

Supplementary Material for “False Proximity: Frozen Code Embeddings Invert Similarity on Behaviourally Divergent Mutant Pairs”

Keletso J. Letsholo¹, Nourchène Elleuch Ben Ayed^{1*},
Achraf Ghorbel², Muath Alrammal³

¹Computer Information Science, Higher Colleges of Technology, Abu
Dhabi, UAE.

²Computer Information Science, Higher Colleges of Technology,
Sharjah, UAE.

³School of Computer Science, University of Wollongong in Dubai,
Dubai, UAE.

*Corresponding author(s). E-mail(s): nbenayed@hct.ac.ae;
Contributing authors: kletsholo@hct.ac.ae; aghorbel@hct.ac.ae;
MuathAlrammal@uowdubai.ac.ae;

This document is the Supplementary Information for the manuscript “False Proximity: Frozen Code Embeddings Invert Similarity on Behaviourally Divergent Mutant Pairs”, submitted to *Automated Software Engineering*. It collects supporting tables and reference material for the main paper. Section and table numbers carry an “S” prefix and are cited from the main text as “Supplementary Section S_n ” and “Table S_n ”. Acronyms follow the main text: AUC (area under the ROC curve), FHSR (false-high-similarity rate), JSD (Jensen–Shannon divergence), IRR (inter-rater reliability), AST (abstract syntax tree), BPE (byte-pair encoding); the mutation categories are COD (conditional negation), ROR (relational-operator replacement), and AI (algorithmic inversion).

Contents

S1 Pairwise Configuration Comparisons (RQ1)	2
---	-------------------

S2 FHSR Percentile Sensitivity	2
S3 Oracle-Verified-Subset Re-run (RQ1)	3
S4 Token-Overlap-Matched Control (Full Detail)	3
S5 RQ3 Validation-Set α Sweep	5
S6 RQ2 Overlap Distributions by Category	6
S7 Inter-Rater Reliability: Full Detail	7
S8 Mutant Candidate Behavioural Classification	7
S9 Excluded-Item Lists	8
S9.1 Problems with no retained (killed) mutant	8
S9.2 Mutant bases with no verified anchored partner	8
S9.3 Difficulty profile of the excluded anchor bases	9
S10 Oracle Generation	9
S11 Implementation and Reproducibility	9
S12 Released-Data Dictionary	10
S13 Worked Example Pairs	10
S13.1 COD mutant (p02687).	10
S13.2 ROR mutant (p02416)	10
S13.2.1 AI mutant (p02952)	11
S13.2.2 Equivalent pair (p03473)	12

S1 Pairwise Configuration Comparisons (RQ1)

Table S1 reports all $\binom{8}{2} = 28$ pairwise comparisons of the embedding configurations referenced in Section 4.1 of the main text. For each pair we give the paired-bootstrap AUC difference ($B = 10,000$) with its Bonferroni-adjusted 95% interval (corrected level $\alpha' = 0.05/28 \approx 0.0018$), and the two-tailed paired Wilcoxon signed-rank test on the per-pair scores with its matched-pairs rank-biserial effect size. A checkmark in the AUC column marks an interval excluding zero; a checkmark in the Wilcoxon column marks $p < \alpha'$. Configuration codes: CB = CodeBERT, GCB = GraphCodeBERT, UX = UniXcoder; the trailing number is the layer.

S2 FHSR Percentile Sensitivity

The main text reports FHSR at the 75th-percentile threshold and summarises the 50th- and 90th-percentile sensitivity as ranges. Table S2 gives the full per-configuration values; the 75th-percentile column reproduces Table 3 of the main text.

Table S1 All 28 pairwise configuration comparisons (RQ1). ΔAUC is $\text{AUC}(\text{a}) - \text{AUC}(\text{b})$. Rank-biserial values near ± 1 reflect the near-degenerate, almost-always-one-directional score differences between the heavily anisotropic CodeBERT CLS configurations.

Config a	Config b	ΔAUC	95% CI	Sig.	Wilcoxon p	r_{rb} (sig.)
CB-CLS-6	CB-Mean-6	-0.004	[-0.014, 0.007]	–	<0.001	+1.000 ✓
CB-CLS-6	CB-CLS-9	+0.017	[0.007, 0.027]	✓	<0.001	-1.000 ✓
CB-CLS-6	CB-Mean-9	+0.003	[-0.007, 0.012]	–	<0.001	+1.000 ✓
CB-CLS-6	CB-CLS-12	-0.036	[-0.051, -0.023]	✓	<0.001	+1.000 ✓
CB-CLS-6	CB-Mean-12	-0.003	[-0.013, 0.008]	–	<0.001	+1.000 ✓
CB-CLS-6	GCB	+0.017	[0.007, 0.028]	✓	<0.001	+1.000 ✓
CB-CLS-6	UX	+0.037	[0.027, 0.049]	✓	<0.001	+1.000 ✓
CB-Mean-6	CB-CLS-9	+0.021	[0.012, 0.030]	✓	<0.001	-1.000 ✓
CB-Mean-6	CB-Mean-9	+0.006	[0.004, 0.009]	✓	0.355	+0.021 –
CB-Mean-6	CB-CLS-12	-0.032	[-0.046, -0.019]	✓	<0.001	-0.495 ✓
CB-Mean-6	CB-Mean-12	+0.001	[-0.004, 0.006]	–	<0.001	-0.252 ✓
CB-Mean-6	GCB	+0.021	[0.012, 0.031]	✓	<0.001	+1.000 ✓
CB-Mean-6	UX	+0.041	[0.030, 0.052]	✓	<0.001	+0.998 ✓
CB-CLS-9	CB-Mean-9	-0.014	[-0.023, -0.007]	✓	<0.001	+1.000 ✓
CB-CLS-9	CB-CLS-12	-0.053	[-0.065, -0.041]	✓	<0.001	+1.000 ✓
CB-CLS-9	CB-Mean-12	-0.019	[-0.029, -0.012]	✓	<0.001	+1.000 ✓
CB-CLS-9	GCB	+0.001	[-0.008, 0.009]	–	<0.001	+1.000 ✓
CB-CLS-9	UX	+0.020	[0.013, 0.029]	✓	<0.001	+1.000 ✓
CB-Mean-9	CB-CLS-12	-0.039	[-0.052, -0.026]	✓	<0.001	-0.514 ✓
CB-Mean-9	CB-Mean-12	-0.005	[-0.010, -0.001]	✓	<0.001	-0.273 ✓
CB-Mean-9	GCB	+0.015	[0.006, 0.023]	✓	<0.001	+1.000 ✓
CB-Mean-9	UX	+0.034	[0.025, 0.045]	✓	<0.001	+0.998 ✓
CB-CLS-12	CB-Mean-12	+0.034	[0.022, 0.046]	✓	<0.001	+0.587 ✓
CB-CLS-12	GCB	+0.054	[0.040, 0.068]	✓	<0.001	+1.000 ✓
CB-CLS-12	UX	+0.073	[0.057, 0.089]	✓	<0.001	+1.000 ✓
CB-Mean-12	GCB	+0.020	[0.011, 0.029]	✓	<0.001	+1.000 ✓
CB-Mean-12	UX	+0.040	[0.028, 0.052]	✓	<0.001	+0.999 ✓
GCB	UX	+0.020	[0.012, 0.028]	✓	<0.001	+0.654 ✓

Across the eight configurations FHSR spans 0.956–0.992 at the 50th percentile and 0.769–0.957 at the 90th, so the failure is not an artefact of the 75th-percentile threshold choice.

S3 Oracle-Verified-Subset Re-run (RQ1)

As a construct-validity check, the equivalent reference distribution is restricted to the 728 oracle-verified analysis-set pairs (`equiv_verification_status = both_pass_verified_samples`); the mutant set (1,833 pairs) is unchanged. Table S3 reports per-configuration AUC and FHSR against the full 784-pair analysis set. Every configuration moves by at most 0.001 in AUC and 0.005 in FHSR, so the agreement-verified equivalent pairs do not drive the result.

S4 Token-Overlap-Matched Control (Full Detail)

The token-overlap-matched control equalises the unigram-Jaccard distribution across the equivalent and mutant classes by coarsened exact matching, then recomputes

Table S2 Per-configuration FHSR at the 50th, 75th, and 90th percentiles of the equivalent-pair score distribution. The 75th column reproduces Table 3 of the main text.

Model	Config	FHSR@50	FHSR@75	FHSR@90
CodeBERT	CLS, L6	0.967	0.927	0.881
	CLS, L9	0.983	0.957	0.899
	CLS, L12	0.956	0.870	0.769
	Mean, L6	0.959	0.915	0.866
	Mean, L9	0.964	0.931	0.885
	Mean, L12	0.966	0.919	0.867
GraphCodeBERT	CLS, L12	0.975	0.948	0.919
UniXcoder	CLS, L12	0.992	0.978	0.957

Table S3 RQ1 discrimination using only the 728 oracle-verified analysis-set equivalent pairs as the reference, versus the full 784-pair analysis set. The mutant set is identical in both.

Model	Config	AUC (728)	AUC (784)	FHSR (728)	FHSR (784)
CodeBERT	CLS, L6	0.055	0.054	0.927	0.927
	CLS, L9	0.037	0.037	0.962	0.957
	CLS, L12	0.090	0.090	0.871	0.870
	Mean, L6	0.058	0.058	0.915	0.915
	Mean, L9	0.051	0.051	0.929	0.931
	Mean, L12	0.057	0.057	0.919	0.919
GraphCodeBERT	CLS, L12	0.037	0.037	0.948	0.948
UniXcoder	CLS, L12	0.017	0.017	0.978	0.978

discrimination on the matched subset under the best configuration. The main text reports the headline: matched AUC 0.352 over $n = 390$ pairs at common support (95% bootstrap CI $[0.300, 0.409]$, $B = 10,000$ resamples of the matched pairs; the upper bound excludes 0.5 and no resample reached it), still far below 0.5. Table S4 gives the deterministic per-band composition (unigram-Jaccard bands of width 0.1); within each band the matched count is the smaller of the two class counts, giving 195 pairs per class.

The matched mass sits in the 0.3–0.8 overlap range where both classes have support: the equivalent class is absent above 0.8 and the mutant class below 0.2. The six common-support bands (those with matched pairs) are $[0.2, 0.3)$, $[0.3, 0.4)$, $[0.4, 0.5)$, $[0.5, 0.6)$, $[0.6, 0.7)$, and $[0.7, 0.8)$, contributing 5, 18, 72, 72, 19, and 9 matched pairs per class respectively (Table S4). Even restricted to this overlap region, the best configuration scores mutants above equivalents (matched AUC 0.352); re-drawing the subsample within each bootstrap resample yields a near-identical interval, $[0.312, 0.414]$, so the conclusion does not depend on the single matching draw.

Table S4 Coarsened-exact-matching bands for the token-overlap-matched RQ1 comparison. “Matched” is $\min(n_{\text{equiv}}, n_{\text{mutant}})$ per band; the matched subset holds 195 equivalent and 195 mutant pairs (390 total). Matched-subset AUC under the best configuration is 0.352 (95% bootstrap CI [0.300, 0.409], $B = 10,000$).

Unigram-Jaccard band	n equiv	n mutant	n matched
[0.0, 0.1)	4	0	0
[0.1, 0.2)	58	0	0
[0.2, 0.3)	219	5	5
[0.3, 0.4)	232	18	18
[0.4, 0.5)	171	72	72
[0.5, 0.6)	72	126	72
[0.6, 0.7)	19	242	19
[0.7, 0.8)	9	347	9
[0.8, 0.9)	0	526	0
[0.9, 1.0]	0	497	0
Total	784	1833	195

S5 RQ3 Validation-Set α Sweep

The α weight for HybridSim is selected on the 20% validation split ($n = 523$: 156 equivalent, 367 mutant) and fixed before any test-set evaluation. Table S5 reports validation AUC for every model and α on the canonical seed-42 split. Cosine alone reaches 0.079; every structural-inclusive variant collapses to at most 1.6×10^{-4} (HybridSim and ASTOnly) or about 0.016 (JaccardPenalty), all far below both the cosine baseline and the 0.5 chance level. The three HybridSim settings are indistinguishable at this scale, so α carries no usable signal; we fix $\alpha = 0.7$. The choice does not affect any conclusion: on the test set all three α give AUC 0.002 (main-text Table 9).

Table S5 Validation-set ($n = 523$) AUC by model and α , canonical seed-42 split. Cosine ($\alpha = 1$) and ASTOnly ($\alpha = 0$) are the endpoints; HybridSim and JaccardPenalty interpolate between them. All structural-inclusive settings lie far below the 0.5 chance level.

Model	$\alpha = 0.3$	$\alpha = 0.5$	$\alpha = 0.7$	Endpoint
HybridSim	0.000	0.000	0.000	–
JaccardPenalty	0.016	0.016	0.016	–
Cosine ($\alpha = 1$)	–	–	–	0.079
ASTOnly ($\alpha = 0$)	–	–	–	0.000

Note. HybridSim and ASTOnly validation AUC are at the 10^{-4} level (HybridSim 0.000157, 0.000140, 0.000140 at $\alpha = 0.3, 0.5, 0.7$; ASTOnly 0.000140) and round to 0.000 at three decimals; JaccardPenalty is 0.016 at all three α . Cosine validation AUC is 0.079.

S6 RQ2 Overlap Distributions by Category

The main text reports category means for token and structural overlap. Table S6 gives the full per-category quartiles of the two per-pair features in the released feature table: unigram token Jaccard and the normalised AST edit distance d_{AST} used in RQ3. The depth-3 AST-overlap feature analysed in RQ2 is released with the RQ2 feature table in the repository.

Table S6 Per-category quartiles of unigram token Jaccard and normalised AST edit distance d_{AST} over mutant pairs. The tight, near-zero d_{AST} quartiles confirm that single-site mutants are structural near-twins of their originals in every category.

Measure	Category	Min	Q1	Median	Q3	Max
Token Jaccard	COD	0.286	0.706	0.829	0.914	1.000
	ROR	0.286	0.704	0.818	0.900	1.000
	AI	0.317	0.675	0.811	0.904	1.000
d_{AST}	COD	0.002	0.006	0.010	0.016	0.091
	ROR	0.001	0.003	0.005	0.010	0.071
	AI	0.000	0.002	0.005	0.009	0.058

Table S7 reports the logistic regression of the high-confusion label on token overlap summarised in the main text (RQ2). Because the two predictors are strongly collinear ($r = 0.94$), the coefficients are read jointly rather than individually.

Table S7 Logistic regression of the high-confusion label on token overlap (mutant pairs, best RQ1 configuration: CodeBERT CLS, layer 12). The model is fit on 80% of the pairs and the AUC is computed on the held-out 20%; the McFadden pseudo- R^2 is reported on the training fit. The predictors are strongly collinear ($r = 0.94$), so the coefficients should be read jointly rather than individually.

Quantity	Value
Coefficient, J_{unigram}	7.009
Coefficient, J_{bigram}	-0.477
Intercept	-2.975
McFadden pseudo- R^2 (train)	0.149
Held-out AUC (20%)	0.768
High-confusion prevalence	0.870
n train / test	1466/367

S7 Inter-Rater Reliability: Full Detail

The two annotators labelled the same 263-pair stratified sample (80 equivalent, 183 mutant: COD 77, ROR 78, AI 28) using the four-label scheme (**equivalent**, **killed**, **neither**, **unclear/undetermined**). Pooled raw agreement is 0.9658 (254/263), chance agreement 0.5440, Cohen’s $\kappa = 0.9250$ (bootstrap 95% CI [0.8765, 0.9676]). Table S8 gives the full rater-by-rater confusion matrix; Table S9 gives per-rater accuracy against the gold (pipeline) labels, by gold label and by category.

The sample is stratified by pair type (10% of each, largest-remainder rounding) and, within each category, by transformation site (10% per site, minimum one, offset against the largest site): for equivalent pairs, 70 oracle-verified, 9 agreement-verified, and 1 interactive-verified; for COD, 76 condition→**not** condition and 1 **not** condition→condition; for ROR, 55 ==→!=, 6 <→>=, 6 >→<=, 4 <=→>, 4 >=→<, and 3 !=→==; for AI, 16 arithmetic complement, 10 **min**/**max**, 1 sort-direction, and 1 slice reversal. Annotation files are blinded: each annotator received the two programs and the oracle evidence with no gold label, category, mutation site, or program-role information, and program order was randomised and recorded in a separate key. The nine disagreements (seven **killed**-versus-**neither**, two **neither**-versus-**equivalent**) were jointly adjudicated against the oracle-scoped protocol; under the supplied evidence all nine match the pipeline label, so no label changed.

Table S8 Rater 1 (rows) by Rater 2 (columns) confusion matrix over the 263 double-rated pairs. Off-diagonal cells are the nine disagreements.

Rater 1	Rater 2			
	equivalent	killed	neither	unclear
equivalent	77	0	0	0
killed	0	174	7	0
neither	2	0	3	0
unclear	0	0	0	0

All nine disagreements fall in two cells: seven **killed**-versus-**neither** (in the mutant strata) and two **neither**-versus-**equivalent** (in the equivalent stratum). The two annotators jointly adjudicated these items against the annotation protocol, which scopes judgements to the supplied oracle evidence; under that evidence all nine match the pipeline-assigned label, so no label changed.

The per-item disagreement list (pair identifier, both rater labels, gold label, and adjudication note) is released with the IRR annotation materials in the repository.

S8 Mutant Candidate Behavioural Classification

Before killed-only retention, every generated mutant candidate is classified by program-to-program comparison against its original (Section 3.2.4 of the main text).

Table S9 Per-rater accuracy against the gold (pipeline) labels, overall and broken down by gold label and by category. “Both” is the fraction of items on which both raters match the gold label.

Stratum	n	Rater 1	Rater 2	Both
Overall	263	0.981	0.962	0.954
Gold = equivalent	80	0.963	0.988	–
Gold = killed	183	0.989	0.951	–
COD	77	0.974	0.948	–
ROR	78	1.000	0.974	–
AI	28	1.000	0.893	–
Equivalent	80	0.963	0.988	–

Table S10 gives the full behavioural-label distribution of the 2,282 classified candidates by category: **killed** candidates are retained, **neither** and **equivalent** are excluded, and no candidate was **undetermined**.

Table S10 Behavioural classification of all 2,282 classified mutant candidates by category, before killed-only retention.

Category	killed (retained)	neither	equivalent	Total
COD	779	20	0	799
ROR	775	24	0	799
AI	279	378	27	684
Total	1,833	422	27	2,282

S9 Excluded-Item Lists

S9.1 Problems with no retained (killed) mutant

Sixteen of the 800 problems contribute no retained mutant pair and are excluded from the analysis set. Fifteen produced classified candidates that never met the killed-on-every-usable-input criterion: p02641, p02664, p02797, p02891, p02983, p02985, p03007, p03073, p03108, p03157, p03252, p03338, p03416, p03583, p03821. The sixteenth, p03046, produced no classified candidate in any category.

S9.2 Mutant bases with no verified anchored partner

Of the 1,118 distinct killed-mutant bases, 20 yielded no equivalent partner that agrees with the base on every usable oracle input and are excluded from the anchored robustness set (Section 3.3.2 of the main text). Listed as problem / base file: p00005/s213956671.py, p02383/s200276064.py, p02618/s083067666.py,

p02618/s253942580.py, p02632/s013569109.py, p02690/s071654179.py,
p02924/s005934625.py, p03026/s005481615.py, p03026/s012333299.py,
p03033/s031058671.py, p03138/s011353755.py, p03340/s003926577.py,
p03402/s029472437.py, p03496/s001461177.py, p03545/s005060674.py,
p03593/s033684239.py, p03634/s019648985.py, p03634/s023402818.py,
p03760/s020539973.py, p03796/s067285721.py.

S9.3 Difficulty profile of the excluded anchor bases

Table S11 compares the 20 excluded bases against the 1,098 retained bases on two size proxies. The excluded bases are longer on both measures; the difference is significant by a two-sided Mann–Whitney test. Because these bases’ mutant pairs remain in the primary analysis set, this skew is confined to the anchored robustness check.

Table S11 Size profile of excluded versus retained anchored-mutant bases. *p*-values are two-sided Mann–Whitney.

Group	<i>n</i>	Median lines	Mean lines	Median tokens	Mean tokens
Excluded	20	24	22.1	319	283.6
Retained	1098	13	15.3	156	197.7
Mann–Whitney: lines $p < 0.001$; tokens $p = 0.002$.					

S10 Oracle Generation

The per-problem oracle merges public sample I/O pairs (parsed from the AtCoder and AIZU HTML problem statements) with extended tests generated by Claude Sonnet 4.6, prompted per problem to propose boundary cases, corner cases, and representative valid inputs. Each generated input is validated by executing accepted submissions: when a reliable expected output can be established it is stored as a fixed oracle pair, and otherwise it is retained as an agreement-only case used for program-to-program comparison. The augmentation run produced 7,620 candidate inputs across the 800 problems: 7,394 validated with fixed outputs, 58 retained as agreement-only, and 168 excluded for runtime errors, compile errors, timeouts, or related validation failures, leaving 7,452 usable generated inputs.

The generation was performed with Claude Sonnet 4.6. The full generation script, the exact prompt template, the decoding configuration, and a per-problem table of fixed-output, agreement-only, and excluded input counts are released with the analysis code in the repository.

S11 Implementation and Reproducibility

All code, the full dataset, mutant-generation scripts, annotation materials, and evaluation notebooks are available at submission from <https://github.com/letsholok/>

[codebert-mutant-similarity](#). Sampling, equivalent- and mutant-pair selection, and the merged oracle of public samples plus Claude-generated extended tests are described in the dataset-construction methodology (Section 3.2 of the main paper); the per-problem oracle counts and full generation provenance are given in the Oracle Generation section above. Mutations are applied with Python’s `ast` module (standard library); submissions using Python 3.9+ constructs (walrus operator, `match`, union-type hints) are excluded before mutation, and each program is executed in a fresh subprocess. AST parsing uses `tree-sitter` Python bindings (v0.20.1) and tree-edit distance uses the `apted` library (v1.0.3); CodeBERT, GraphCodeBERT, and UniXcoder embeddings are extracted with `transformers` v4.35.0 and PyTorch 2.1.0. Embedding extraction runs on a single NVIDIA A100-40GB GPU and AST-distance computation is CPU-bound. The RQ3 20/80 validation/test split uses stratified sampling over the joint (category, label) strata with seed 42.

S12 Released-Data Dictionary

The released artefact (<https://github.com/letsholok/codebert-mutant-similarity>) contains three primary CSV files. Tables S12–S14 document their columns.

S13 Worked Example Pairs

Table S15 gives four worked examples under the best configuration (CodeBERT CLS, layer 12). The three single-site mutants score near 1.0 with near-zero AST distance despite diverging in behaviour, while the independently written equivalent pair scores *lower* on cosine and far higher on d_{AST} : the inversion in miniature.

S13.1 COD mutant (p02687).

The original prints `ARC` when the input equals `ABC` and `ABC` otherwise; the mutation negates the condition (`if cont == 'ABC'` becomes `if not (cont == 'ABC')`), swapping the two outputs on every input. Cosine = 0.999, d_{AST} = 0.022.

```
cont = input()
if cont == 'ABC':
    print('ARC')
else:
    print('ABC')
```

S13.2 ROR mutant (p02416)

The loop terminates when the line equals `'0'`; the mutation replaces `==` with `!=` at the guard, inverting the termination condition. Cosine = 0.998, d_{AST} = 0.009.

```
while True:
    n = input()
    if n == '0':
        break
    print(sum([int(x) for x in list(n)]))
```

Table S12 Columns of the killed-and-classified mutant-pair file (`all_mutant_pairs.csv`).

Column	Description
<code>pair_id</code>	Unique identifier of the mutant pair.
<code>problem_id</code>	Project.CodeNet.Python800 problem identifier.
<code>label</code>	Short behavioural label of the retained pair (<code>killed</code>); the oracle-qualified form is in <code>classification_status</code> .
<code>category</code>	Mutation category: COD, ROR, or AI.
<code>program_A.file</code> , <code>program_B.file</code>	Filenames of the original (<code>program_A</code>) and its single-site mutant (<code>program_B</code>); the mutant filename is the base name with a category-tagged <code>mutant</code> suffix.
<code>program_A.src</code> , <code>program_B.src</code>	Source text of the two programs.
<code>mutation_operator</code>	The operator applied to produce the mutant.
<code>mutation_site</code>	Location of the single mutated site.
<code>original_token</code> , <code>mutated_token</code>	The token at the mutation site before and after mutation.
<code>mutation_template</code>	The inversion template / operator pattern applied.
<code>io_verified</code>	Whether the pair passed I/O verification.
<code>n_io_pairs_tested</code>	Number of oracle input/output pairs tested.
<code>n_usable</code>	Number of usable oracle inputs (after excluding execution failures and timeouts).
<code>n_agree</code>	Number of usable inputs on which the two programs produced identical output.
<code>n_diverge</code>	Number of usable inputs on which the outputs diverged.
<code>n_original_errors</code> , <code>n_mutant_errors</code>	Number of inputs on which the original / mutant raised an execution error.
<code>oracle_io_source</code>	Provenance of the oracle inputs used.
<code>classification_status</code>	Behavioural label: <code>killed_on_oracle</code> , <code>partial_agreement_on_oracle</code> , or <code>equivalent_on_oracle</code> .
<code>sample_inputs.json</code>	JSON list of the oracle inputs used.
<code>sample_expected_outputs.json</code>	JSON list of expected outputs, where a fixed expected output was established.
<code>program_A.outputs.json</code> , <code>program_B.outputs.json</code>	JSON list of each program's output per oracle input.
<code>program_A.statuses.json</code> , <code>program_B.statuses.json</code>	JSON list of each program's per-input execution status.
<code>diverging_sample_indices.json</code>	Indices of oracle inputs on which the outputs diverged.
<code>agreeing_sample_indices.json</code>	Indices of oracle inputs on which the outputs agreed.

S13.2.1 AI mutant (p02952)

The original counts qualifying values; the mutation replaces `+=` with `-=`, negating the running count. Cosine = 0.999, d_{AST} = 0.005.

```

N = int(input())
ans = 0
for i in range(1,N+1):
    if len(str(i)) == 1 or len(str(i)) == 3 or len(str(i)) == 5:
        ans +=1
print(ans)

```

Table S13 Columns of the anchored equivalent-partner file (`anchored_equiv_pairs.csv`).

Column	Description
<code>anchor_pair_id</code>	Unique identifier of the anchored pair.
<code>problem_id</code>	Problem identifier.
<code>label</code>	Behavioural label of the base-partner pair; equivalent for every row in this file.
<code>category</code>	Constant tag anchored.equivalent ; this file holds only anchored equivalent pairs, not mutation categories.
<code>base_file, partner_file</code>	Filenames of the anchor base and its verified equivalent partner.
<code>base_src, partner_src</code>	Source text of the base and partner.
<code>program_A_file, program_B_file</code>	Pair filenames in normalised order: program_A is the base/anchor, program_B the verified equivalent partner (mirroring base_file/partner_file).
<code>program_A_src, program_B_src</code>	Source text for program_A (base) and program_B (partner).
<code>equiv_verification_status</code>	Verification status of the base-partner agreement.
<code>io_verified</code>	Whether the pair passed I/O verification.
<code>n_io_pairs_tested</code>	Number of oracle inputs tested.
<code>sample_inputs_json</code>	JSON list of oracle inputs used.
<code>sample_expected_outputs_json</code>	JSON list of expected outputs, where available.
<code>program_A.outputs_json, program_B.outputs_json</code>	JSON list of each program's output per oracle input.

S13.2.2 Equivalent pair (p03473)

Two independently written programs that both print $48 - M$. They share little surface form, so cosine is *lower* (0.986) and d_{AST} far higher (0.481) than for any of the behaviourally divergent mutants above.

```

nl = lambda: list(map(int, input().split()))
sl = lambda: input().split()
n = lambda: int(input())
s = lambda: input()

M = n()

print(48 - M)

```

```

def main():
    m = int(input())
    x = 48 - m
    print(x)

if __name__ == "__main__":
    main()

```

Table S14 Columns of the equivalent-pair file (`equiv_pairs_flagged.csv`).

Column	Description
<code>pair_id</code>	Unique identifier of the equivalent pair (e.g. <code>p00000_equivalent</code>).
<code>problem_id</code>	Problem identifier.
<code>label</code>	Behavioural label; equivalent for every row.
<code>category</code>	Constant tag equivalent .
<code>program_A_file</code> , <code>program_B_file</code>	Filenames of the two equivalent submissions.
<code>program_A_src</code> , <code>program_B_src</code>	Source text of the two submissions.
<code>equiv_verification_status</code>	<code>both_pass_verified_samples</code> , <code>both_outputs_agree</code> , or <code>both_outputs_agree_interactive</code> .
<code>equiv_combination</code>	Which eligible-pool submissions were combined to form the pair (e.g. <code>eligible[0]+eligible[1]_confirmed</code>).
<code>io_verified</code>	Whether the pair passed I/O verification (True for all 800 pairs).
<code>n_io_pairs_tested</code>	Number of oracle inputs tested.
<code>sample_inputs_json</code>	JSON list of the oracle inputs used.
<code>sample_expected_outputs_json</code>	JSON list of expected outputs, where available.
<code>program_A_outputs_json</code> , <code>program_B_outputs_json</code>	JSON list of each program’s output per oracle input.
<code>in_analysis_set</code>	Boolean flag: True for the 784 analysis-set pairs, False for the 16 released-but-excluded pairs.

Table S15 Four worked examples (best configuration). The mutants are scored more similar to their originals than the genuine equivalent pair is to its partner.

Problem	Type	Cosine	d_{AST}
p02687	COD mutant	0.999	0.022
p02416	ROR mutant	0.998	0.009
p02952	AI mutant	0.999	0.005
p03473	Equivalent	0.986	0.481