26.0 / 30.0

Signature

Author clearly identified with name and id near the top of the file.	**1.0**
Some identification, but missing name or id, or too far from the top of the file.	0.5
No author identification.	0.0

1 or 2 line summary of file purpose

Good, clear, helpful one line summary near the top of the file explaining what the program does (not just what subject or project it's for).	**1.0**
Unclear or not very helpful summary near top of file.	0.5
No brief summary near top of file.	0.0

Quality of file-level documentation

Clear, comprehensive but not overwhelming description of the purpose of the file and the approach taken, without assuming the reader knows the project.	**3.0**
Clear description of the purpose of the file and the approach taken, but incomplete or too much irrelevant information.	2.0
Brief description of the purpose of the file and the approach taken, but significant detail omitted or assumes knowledge of the project.	1.0
Little or no file-level documentation.	0.0

Quality of function/predicate/type level documentation

Outstanding documentation of every function/predicate/type explaining purpose, meaning of arguments and output, and any implementation subtleties. No unhelpful comments (eg, saying "increment i" next to a statement that increments i).	**5.0**
Excellent documentation of almost every function/predicate/type explaining purpose and meaning of arguments and output. Subtleties explained, but no unhelpful comments.	4.0
Good, clear documentation of important functions/predicates/types explaining meaning of arguments and output. No unexplained mysterious code, nor excessive comments about self-explanatory code.	3.0
Some documentation for many functions/predicates/types explaining key arguments and output, but some missing documentation or excessive comments.	2.0
Some documentation of some functions/predicates/types, but not particularly helpful or too distracting.	1.0
Little or no useful documentation of functions/predicates/types or purposes, or excessive unhelpful comments making the code harder to understand.	0.0

> Awesome! Comprehensive and clear documentation.

Readability

Beautifully readable, with neat, consistent layout. A pleasure to read.	**5.0**
Very readable and tidy, with good code layout. Easy to navigate.	4.0
Well presented, neat, mostly consistent layout.	3.0
Mostly readable, but with some messy parts. Layout inconsistent in places or some long lines.	2.0
Hard to read, with inconsistent or poor layout or many long lines.	1.0
Little evidence of effort to layout the code. Illegible.	0.0

> Outstanding formatting.

Understandability

A clear approach, well organised, with any subtleties clearly explained. Names well chosen and clear but not too long.	5.0
Cleanly organised structure, with subtleties explained. Names are understandable.	**4.0**
Functions/predicates are pretty understandable. Names are mostly clear enough. Some effort to organise the presentation.	3.0
Some of the code is unclear without explanation. Some names are a bit cryptic. Not much organisation of the code.	2.0
Much of the code and/or many of the names are hard to understand. Little effort to group related things together.	1.0
Code is completely cryptic. Names give little hint of purpose.	0.0

> Division of code into sections is excellent, headers are nice. Variable names mostly very clear.

Abstraction

Well structured code, each part with a single purpose, no code repetition, no divided responsibilities, succinct definitions. Simple, elegant code.	5.0
Well thought-out structure, no code repetition, no long definitions, simple code.	4.0
Good structure, no code repetition, reasonable length definitions, code could be a bit simpler.	**3.0**
Some apparent structure, not a lot of redundant code, but code could have been simpler with a bit more abstraction.	2.0
Not much evident structure; could could be substantially improved with some abstraction.	1.0
Spaghetti code.	0.0

> Listing all cards manually is not ideal, you should generate them from a few small predicates instead. Good job creating nice interfaces for recursive predicates with accumulators, eg `sum_to_15`. Code mostly simple and clear.

Use of Language and Libraries

Elegant use of language and library facilities; no wheels reinvented; code reflects mastery of the language and libraries.	5.0
Excellent use of language and library facilities; not much could be improved with better language or library use; code reflects good understanding of the language and libraries.	**4.0**
Program demonstrats good use of language and library facilities in many places, although some some opportunities for more sophisticated usage were missed.	3.0
Program makes good use of language and library facilities in a few places, but misses many opportunities for better usage.	2.0
Program mostly uses the most basic language and library facilities, but misses many opportunities for better usage.	1.0
Program hardly uses any language and library facilities, reflecting little understanding of the language or library.	0.0

> Cool use of aggregate_all! Overall good use of library predicates.

- ✔ hand_value for a pair
- ★ hand_value test 1
- ★ hand_value test 2
- ★ hand_value test 3
- ★ hand_value test 4
- ★ hand_value test 5
- ★ hand_value test 6
- ★ hand_value test 7
- ★ hand_value test 8
- ★ hand_value test 9
- ★ hand_value test 10

- ✔ select_hand simple test result percentage: 100
- ★ select_hand test 1 result percentage: 100
- ★ select_hand test 2 result percentage: 100
- ★ select_hand test 3 result percentage: 100
- ✘ select_hand test 4 Your submission exited with status 1 when it should have exited with 0.

```
ERROR: Prolog initialisation failed:
ERROR: format/2: Illegal argument to format sequence ~d: 87.5
```

- ★ select_hand test 5 result percentage: 100
- ★ select_hand test 6 result percentage: 100
- ★ select_hand test 7 result percentage: 100
- ★ select_hand test 8 result percentage: 100
- ★ select_hand test 9 result percentage: 100
- ★ select_hand test 10 result percentage: 100