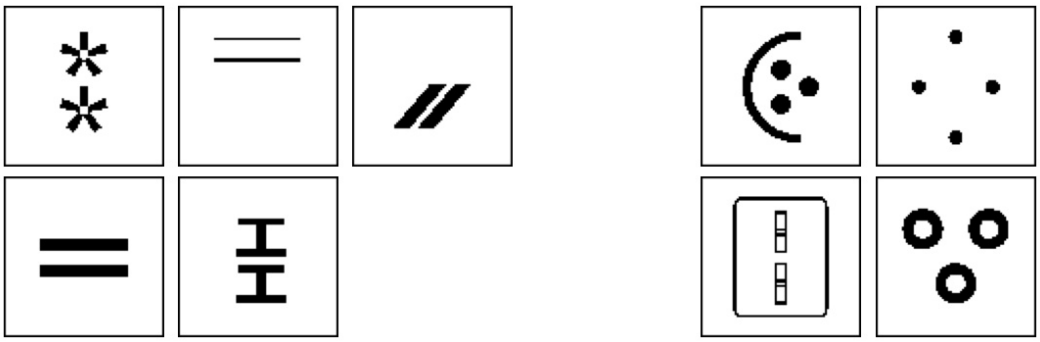
Problem 10. Constant in exhibiting concentric shapes. Answer=1.



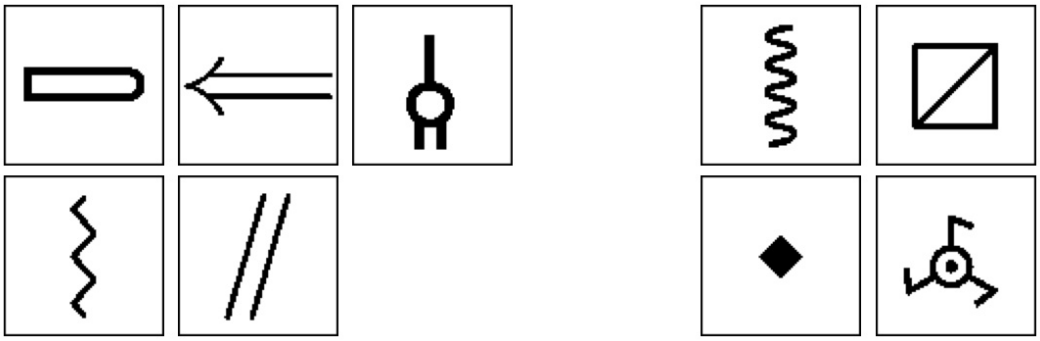
Problem 11. Constant in exhibiting connected items facing different directions. Answer=3.



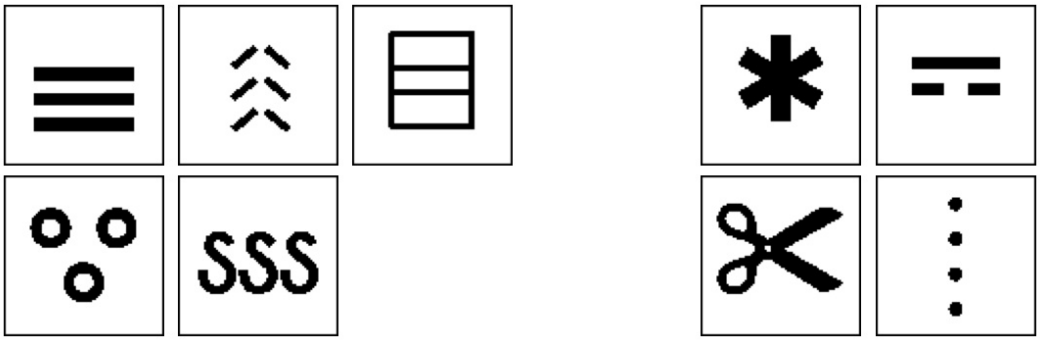
Problem 12. Constant number of connected items (two). Answer=1.



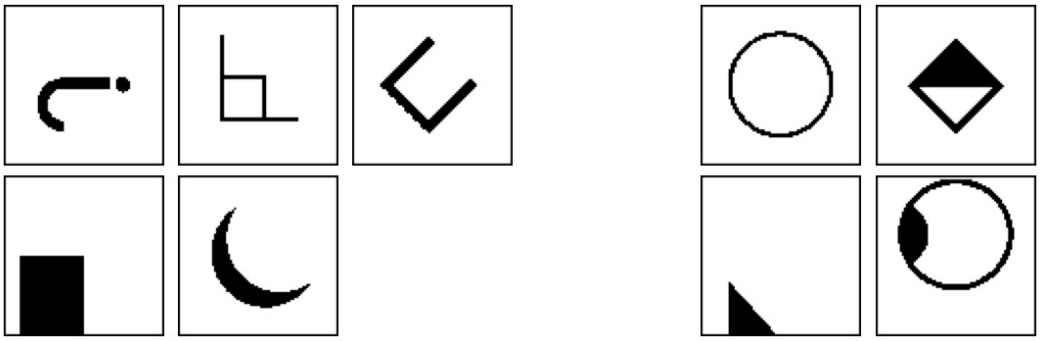
Problem 13. Constant number within groups (two). Answer=3.



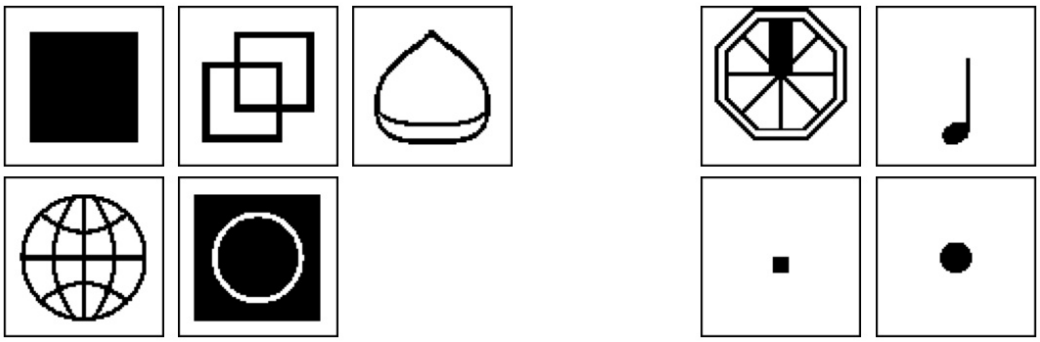
Problem 14. Constant elongation (high). Answer=1.



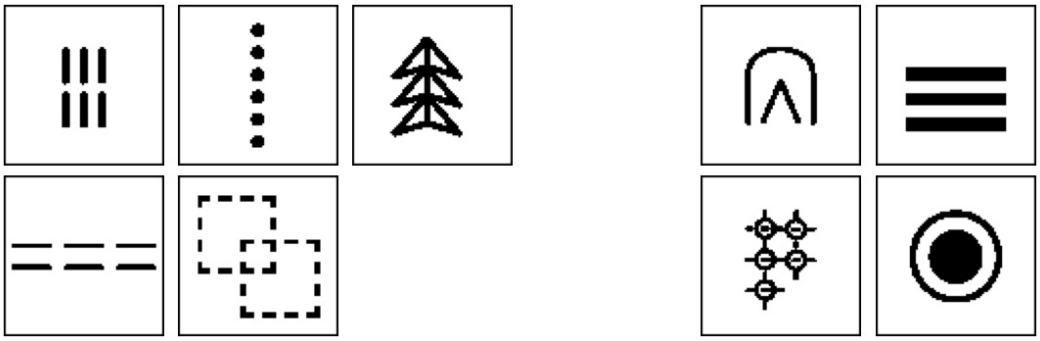
Problem 15. Constant gestalt number (three). Answer=2.



Problem 16. Constant global mass centroid (South-West). Answer=3.

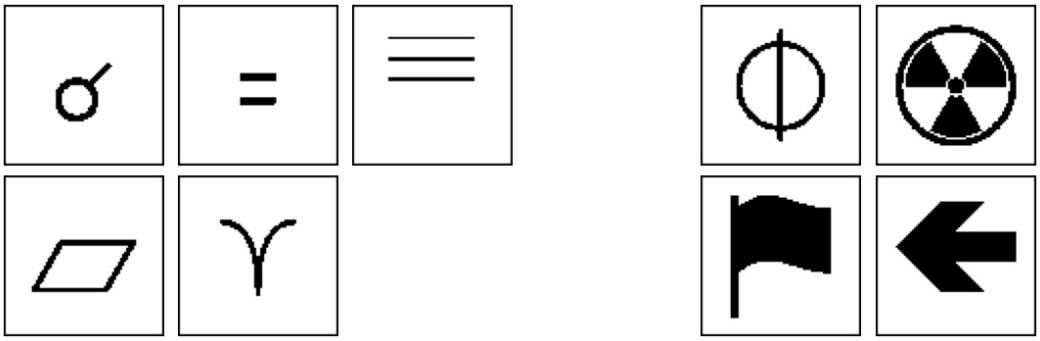


Problem 17. Constant global size (large). Answer=1.

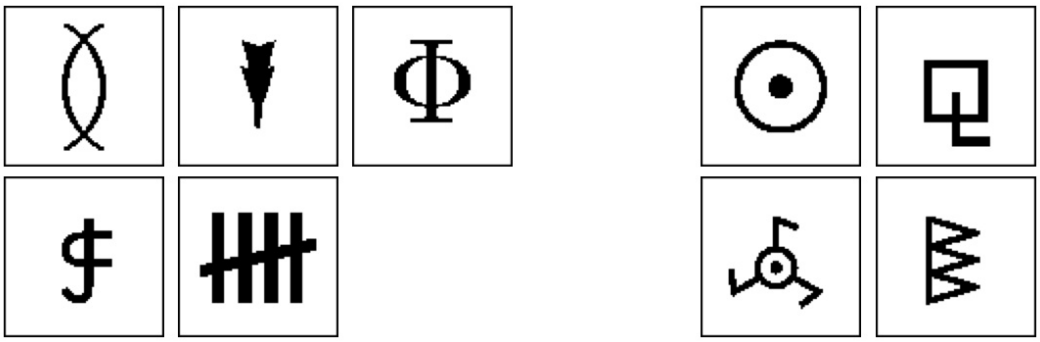


Problem 18. Constant in exhibiting “balanced” groups (same object and with symmetry). Answer=2.

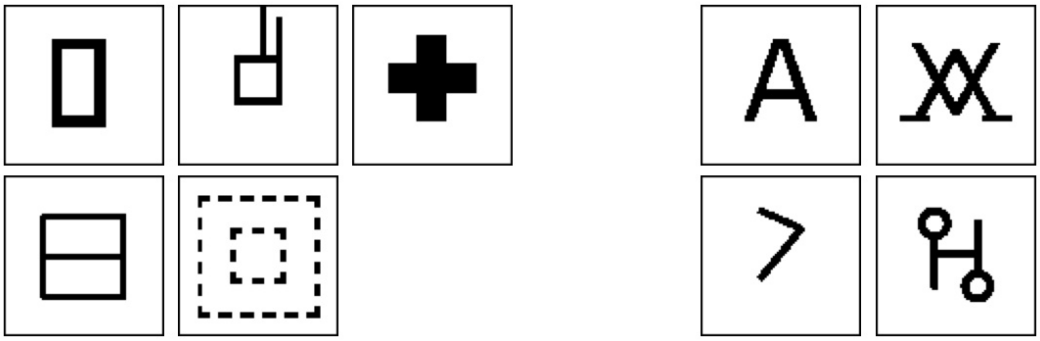
Problem 19. Constant horns. Answer=2.



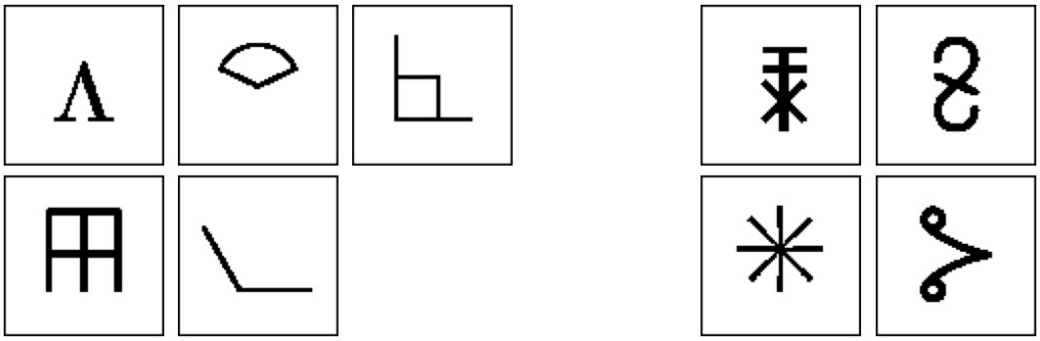
Problem 20. Constant ink level (low). Answer=1.



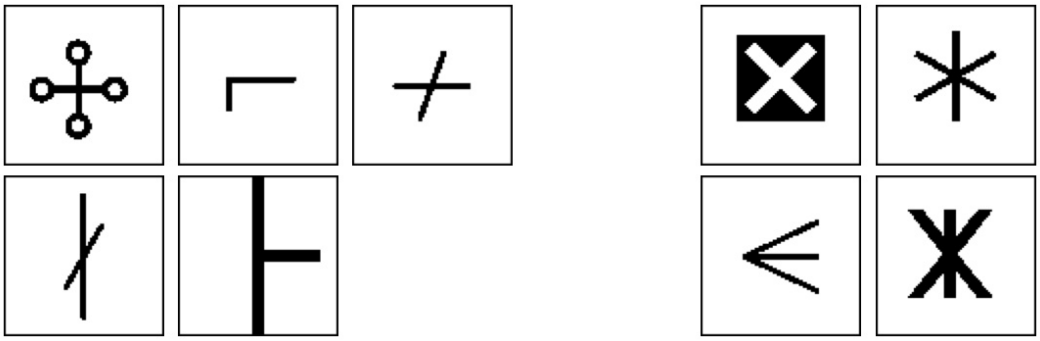
Problem 21. Constant interaction (overlap). Answer=2.



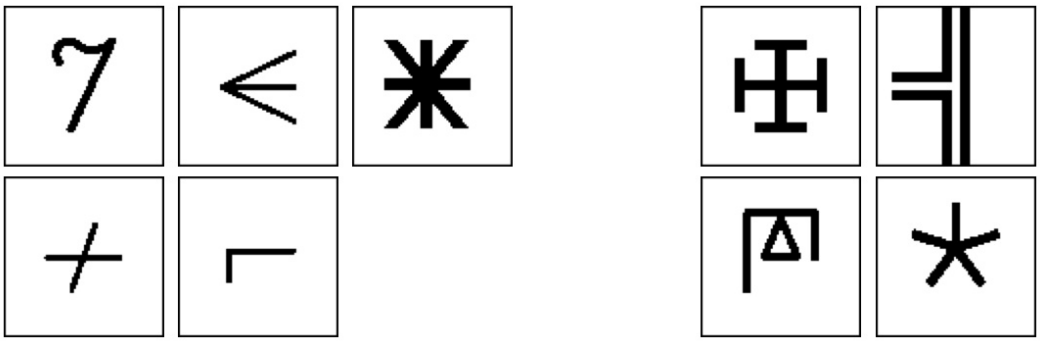
Problem 22. Constant in exhibiting a type of intersection (orthogonal). Answer=4.



Problem 23. Constant in exhibiting an intersection with a number of emanating lines (two). Answer=4.

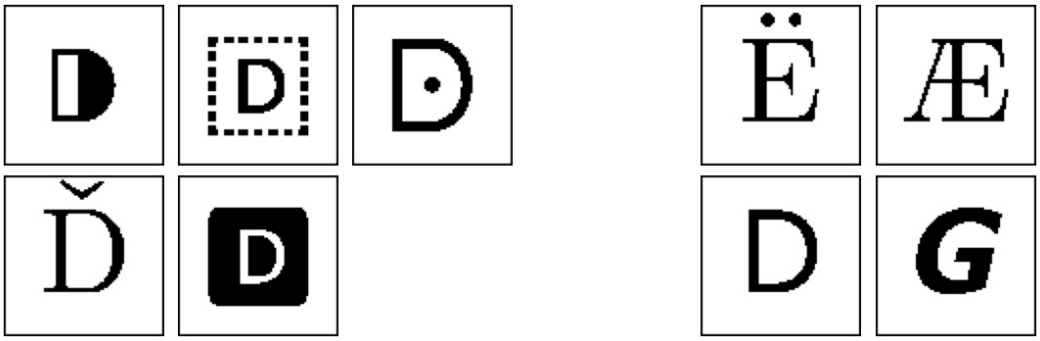


Problem 24. Constant in exhibiting an intersection with a minimum number of lines (two). Answer=1.



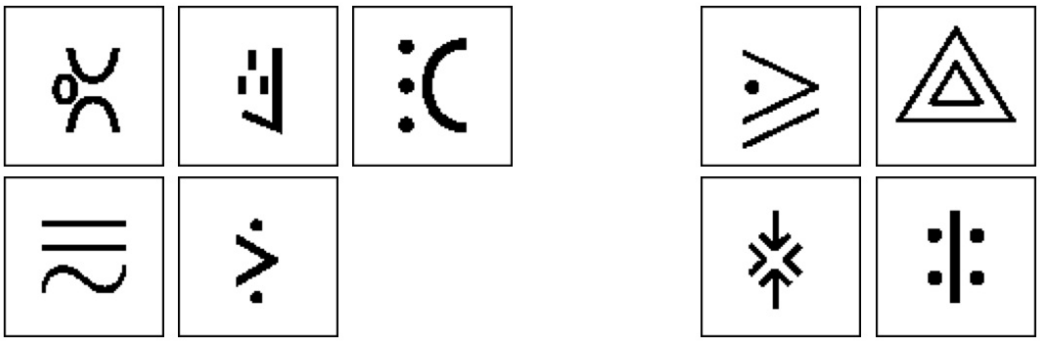
Problem 25. Constant number of intersections (one). Answer=4.

Problem 26. Constant in character style (bold). Answer=3.

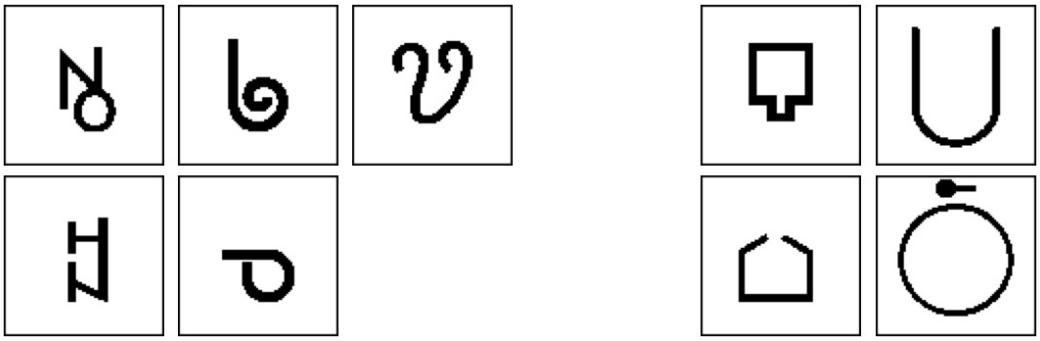


Problem 27. Constant Latin character (uppercase-D). Answer=3.

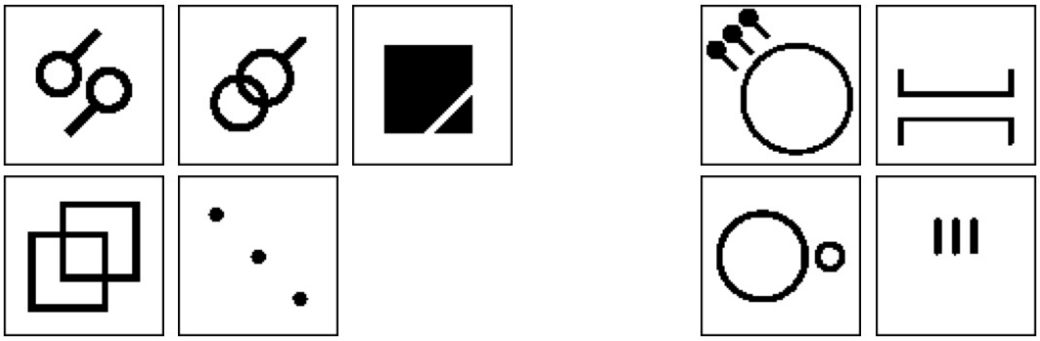
Problem 28. Constant in exhibiting negative space. Answer=2.



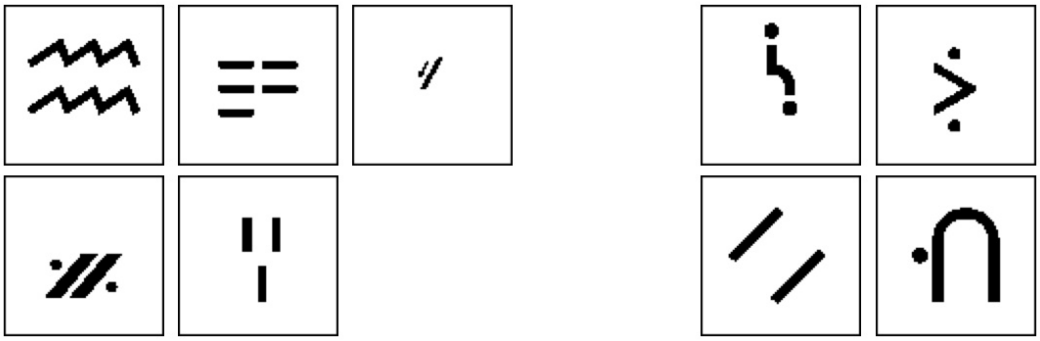
Problem 29. Constant in exhibiting an odd-one-out scenario. Answer=4.



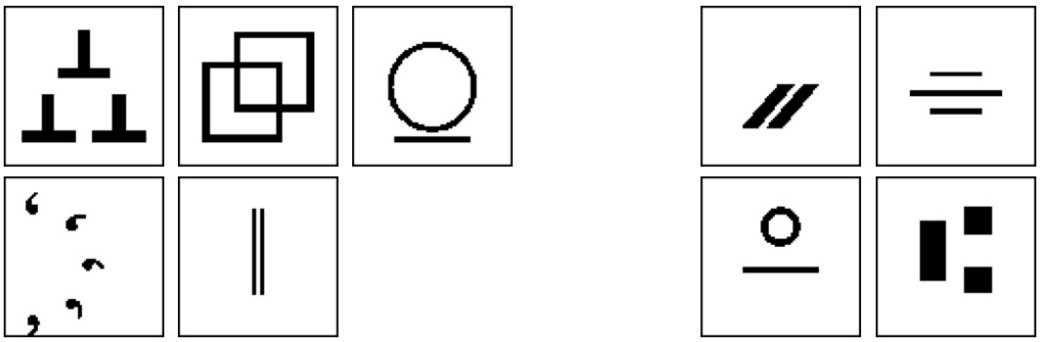
Problem 30. Constant width of opening (narrow). Answer=3.



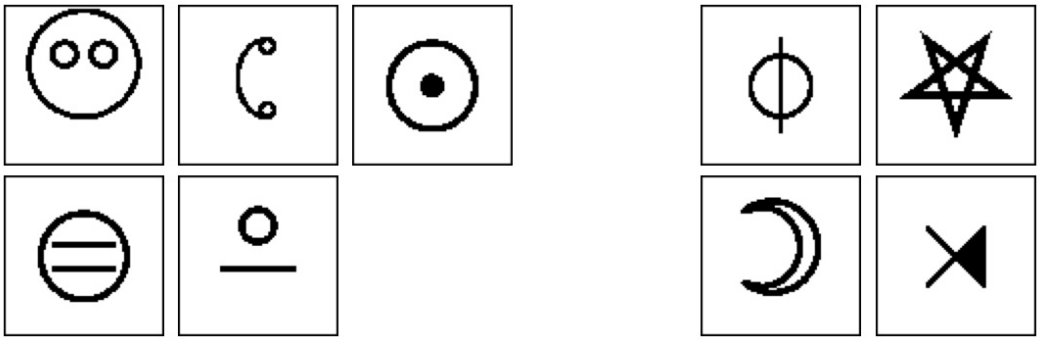
Problem 31. Constant in exhibiting a type of relational position (diagonal). Answer=1.



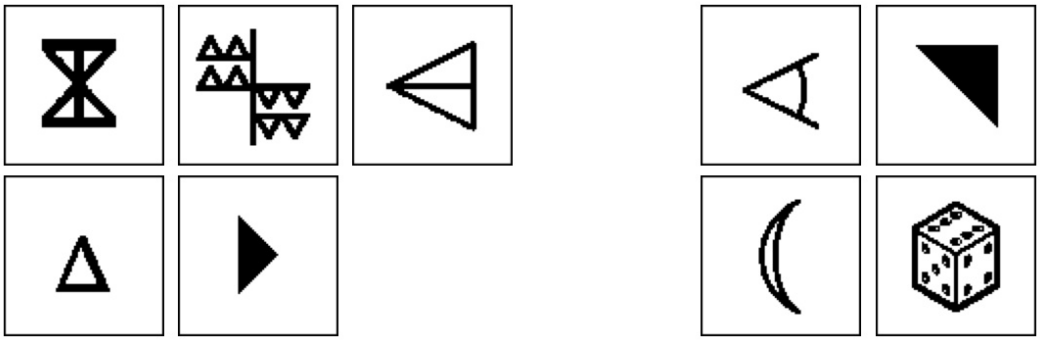
Problem 32. Constant in exhibiting a type of relational rotation (parallel). Answer=3.



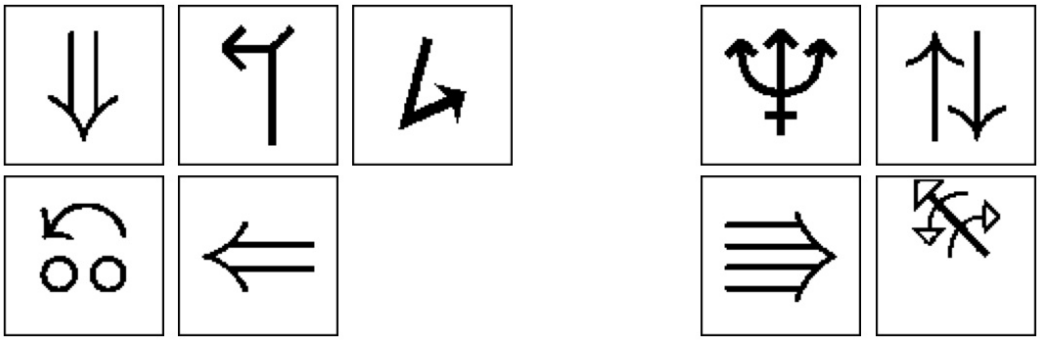
Problem 33. Constant in relational size (equal). Answer=1.



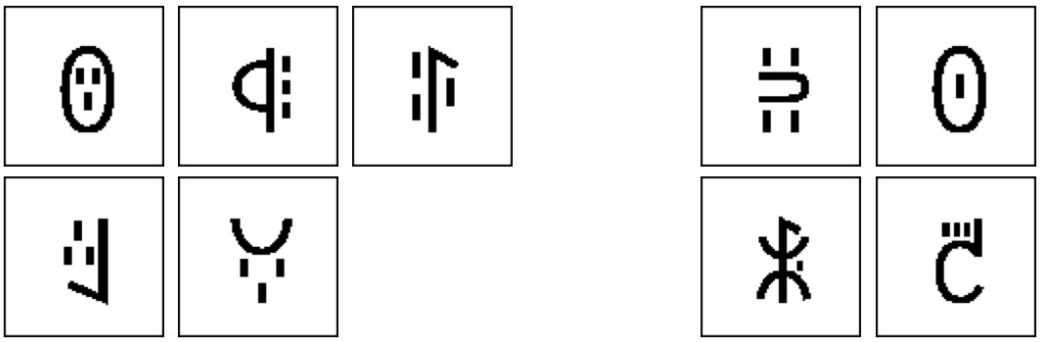
Problem 34. Constant in number of shape sides (one). Answer=1.



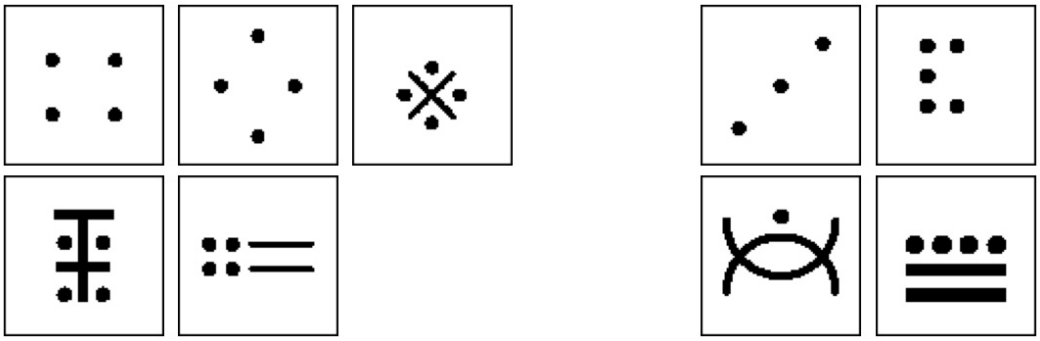
Problem 35. Constant in shape type (triangle). Answer=2.



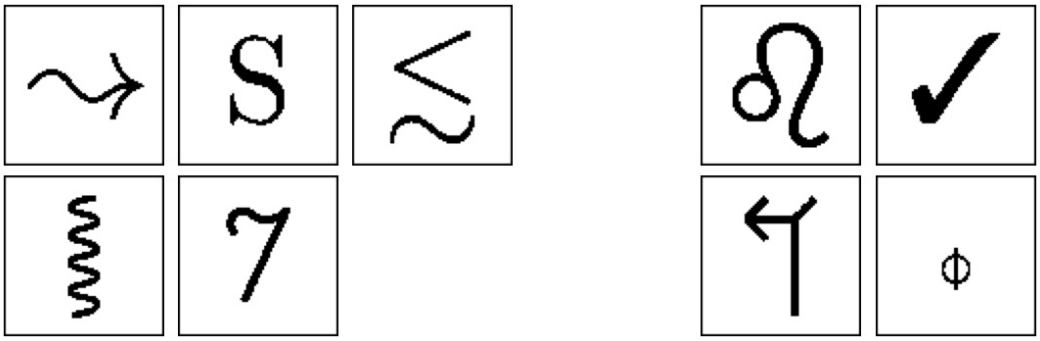
Problem 36. Constant number of arrows (one). Answer=3.



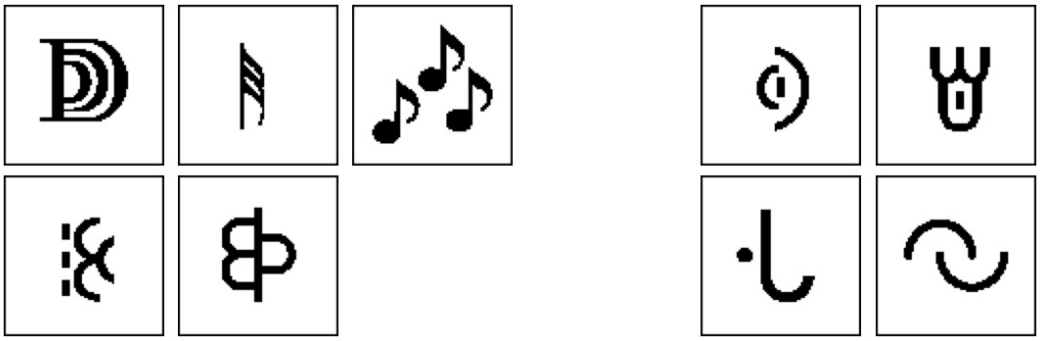
Problem 37. Constant number of dashes (three). Answer=4.



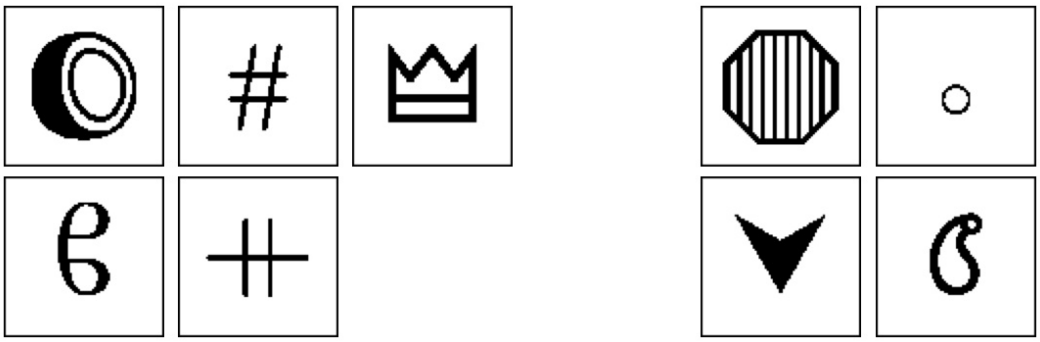
Problem 38. Constant number of dots (four). Answer=4.



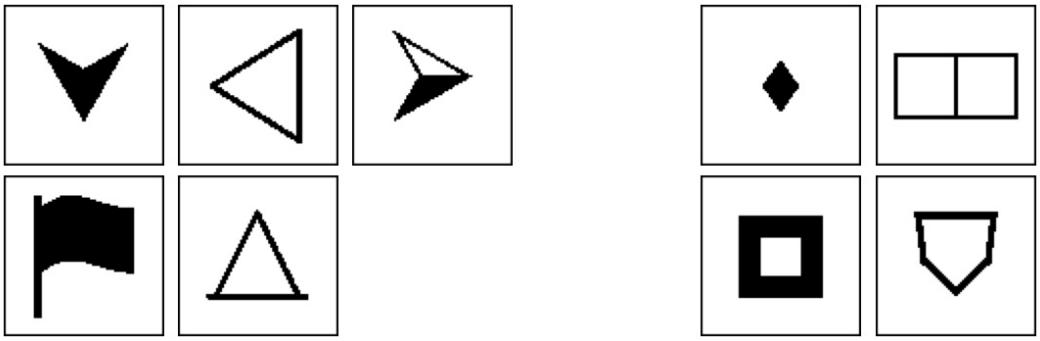
Problem 39. Constant in exhibiting a particular stroke feature (“wiggle”). Answer=1.



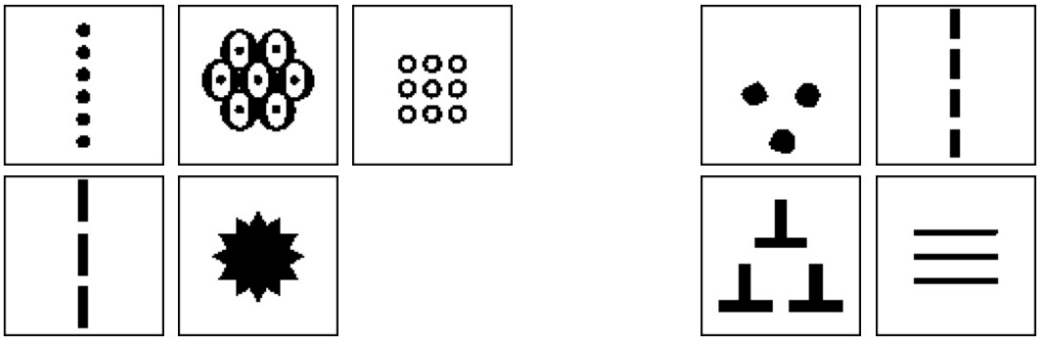
Problem 40. Constant number of curved line strokes (three). Answer=2.



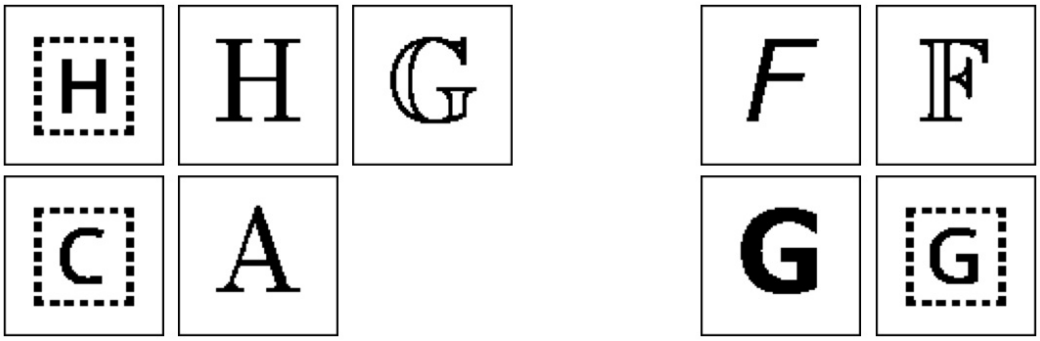
Problem 41. Distribute three base style (curved, point, flat). Answer=1.



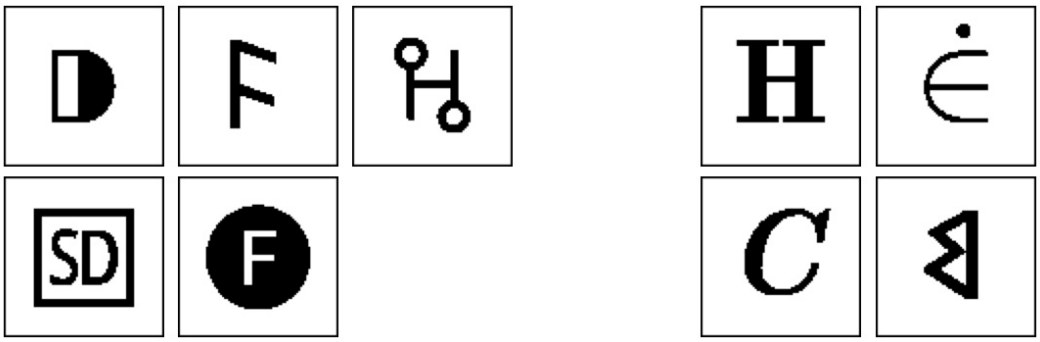
Problem 42. Distribute three in closed fill style (full, empty, half-shaded). Answer=3.



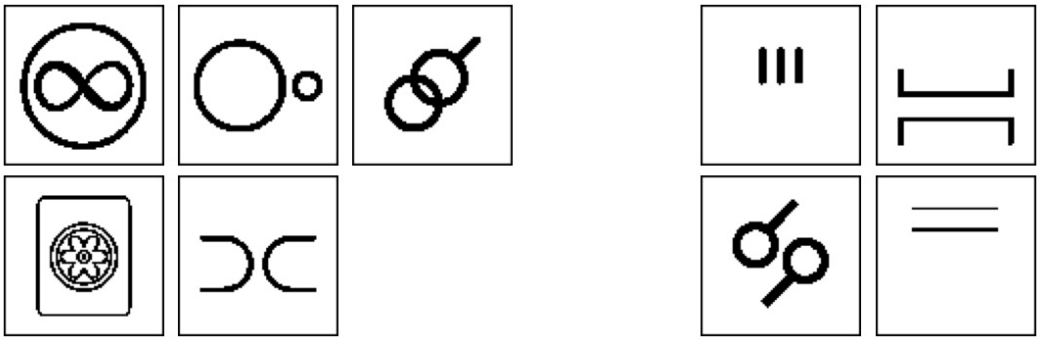
Problem 43. Distribute three closure shapes (line, circle, square). Answer=4.



Problem 44. Distribute three character style (dashed, thin, empty-bold). Answer=2.

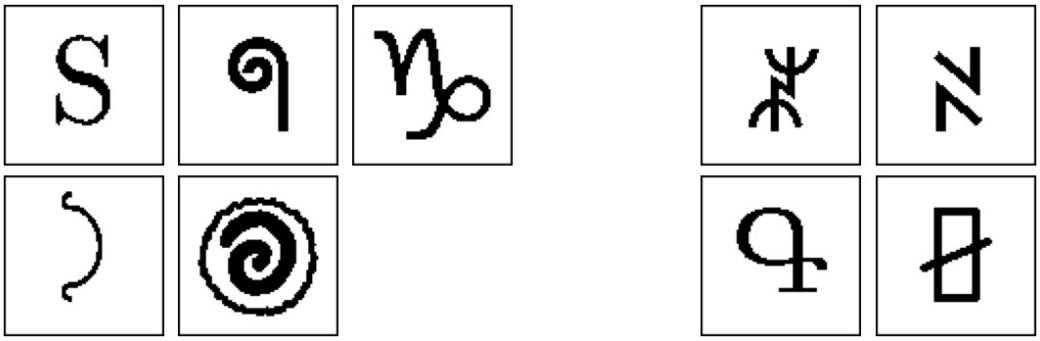


Problem 45. Distribute three Latin characters (D, F, H). Answer=4.

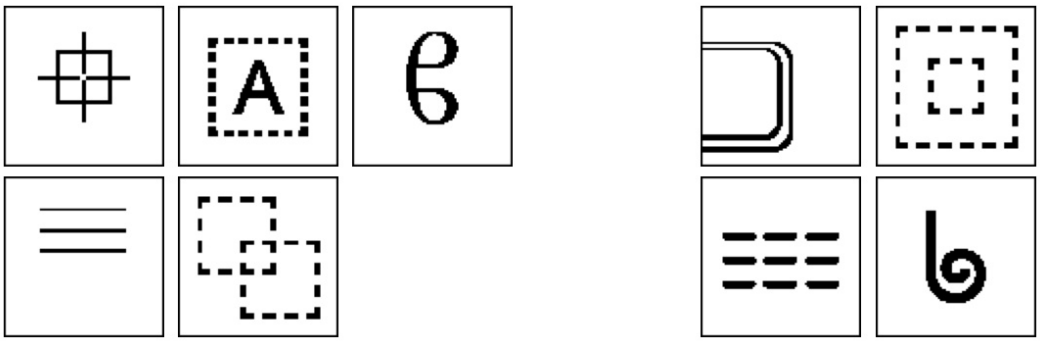


Problem 46. Distribute three relational positions (centre, horizontal, diagonal). Answer=3.

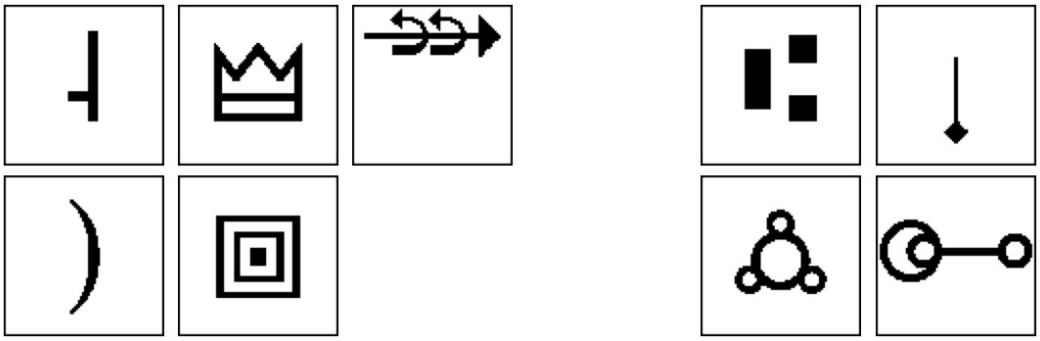
Problem 47. Distribute three kinds of relational rotation (parallel, towards, converging). Answer=2.



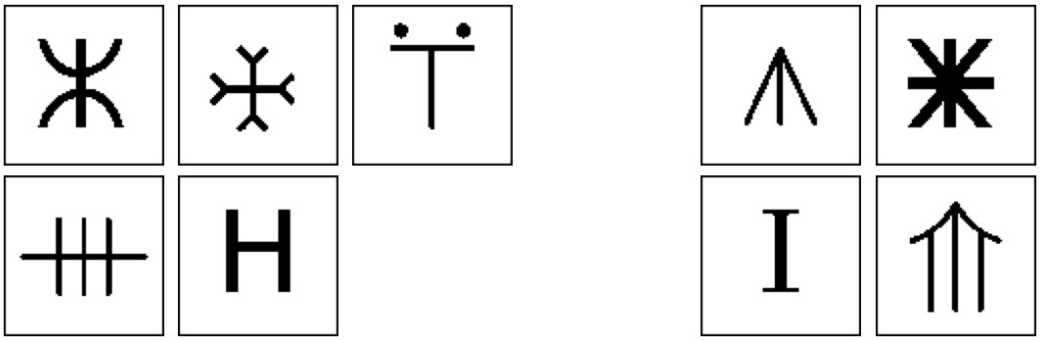
Problem 48. Distribute three line stroke features (wiggle, spiral, loop). Answer=3.



Problem 49. Distribute three line rendering styles (thin, dashed, bold). Answer=4.



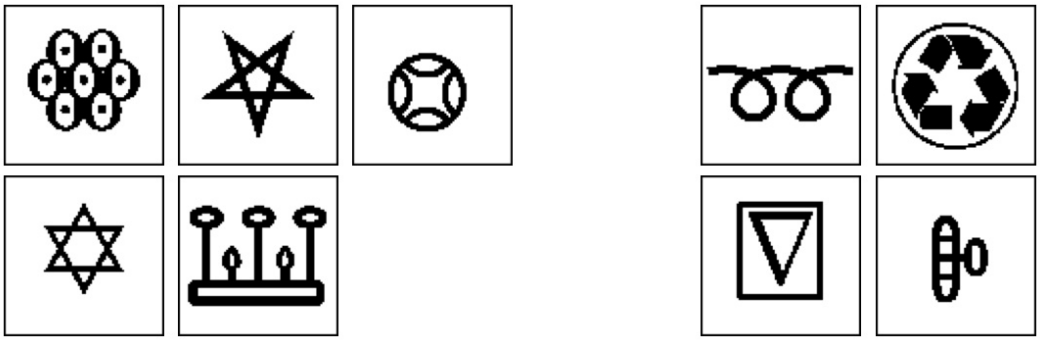
Problem 50. Progression in aspect ratio (tall, square, wide). Answer=4.



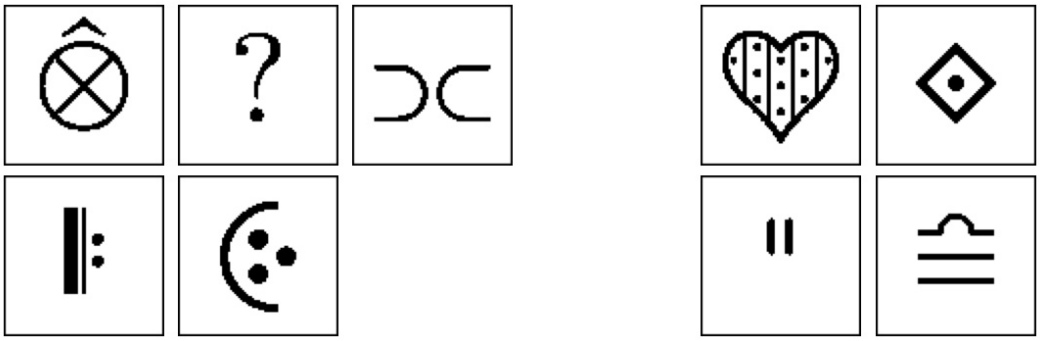
Problem 51. Progression in number of base contacts (3, 2, 1). Answer=3.

Problem 52. Progression in number of internal solid components (1, 2, 3). Answer=1.

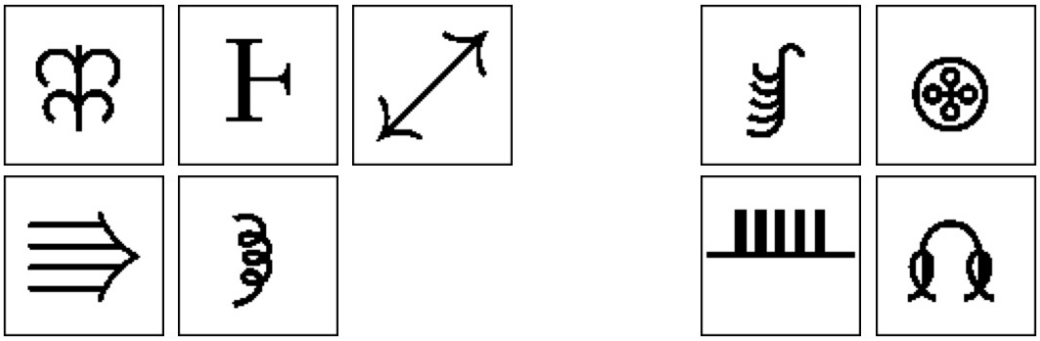
Problem 53. Progression in number of total solid components (2, 3, 4). Answer=4.



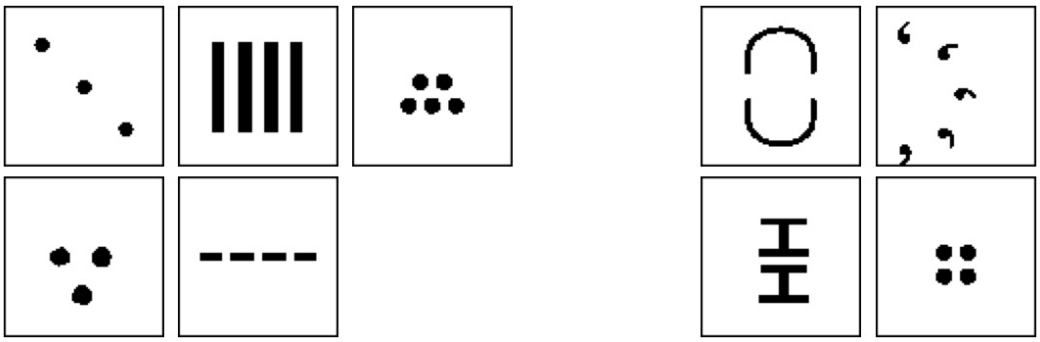
Problem 54. Progression in number of internal spaces (7, 6, 5). Answer=4.



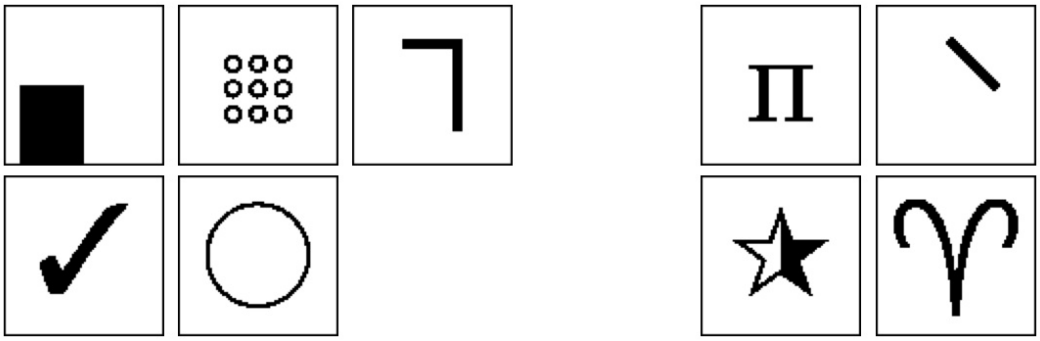
Problem 55. Progression in number of unique solid components (3, 2, 1). Answer=3.



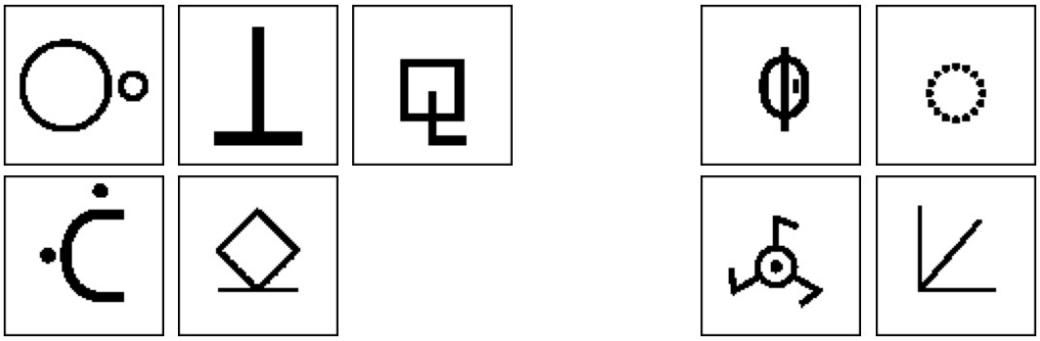
Problem 56. Progression in number of connected components (4, 3, 2). Answer=4.



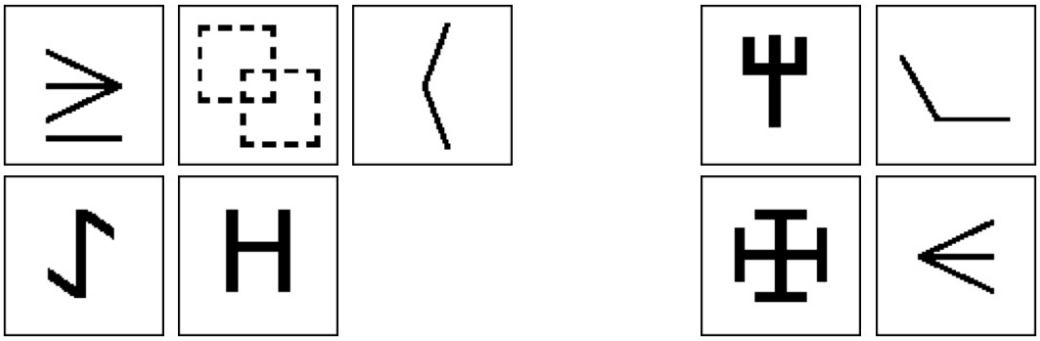
Problem 57. Progression in number of disconnected components (3, 4, 5). Answer=2.



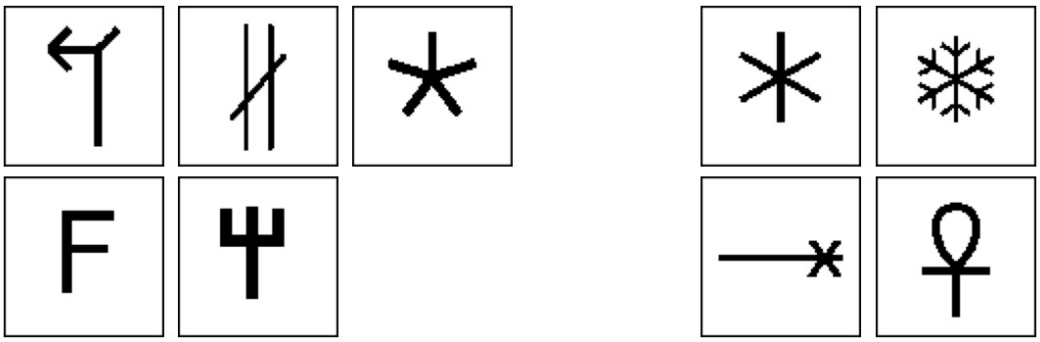
Problem 58. Progression / movement in global mass centroid (South-West, Centre, North-East). Answer=2.



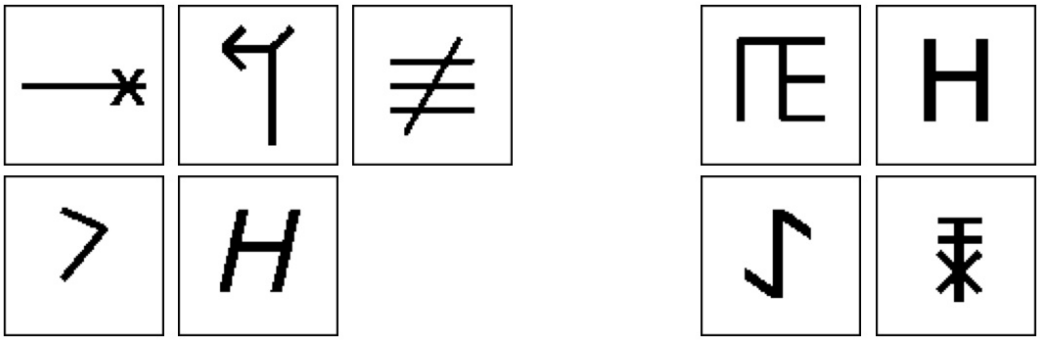
Problem 59. Progression in interaction type (none, touching, overlap). Answer=1.



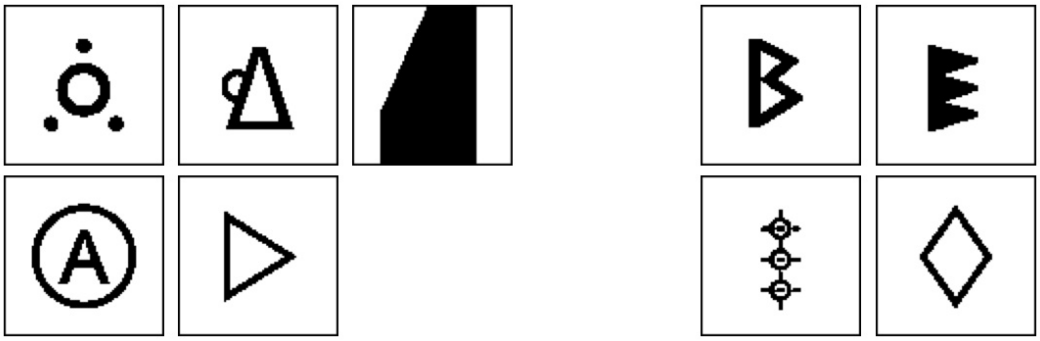
Problem 60. Progression in angle of a contained intersection (acute, orthogonal, obtuse). Answer=2.



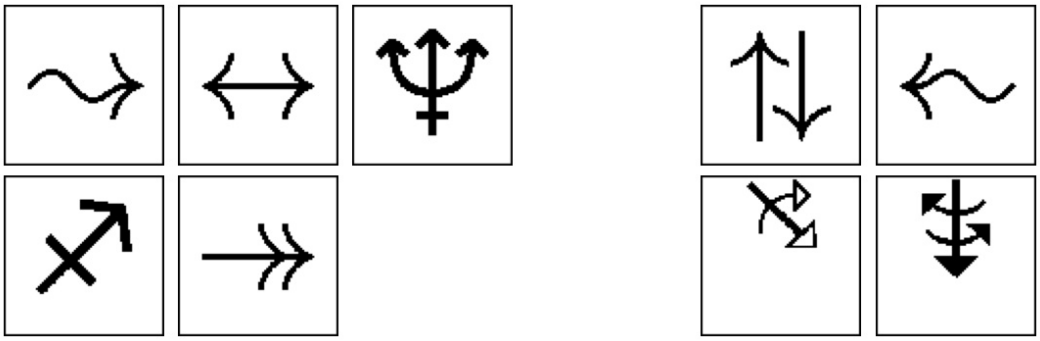
Problem 61. Progression in number of emanating lines from a contained intersection (3, 4, 5). Answer=4.



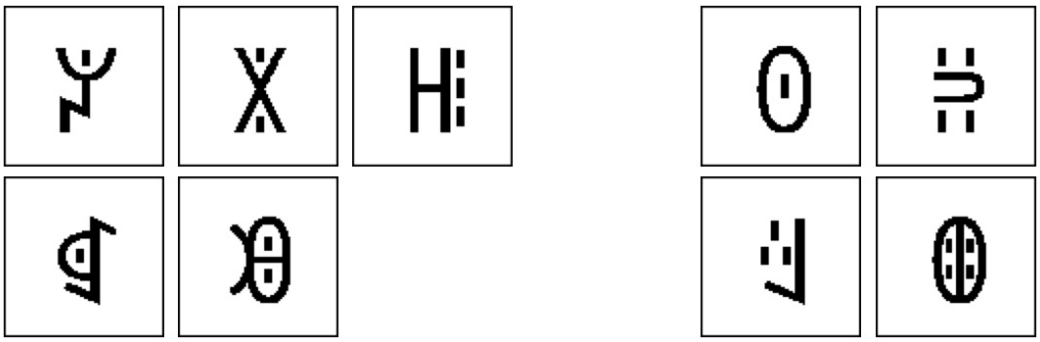
Problem 62. Progression in number of intersections (1, 2, 3). Answer=4.



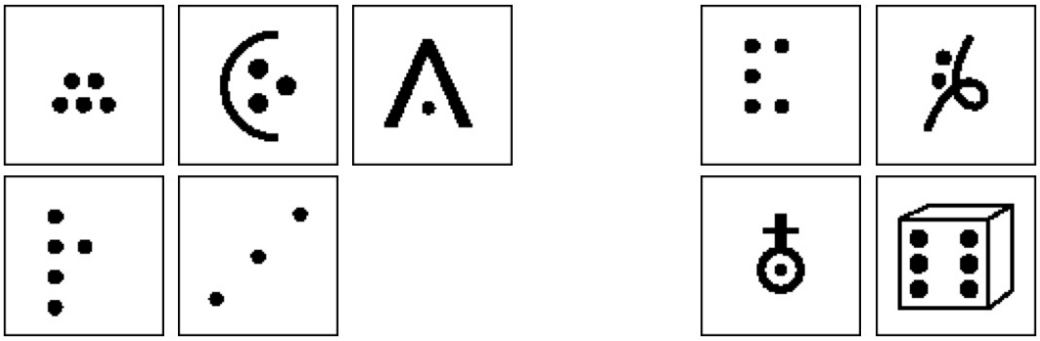
Problem 63. Progression in number of shape sides (1, 3, 5). Answer=1.



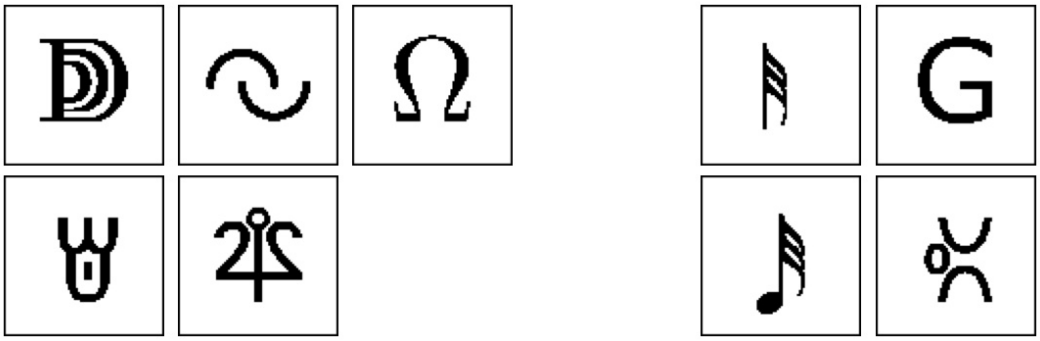
Problem 64. Progression in number of arrows (1, 2, 3). Answer=4.



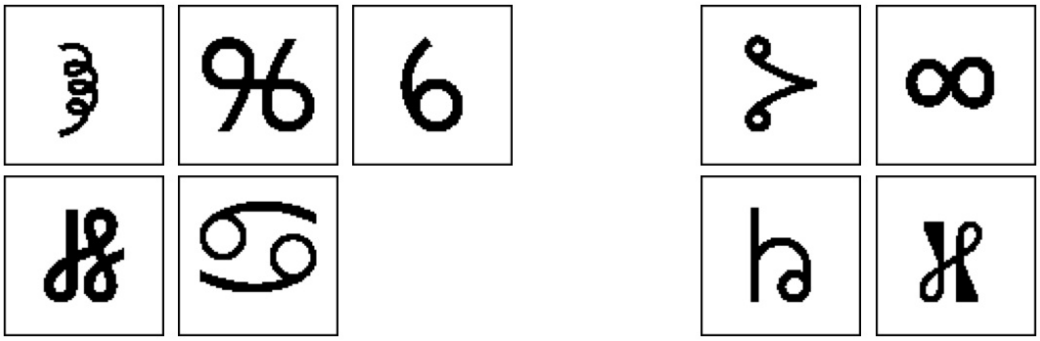
Problem 65. Progression in number of dashes (1, 2, 3). Answer=3.



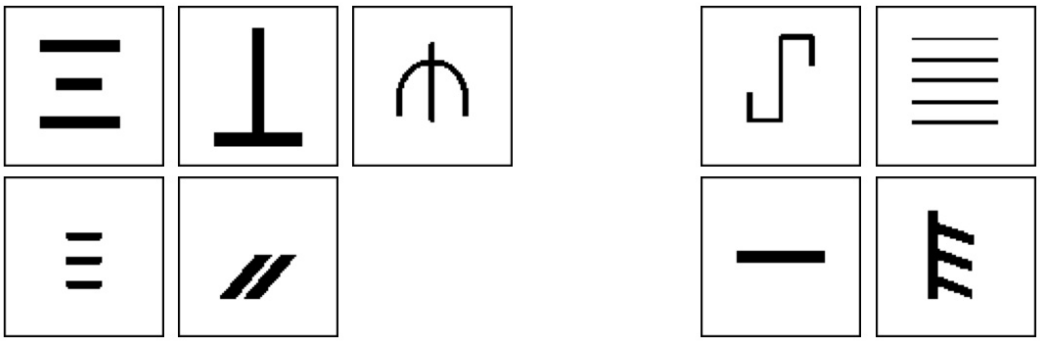
Problem 66. Progression in number of dots (5, 3, 1). Answer=3.



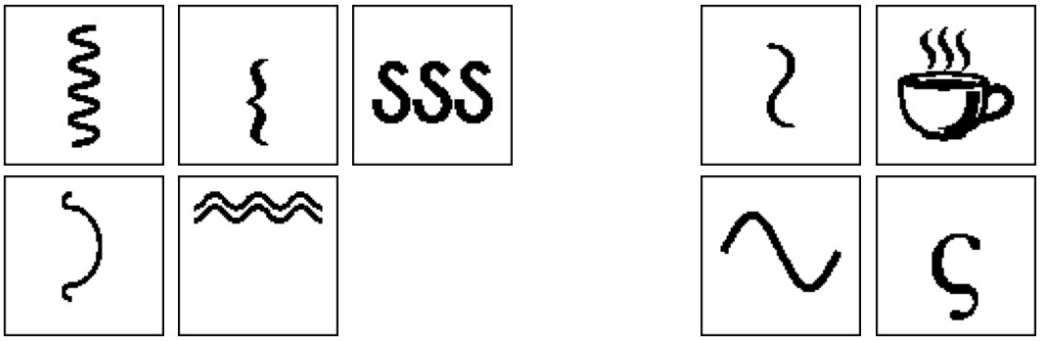
Problem 67. Progression in number of curved strokes (3, 2, 1). Answer=2.



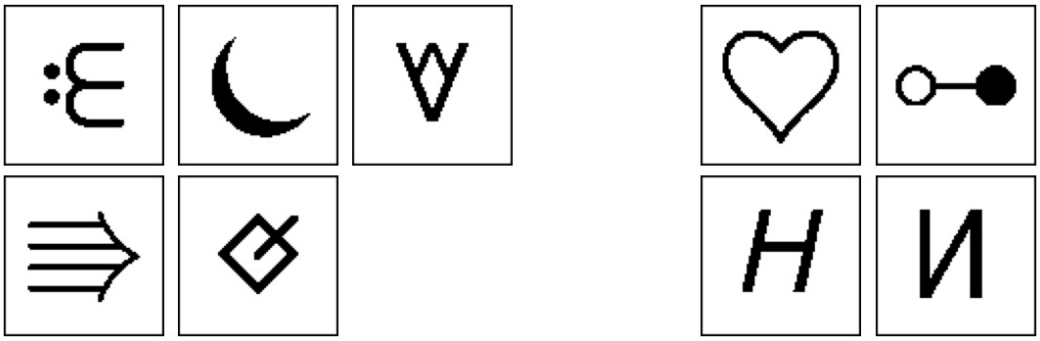
Problem 68. Progression in number of loops (3, 2, 1). Answer=3.



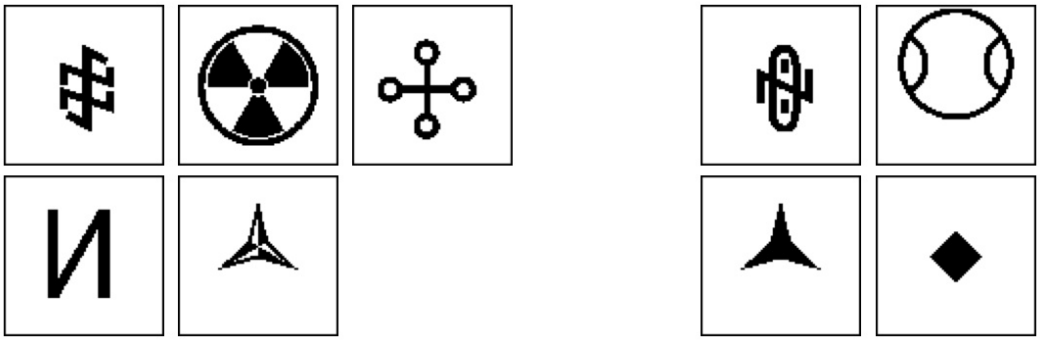
Problem 69. Progression in number of straight lines (3, 2, 1). Answer=3.



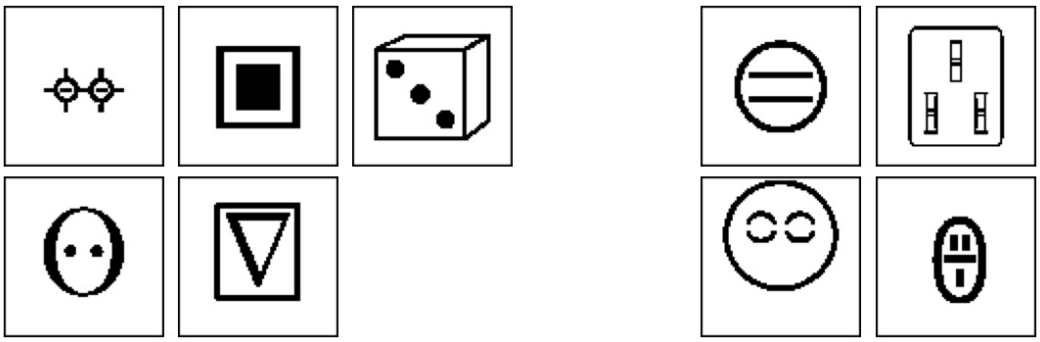
Problem 70. Progression in number of “wiggly” lines (1, 2, 3). Answer=2.



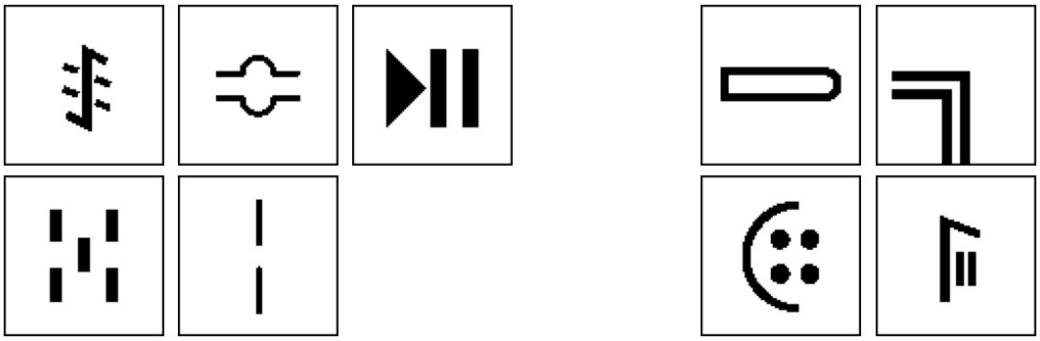
Problem 71. Progression in symmetry angle (horizontal, diagonal, vertical). Answer=1.



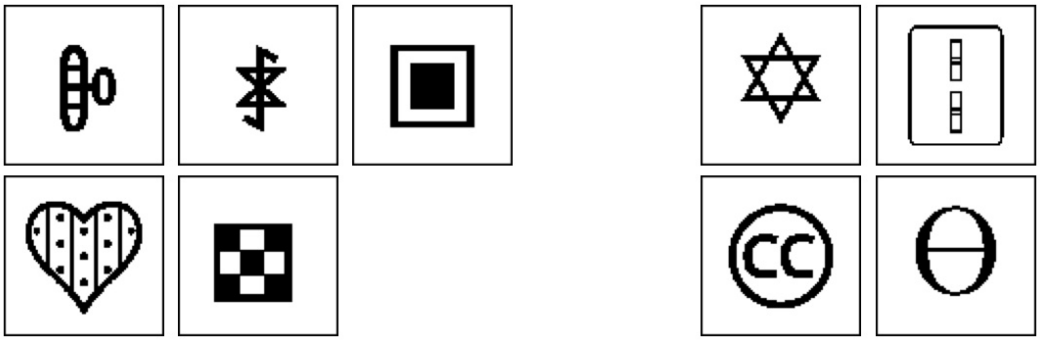
Problem 72. Progression in degree of rotational symmetry (2, 3, 4). Answer=4.



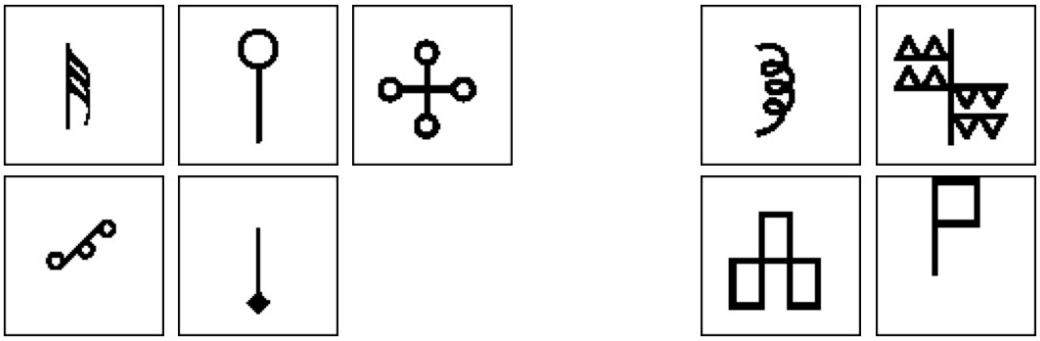
Problem 73. Arithmetic in the number of internal solid components (2 + 1 = 3). Answer=2.



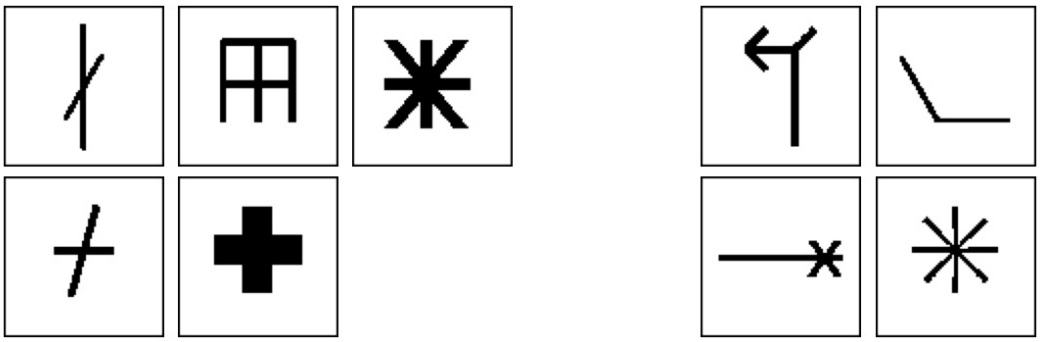
Problem 74. Arithmetic in the number of total solid components (5 - 2 = 3). Answer=4.



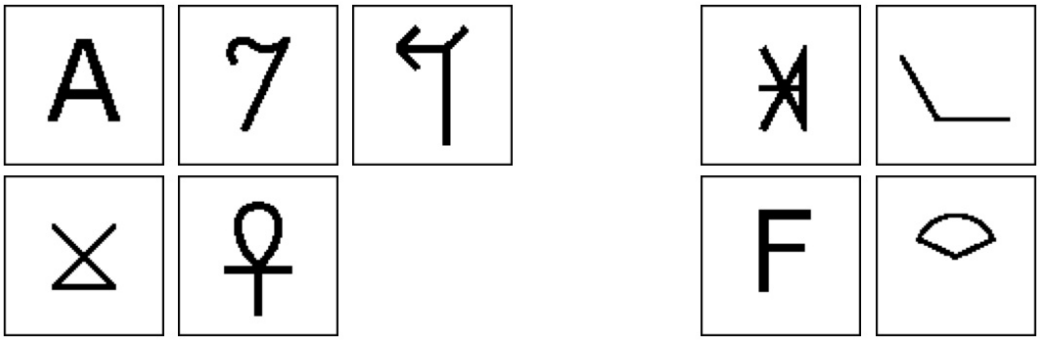
Problem 75. Arithmetic in the number of internal spaces (5 - 4 = 1). Answer=3.



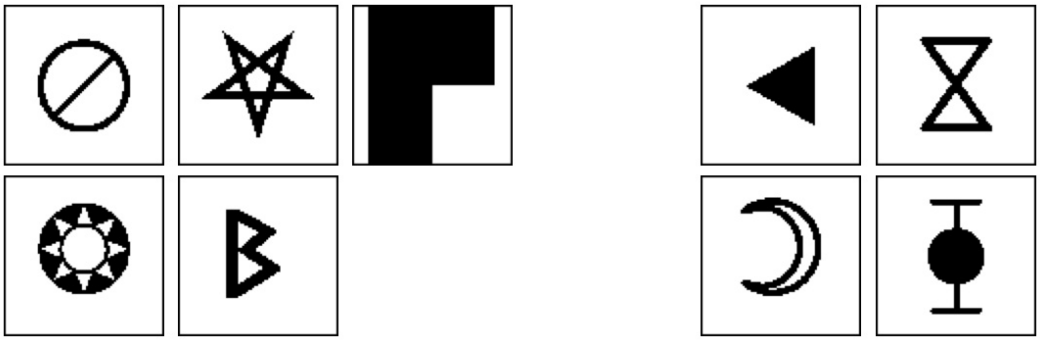
Problem 76. Arithmetic in the number of connected elements (3 + 1 = 4). Answer=2.



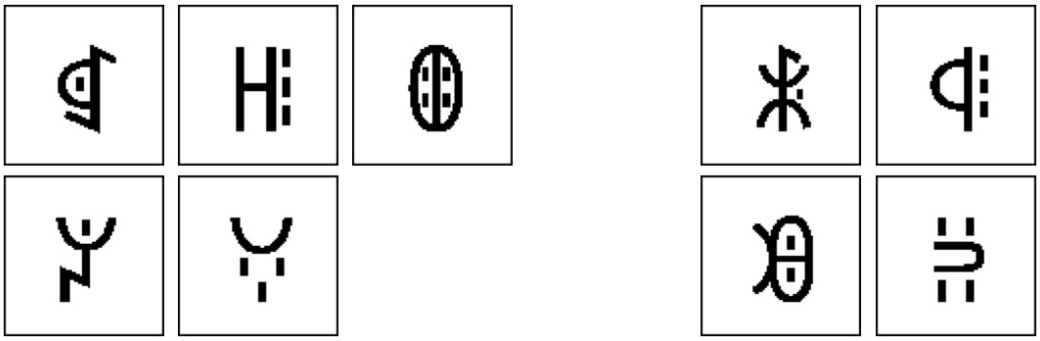
Problem 77. Arithmetic in the number of emanating lines in intersections (4 + 4 = 8). Answer=4.



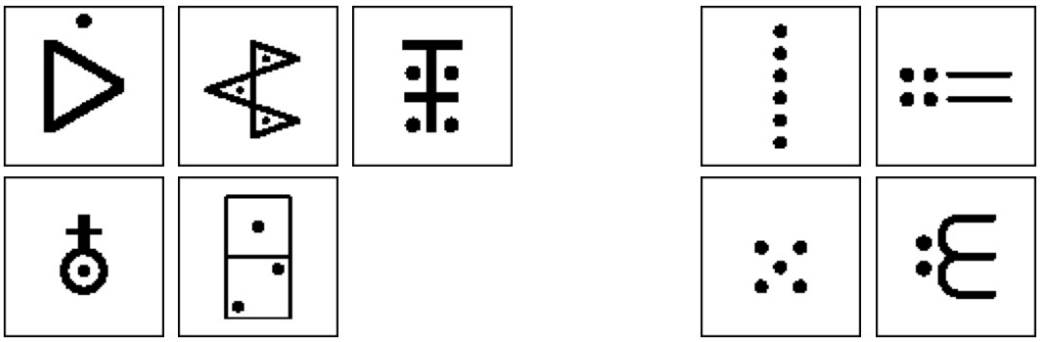
Problem 78. Arithmetic in the number of intersections (3 - 1 = 2). Answer=3.



Problem 79. Arithmetic in the number of shape sides (1 + 5 = 6). Answer=2.

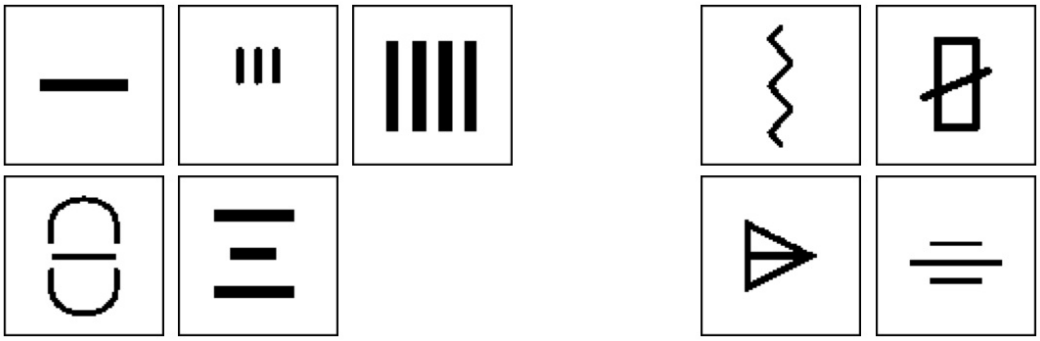


Problem 80. Arithmetic in the number of dashes (1 + 3 = 4). Answer=4.

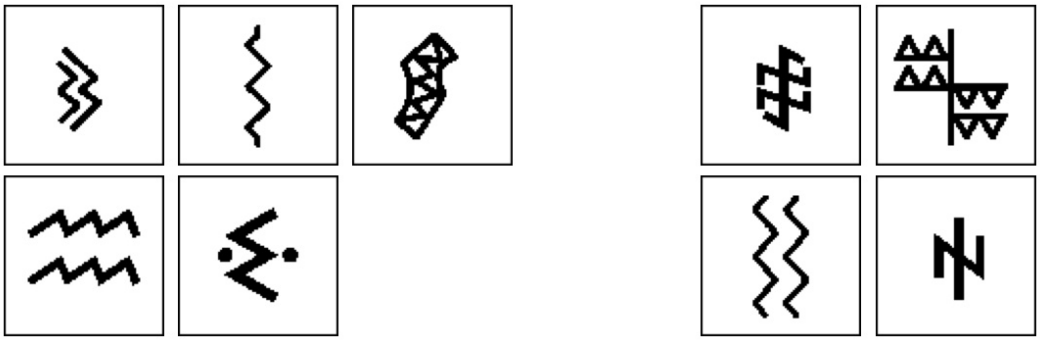


Problem 81. Arithmetic in the number of dots (1 + 3 = 4). Answer=2.

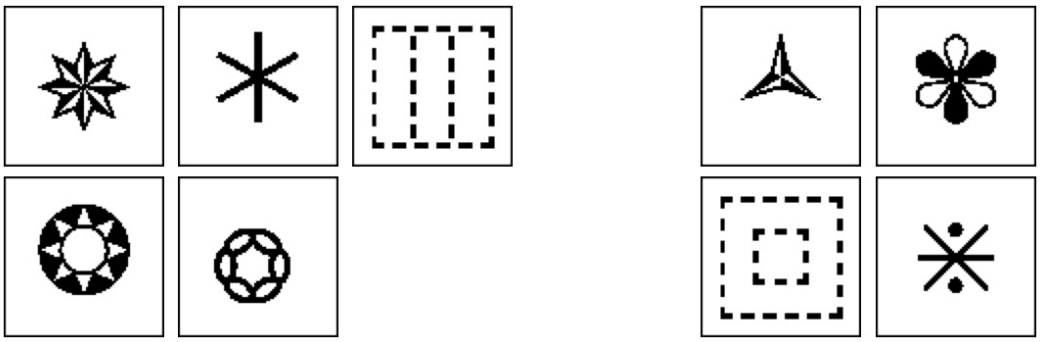
Problem 82. Arithmetic in the number of loops (1 + 1 = 2). Answer=3.



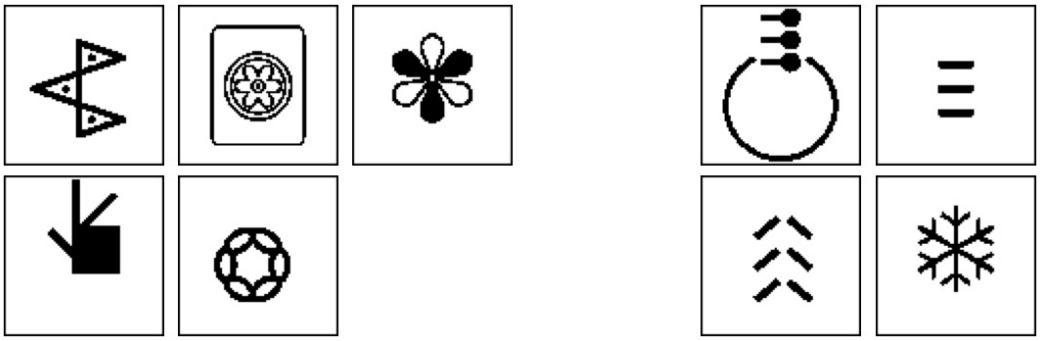
Problem 83. Arithmetic in the number of straight lines (1 + 3 = 4). Answer=3.



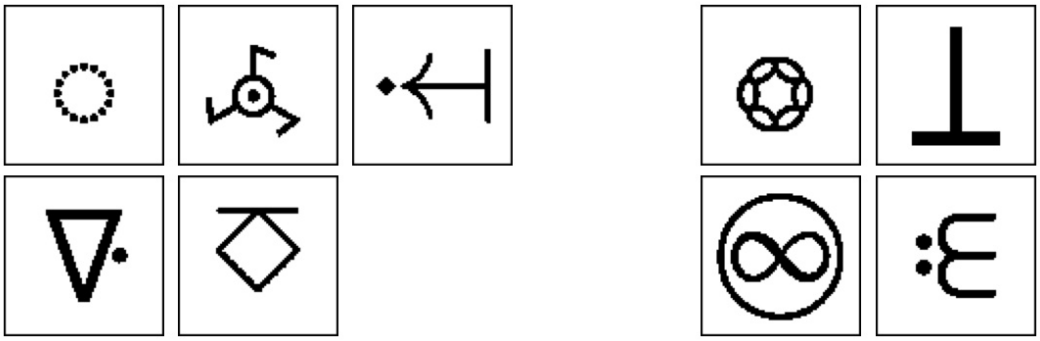
Problem 84. Arithmetic in the number of zig-zag lines (2 - 1 = 1). Answer=4.



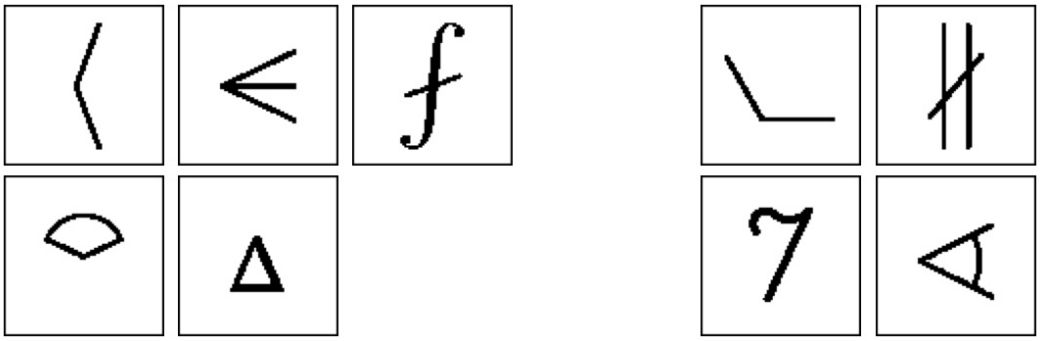
Problem 85. Arithmetic in the degree of rotational symmetry (8 - 6 = 2). Answer=4.



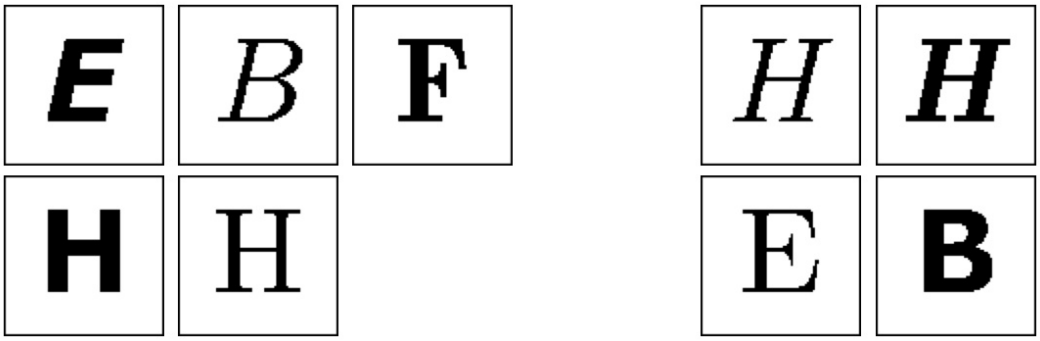
Problem 86. Union in gestalt number (3, then 6, then both 3 and 6). Answer=3.



Problem 87. Union in interaction type (none, then touching, then both none and touching). Answer=4.

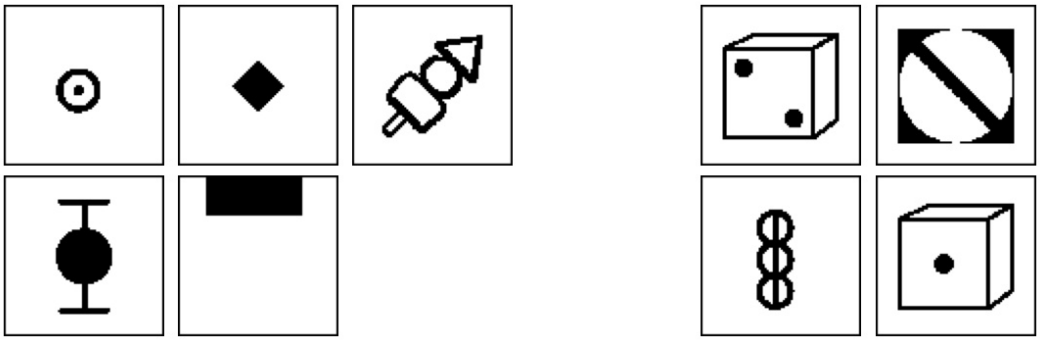


Problem 88. Union in intersection angle (obtuse, then acute, then both obtuse and acute). Answer=2.

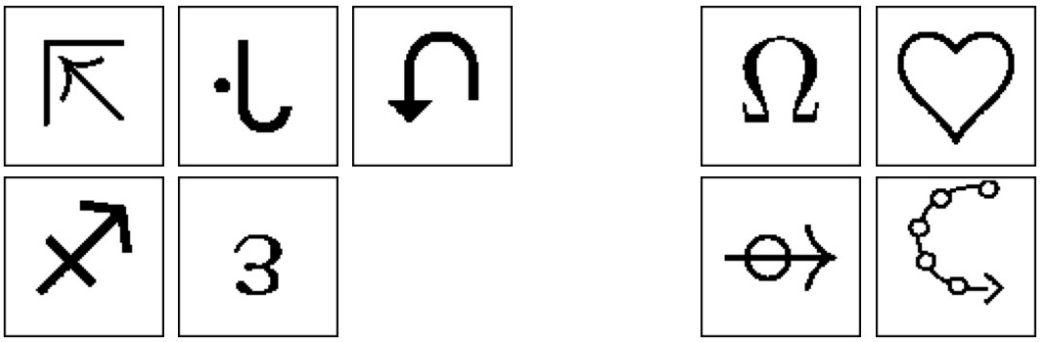


Problem 89. Union in character style (bold, then serif, then both bold and serif). Answer=2.

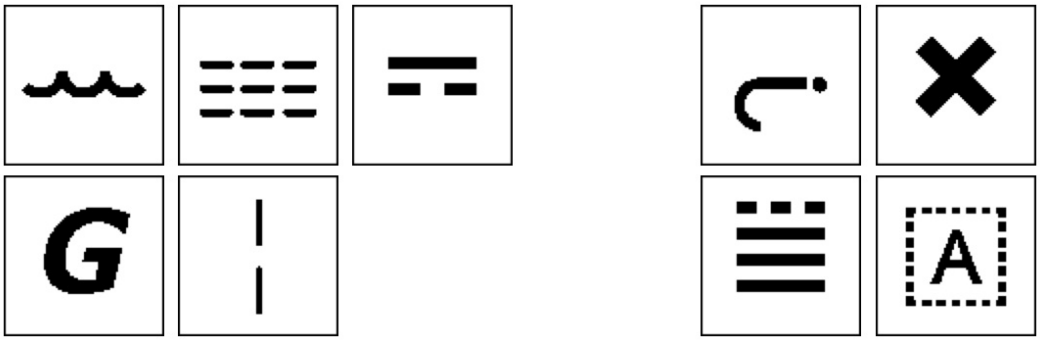
Problem 90. Union in relational position (horizontal, then vertical, then both horizontal and vertical). Answer=2.



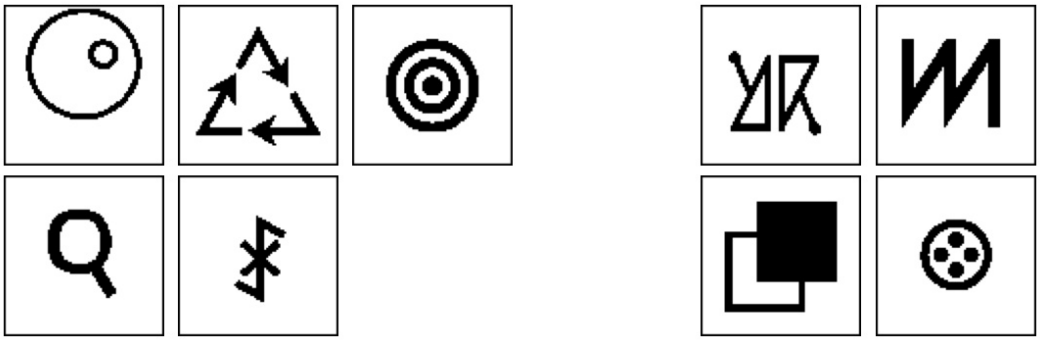
Problem 91. Union in shape type (one-sided, then four-sided, then both one and four-sided). Answer=2.



Problem 92. Union in line stroke feature (arrow, then curve, then curved arrow). Answer=4.



Problem 93. Union in line stroke style (bold, then dashed, then bold-dashed). Answer=3.



Problem 94. Union in symmetry angle (diagonal, then rotational, then both diagonal and rotational lines). Answer=4.