	PC	Ptos PC	CC	Ptos CC	RACC	Ptos RACC	CACC	Ptos CACC	Comentarios Generales	Nota
GRUPO 1	Perfecto! Se tiene coverage 100% :D.	1	Se tienen demasiados tests innecesarios (igual se tiene coverage 100%). Lo que se utilizó acá como caso de test eran en realidad Requisitos para el coverage que debían ser cubiertos por casos de tests que los agruparan.	0,5	Mismo comentario que en CC sobre los TR y los TC. Faltan casos en su análisis, recuerden que esto es sobre cláusula activas: por ejemplo, falta el coverage para `p1' y 'q1` en el primer predicado -> (p1=F, q1=F y r1=T), pero en el test `test_current_hp_lower_than_50_and_healthier_than_other` se cubre. Creo que fue suerte. Se tiene, bajo sus supuestos de predicados coverage 100%.	1,5	Bien aprovechada la batería de CACC.	1,5	Se confunden entre TR y TC. Sí plantearon el elsif y else como predicados. al momento de analizar los predicados de elsif y else, no analizan por clausulas (CACC y RACC).	5,5
GRUPO 2	No se cubrió el caso del predicado del `else` sea true, ya que se planteó mal el `elsif` (ya que no se puede tener el p1 en True y el p2 en True al mismo tiempo). Sus casos debieron estar en tests separados. Otro tema es que es innecesario hacer asserts por cada predicado en la función. Bastaba con realizar el assert sobre el método `attack`. Por último, considerando que están haciendo asserts sobre los predicados, faltarían asserts para comprobar que se cumplen las condiciones puestas en los comentarios.	0,5	Perfecto! Se tiene coverage 100% :D. Ojo que declaran que en el primer test todas las cláusulas son true pero `other.faster_than?(self)` es false (y true en el segundo test así que igual se cubre).	1	Sin tests :(. Tuvieron varias horas del día para traspasarlos. En el predicado 1 falta el coverage para c3. Además, el T4 es redundante. En el predicado 2 y 3 no se trabaja correctamente sobre las cláusulas. En el predicado 2 el T5 es redundante. En el predicado 4 solo se cubre a c6. [1]	1	El coverage se debía hacer sobre todas las cláusulas de los predicados, no solo tomando una como activa :(Tomaron el ejemplo del libro demasiado literal. Por ejemplo, para el p1 se debía utilizar su estructura: `a or b and c` y no la puesta en el libro. Viendo sus casos, en el caso 2 del predicado 1 el resultado del predicado debiese ser False (porque la cláusula 3 es false).	1	Plantearon bien el elsif y else (al menos en CACC y RACC) En CACC y RACC se trabaja sobre las cláusulas de los predicados. Si bien plantearon bien el elsif y el else, debieron haber trabajado con sus cláusulas y no sobre los "subpredicados".	4,5
GRUPO 3	Si se dan cuenta, en los momentos en que se prueba el primer bloque (if/elsif/else), automáticamente están probando el segundo bloque. Si se tiene el coverage 100% pero por ejemplo en el caso `assert_equal (33, @thirdPokemon.attack(@forthPokemon))` se tiene que cubre lo mismo que `assert_equal(13.5, @thirdPokemon.attack (@opponentPokemon))`. Otro tema es que los distintos asserts debiesen haber estado en distintos tests (funciones). Se debía borrar el test de ejemplo dado. Grande tigretony	0,75	Mismo comentario anterior sobre la redundancia en los tests. Si se tiene coverage 100%. Se plantea en los comentarios que para probar el valor de una cláusula, el resto DEBE tener cierto valor. Esto es solo verdad si uno busca forzar casos en que esa cláusula sea activa. Lo cual es necesario para los casos posteriores.	0,75	Para los predicados (if/elsif/else) se tienen mal planteadas las cláusulas activas. Para tener coverage 100%, por ejemplo, con la cláusula `@current_hp % 2 == 0`, la cláusula `@current_hp % 5 == 0` DEBE ser F. Me costó entender que al plantear `A && B || C`, A era `@current_hp < 25`. Dicho esto, se tiene que solo se cubre, para el p1, la cláusula A. Para el predicado del elsif, sí eran factibles :(. Para el predicado de las bonificaciones, los casos para A sí son factibles (B y C DEBEN ser T). Con respecto a B y C, se tiene buen ojo al ver que comparten un caso entre ellos.	1	Para el primer y segundo predicado, se plantean mal las cláusulas activas, no completando el coverage. Además no se considera el elsif.	0,5	No se considera el predicado del else.	4
GRUPO 4	Se tiene coverage 100%. Si se fijan en los primeros 3 tests se tiene el caso en que el predicado de desventaja es true. Y después hacen un tests para "solo ver eso", por lo que ese test es absolutamente redundante. Además, en el último test, se ve que el bonus da +1, por lo que se prueba eso 2 veces también (al final el 3er test no aporta nada). Bulbasaur water pls	0,75	Se tiene coverage 100%. En este caso se tiene altísima redundancia debido a que se confunden los conceptos de TR y TC.	0,5	Para el primer if, se cubre bien sobre la cláusula c. Sin embargo, sobre a y b faltan casos :(. Los tests `test_vida_set2_if_false` y `test_vida_set3_if_false` son redundantes. Para el elsif, falta el caso en que uno sea F y el otro T (2 casos). Naturalmente, uno está cubierto en el primer if (a = T, b = F). En el if de bonificación, para tener coverage completo, faltó el caso a=F, b=T y c=T. Hay varios casos redundantes :(.	1,5	Para el if, tengo el mismo comentario; falta el caso en que a y b sean F y c sea T. Además, hay muchos casos redundantes. Para el elsif, tengo el mismo comentario que en RACC. En el último predicado, para que a sea la cláusula mayor, debe poder definir el valor del predicado: por lo que b y c DEBEN ser T. De esta manera, si a es T, el predicado es T; si es F, el predicado es F. Se tienen dos TRs cubiertos: uno para `@third_pokemon.speed > 5` y el otro para `@third_pokemon.faster_than?(@fourth_pokemon)`.	1	Se confunden entre TR y TC. Se considera el `elsif`, pero se plantea mal. No se considera el `else`. (nota máxima 5,5)	4,75
GRUPO 5	Se tiene coverage 100%. Ojo con ese "segundo else": no existe, en este caso por suerte todos sus TRs se complementaban con los TRs del if, pero está mal.	1	Perfecto! Se tiene coverage 100% :D.	1	Muy bien en general. Ojo que para tener coverage 100%, se deben considerar como cláusulas activas a todas las cláusulas de los predicados y no solo 1.	1,5	Muy bien en general. Ojo que para tener coverage 100%, se deben considerar como cláusulas activas a todas las cláusulas de los predicados y no solo 1.	1,5	Bien planteados el elsif y el else. El "segundo else" no debería estar.	6
GRUPO 6	Se tiene coverage 100% :D. Pero hay tests redundantes. Se pueden dar cuenta que en los primeros 3 tests, también se evalúa el if de desventaja. El setup no se usa :P.	0,75	También se tiene coverage de 100%, pero se tiene redundancia. En CC da lo mismo si la cláusula no se evalúa.	0,75	Se esperaba que se separaran los tests. En su caso se vuelve inmantenible. El segundo predicado está bien analizado. En el primero, se agrupa tanto el if como el elsif en uno solo lo cual es erróneo. Son separados y si bien el elsif incluye al predicado del if, lo incluye con un `!`	0,5	Para el primer predicado se tiene coverage completo, hay tests que pueden removerse (se dice pero no se sacan :/). Mismo comentario para el segundo. Mismo comentario para el tercero. Se nota que se entendió el concepto :D! El setup siempre se llama previo a los tests. Si se deseaba tener una función para ser llamada (y poder cambiarle los argumentos), se pueden crear funciones auxiliares.	2	Elsif y else no se plantean correctamente. (nota máxima 5,5)	5
GRUPO 7	Se tiene coverage completo pero mucha redundancia. Para este caso se confunde el concepto de TR y TC. Al parecer faltó comunicación porque en CC sí se hace bien la diferencia.	0,5	Perfecto :D.	1	Los casos infactibles están correctos :) Solo se hace coverage completo para una cláusula por predicado (el resto queda parcialmente cubierto)	1,5	Con respecto al primer predicado, tengo el mismo comentario que en RACC. En el segundo, se tiene que el test `test_second_condition_active_clause_true` no tiene cláusulas (exclusivas) del elsif activas, además, hace que el predicado del elsif sea false, no true. En este predicado no se obtiene el coverage completo de ninguna cláusula. En el else, en `test_third_condition_active_clause_true` la cláusula `current_hp < 50` no es activa. De hecho no hay cláusulas activas con los valores dados. En `test_third_condition_active_clause_false` se tienen solo 3 TRs pero no alcanza a cubrir ninguna cláusula de manera completa :/.	0,5	Bien planteador el elsif y el else :D.	4,5
GRUPO 8	Se pusieron todas las combinaciones factibles. Se esperaba que pusieran la cantidad justa y necesaria solamente. Tienen coverage 100% pero con redundancia :).	0,5	Se entiende la separación por tests. Se tiene covergae 100% pero con redundancia, ya que (por ejemplo) si te fijas en el `test_clauses_1_1` ya cubres que la cláusula `@current_hp < 50` es true, `other.healthier_than (self)` es true, `bonification <= 1` es true, `other.faster_than(self)` es true y `@speed > 5` es true.	0,75	Se entiende el concepto, pero no se realiza el coverage completo (solo se realiza con una cláusula en cada predicado). Ojo que al separarlo por predicado se genera redundancia innecesaria.	1,5	Mismo comentario.	1,5	No consideraron elsif y else. (nota máxima 5,5)	5,25
GRUPO 9	Se tiene coverage 100% con redundancia. Bien generados los requisitos. Sin embargo, en los tests se debía testear el método `attack`. Lo que se hizo fue tomar el código del método y testear los predicados de manera independiente. Se tiene que cada test se encarga de cubrir un TR en específico, generando alta redundancia.	0,5	Mismo comentario	0,5	No se tiene coverage completo. P2 y P4 sí eran alcanzables. Se entendió el concepto de cláusula activa, pero se confundieron con que cada predicado tenía solo 1 cláusula mayor. La idea era que fueran rotando la cláusula mayor entre las cláusulas de un predicado y que tengan cobertura de sus TRs. Me encantó el desarrollo lógico del P3.	0,5	No se tiene coverage completo. Se entendió el concepto de cláusula activa, pero se confundieron con que cada predicado tenía solo 1 cláusula mayor. La idea era que fueran rotando la cláusula mayor entre las cláusulas de un predicado y que tengan cobertura de sus TRs.	1,5	Consideran bien elsif y else. Tests los realizan sobre la lógica del método y sobre la interfaz del modelo. Lamentablemente, las cláusulas mayores/menores sí eran alcanzables en RACC y CACC.	4
GRUPO 10	Se tiene coverage completo pero mucha redundancia. Con `test_battle_full_hp_elements_advantage`, `test_battle_less_hp_elements_disadvantage` y `test_low_even_hp_elements_disadvantage` bastaba.	0,5	No se tiene coverage completo ya que `@speed > 5` no es false nunca. Si te fijas bien, en la mayoría de tus tests tienes que `bonification <= 1` es True, `other.faster_than?(self)` es false y `@speed > 5` es True. Aprovechando poco los mismos tests ya creados.	0,5	Solo se cubren, para el primer predicado, el caso True de `b` y de forma completa a `c`. Se tiene mucha redundancia para esto: solo con `test_even_hp_lower_than_25_advantage` y `test_even_hp_between_25_50_advantage` se cubría lo mismo :(.	0,5	En blanco :(	0	Sin documentación, es difícil comprobar si su análisis fue correcto o no. No se sabe si considera elsif y else, pero no los considera en sus coverages.	2,5
GRUPO 11	Sin considerar el else, tiene coverage completo. Ojo que existe redundancia, ya que en el primer test, el p2 es true (por lo que el segundo test se vuelve irrelevante)	0,75	Coverage completo pero mismo comentario de la redundancia.	0,75	Perfecto :D. Los TRs de `other.faster_than?(self)` y `@speed > 5` están alrevés, pero en términos de coverage da lo mismo. Muy bien hecho. Un consejo para ahorrarse tests es que si se fijan, el último predicado es independiente al resto, por lo que los valores usados para cubrir las cláusulas de este predicado (valores en los casos 6, 7, 8, 9) pudieron haber sido puestos en casos anteriores sin afectar al resto de los análisis. (quizás esto resultó por _congelar_ los valores para el resto de las cláusulas, pero son con respecto a cada predicado :))	2	Muy bien :D.	2	No consideran bien el elsif y el else, sin embargo, sí se cubre bien :D	6,5
GRUPO 12	Coverage completo :D.	1	Perfecto :D.	1	No se tiene coverage completo de ninguna cláusula en ningún predicado :(. Se tienen coverage parciales y la mayoría redundantes.	0,25	No se tiene coverage completo. Por ejemplo para `@speed > 5` no se tiene como cláusula activa donde su valor sea F. Se tienen tests también muy redundantes como `test_bonus_bonification_by_disadvantage_T_F` que no agrega ningún TR nuevo a lo que ya han agregado los anteriores :/. Sin la documentación apropiada no se puede sacar más.	1	Sí consideran el elsif y else :D. Poca documentación hace difícil la corrección.	4,25
GRUPO 13	Perfecto :D.	1	Coverage incompleto: `bonification <= 1` no se hace false. Ojo que separaron los predicados para los tests, pero lo 3 primeros tests también cubren valores de las cláusulas del predicado de "desventaja". Por ejemplo, el test `test_desventaja_todo_true` no suma ningún TR cubierto por tests anteriores.	0,5	El coverage no está completo, ya que lo hicieron solo con una cláusula. La idea era que, para cada predicado, usaran todas las cláusulas como mayores y sus TRs fueran cubiertos.	1,5	Sucede lo mismo que el RACC con las cláusulas activas. Eso sí, en este caso, no se cubre la cláusula `@current_hp < 25` en el primer predicado (cuando es false no es cláusula mayor). Tampoco la cláusula `@current_hp < 50` en el segundo predicado, ya que para que sea la cláusula mayor `other.healthier_tan?(self)` debe ser False y es true en los primeros dos tests.	1	Sí considera el elsif y else, pero no los utiliza en RACC y CACC. (nota máxima 5,5)	5
GRUPO 14	La separación por predicado hace que tengan más tests de los necesarios. Por ejemplo, en el primer test el predicado `bonification <= 1 || (other.faster_than?(self) && @speed > 5)` es True pero igual después hacen un test para buscar cubrirlo. Además, no existe un predicado 5 :/.	0,75	Los tests debían ser realizados sobre el método `attack`, no probar con las cláusulas directamente. Es muy raro que sea la única batería con esta forma de testear.	0	El test `test_attack_not_multiplied_by_3` es redundante (se cubre también con `test_attack_multiplied_by_1_5` `test_attack_not_multiplied_by_1_5`) Como consejo les digo que les faltó, luego de crear los TRs, juntarlos en TCs. Dejando de lado eso, está muy bien :D.	2	Idem	2	muy bien :D Si bien no declaran formalmente el elsif y else, realizan el coverage completo.	5,75
GRUPO 15	Sí tienen covergae completo pero confunden TR con TC: por ejemplo en los primeros dos tests buscan probar valores para que el primer predicado sea true y false. Pero en estos tests los otros predicados también adquieren valores, los cuales cubren casos, pero luego realizan un tests específico para ese caso, siendo redundante. La idea de los tests es probar que el método haga lo que se piensa que haga. No se debe copiar el código del test en los tests porque si se desea cambiar, habría que cambiar todos los tests por eso.	0,5	No se tiene covergae completo :(, faltó que el valor de la cláusula `@current_hp < 50` sea false. En el Readme lo tenían bien, pero en el test le pusieron 31 de video, no 51. Lo bueno es que no tienen tests redundantes :D.	0,75	No se tiene coverage completo D:. La idea era que fueran rotando el rol de cláusula mayor entre las cláusulas y cubriéndo sus TRs. Ahora bien, con una cláusula mayor en cada predicado, los casos teóricos (readme) están bien. Ojo con usar random en los tests. Por ejemplo, en el test1, no se asegura que la c2 sea True. Tampoco se puede el caso dos, no pudiendo asegurar coverage completo (bajo RACC) con estos dos tests (porque se necesita que el valor de las cláusulas menores sean iguales en ambos casos). En el segundo predicado vuelven a usar random, pero ahora, con las posibilidades entregadas, sí se aseguran que sean los mismo valores de las cláusulas. Ocurre lo mismo con el tercero :D.	1,5	En blanco :(	0	Confunden TR con TC. No consideran completos los predicados de else y elsif.	3,75
GRUPO 16	Perfecto! Se tiene coverage 100% :D.	1	Muy bien :D, se tiene coverage 100% Pero con el test 1 y 3 ya se tiene el coverage completo, dejando al test 2 como redundante. Fíjate que en el test 1, las cláusulas del `elsif` también son true!	0,75	Muy bien trabajado :D. Se tiene coverage 100% con sus tests. Se nota que entendieron lo de cláusula activa. Un consejo es que pudieron haberse ahorrado casos de tests si realizaban una iteración de combinar tests.	2	El coverage no está completo :(, faltaron casos para el segundo predicado. Los otros dos están completamente cubiertos :D. Además, se tienen 4 tests redundantes: - `test_healthier_than_determines_second_ci_true` - `test_hp_less_than_50_determines_second_ci_false` - `test_hp_less_than_50_determines_second_ci_true` - `test_div_by_5_determines_first_ci_true_faster_than_determines_third_ci_true`.	1,5	Bien en general :D	6,25
GRUPO 17	No se tiene coverage completo debido a que no se tiene el caso en que el segundo predicado es F (dada su forma de proponerlos, ya que en sí, en el primer test, el segundo predicado es false -> es elsif y el if es True).	0,5	Perfecto :D.	1	No se tiene coverage completo :(, faltó el caso para el active clause `@speed > 5` sea falso. Pero muy bien. Ojo que el nombre `test_p1_false_for_ac1` se usa dos veces (la primera vez es igual a `test_p1_true_for_ac2` :/). En el primer caso, la segunda cláusula del predicado es True (no False). El test `test_p2_false_for_ac4` no aporta nada, no hace lo que dice el comentario (es igual al anterior).	1,5	En blanco :(	0	Bien en general, una lástima perder los puntos en CACC :(, bastaba con un comentario o copiar/pegar los tests de RACC.	4
GRUPO 18	Se tiene coverage completo, pero se confunen entre TR y TCs. Por ejemplo, el primer test toma en consideración que el segundo predicado es False, y que el tercer predicado es False también.	0,5	Mismo comentario anterior.	0,5	Bien en general, eso sí, para el predicado 1 y 3 se debían considerar (por turnos) a todas las cláusulas como activas para tener el coverage completo.	1,5	Mismo comentario con respecto a que el coverage debía ser sobre todas las cláusulas de los predicados. En su análisis, para el coverage para el segundo predicado es erróneo, ya que para que a o b sean cláusulas activas, el otro debe ser false. Por lo tanto, solo se cubre la mitad de los TRs. Además, se tiene que el test `test_predicado_2_2_false` es redundante (`test_predicado_2_1_false` cubre lo mismo)	1	Se confunden entre TRs y TCs. Para los tests era innecesario que pusieran la lógica del método `attack`. En el caso de que quisieran hacerlo así, ojalá tener una función auxiliar para obtener el `raw_damage` y así, ante una modificación en su cálculo, no tengan que modificar todos los tests.	4,5
GRUPO 19	Se tiene coverage completo. Sin embargo, se tienen muchos casos redundantes porque se confunden en los conceptos de TR y TC. Nota: En el pc.md, en el predicado 4, p4 es false, no true.	0,5	Perfecto :D.	1	Muy bien :D, se tiene coverage completo. Hay un único test redundante: el test `test_6` no cubre ningún TR "nuevo". (documentación) No se entiende la relación entre los TRs y los TCS. Hubiese sido útil haberle puesto ids a cada uno. (documentación) En el predicado 3, en la cláusula activa h se "copió y pegó mal" con respecto a la g.	2	La verdad no entendí su documentación, recuerden que es una actividad evaluada. No se tiene coverage completo y se tienen varios tests redundantes (4 tests) PD: el setup que tienen no se utiliza, sería mejor sacarlo.	1	Bien considerados el elsif y el else. Se confunden en TR y TC.	5,5
GRUPO 20	Muy bien :D. Un comentario nomás: en el test 2, el predicado 3 es false y en el test 3 el predicado 3 es true.	1	Perfecto :D.	1	Muy bien explicado :D y se tiene coverage completo. Eso sí, sin disminuir el puntaje, existe un test redundante: `test_predicate_2_term_1_true`, el cual se cubre con los primeros tests.	2	Al parecer no hubo coordinación, ya que en estos test, los casos redundantes (al menos de manera fácil de visualizar como `test_that_hp_not_divisible_by_2` y `test_that_hp_not_divisible_by_5`) no fueron comentados D:. También hay otras redundancias, pero en general se nota que entendieron el concepto :D.	1,5	El teardown era innecesario en estas baterías.	6,5
GRUPO 21	Está bien el coverage, pero los nombres usados para los pokémons v/s sus tipos va en contra de las buenas prácticas de la programación. Si se fijan, si llamaban a los pokémones "a", "b" y "c" el código es más mantenible.	1	Perfecto :D.	1	Con respecto al predicado 1, solo se hizo la cobertura completo con C3, y bastaba con una combinación de las 3 (1 y 2 | 3 y 4 | 5 y 6). Con la C1 y C2 se tiene cobertura parcial. Con el predicado 2 sí existían cláusulas activas :(. Con el predicado 3 está bien. Se nota que el concepto de cláusula activa se entendió, pero se quedaron demasiado con el ejemplo del libro :/. Además, les faltó separar los TRs de los TCs para crear menos tests.	1	El predicado 2 se cubre completamente. El predicado 1 sólo con respecto a la 3era cláusula :P. Con respecto al predicado 3, no se cubre ninguna cláusula :(La idea no era copiar el código del método en cada test. Al menos pudieron haber creado una función auxiliar que calculara el `raw_damage` (mejora mantenibilidad)	1	Ojo con cómo hacen los tests, estos son código que posteriormente se debe mantener, por lo que es bueno hacerles revisiones para evitar circunstancias en que no sea legible o que haya mucha duplicidad de código.	5
GRUPO 22	Se tiene coverage completo :D. Ojo que sus asserts pueden generar muchísimos falsos positivos.	1	Se tiene coverage 100%. Ojo que al separar por predicado se generan redundancias. Por ejemplo, el caso que en el predicado 3 todas sus cláusulas sean false, ya está cubierto por todos los tests anteriors a ese predicado. Otro ejemplo es el caso de trues en el predicado 2.	0,5	Ojo que se está creando un test por cada TR. Si bien se evitan las redundancias obvias, esto se presta para que se genere redundancia igual. Se nota que se entendió el concepto de active clause :D. El único test redundante termina siendo `test_active_f_false` ya que las cláusulas del predicado 3 toman el mismo valor en los TRs de los predicados 1 y 2. También se tiene que los TRs del predicado 1 cubiertos por el test `test_active_c_true`, están también cubiertos por: `test_active_g_false` y `test_active_h_false`. O bien que los TRs del predicado 1 cubiertos por `test_active_c_false`, son cubiertos por `test_active_d_false`. Faltó una agrupación de los TRs a TCs, y no llegar y programarlos.	1,5	Mismo comentario anterior, a excepción de que en el predicado 3, los TRs ya se cubrían en los tests anteriors (ya que F F F no está aquí). Solo que acá faltó el caso en que en el segundo predicado, ambos fueran falsos. Muy buena práctica el método auxiliar :D	1,5	Muy bien en general. Ojo con los falsos positivos ya que sí no se detectan de forma temprana, eventualmente pueden dar una sensación de falsa seguridad.	5,5
GRUPO 23	Perfecto :D.	1	Perfecto :D.	1	En este criterio, se debía realizar cobertura utilizando a todas las cláusulas como mayores. Además, no se realiza cobertura sobre el segundo predicado (se realiza de manera parcial, que dado el análisis hecho, fue casualidad).	0,5	Análisis súper completo sobre los TRs. Eso sí, en el segundo predicado no se toma en cosideración que es un elsif, tomando como TR válido uno que no sirve (ya que el if sería verdadero, dejando inactivas las cláusulas del elsif). Nota máxima 5,5 :/.. aunque no los afecta :D. Lo único que faltó fue agregar los TRs en TCs para evitar redundancia y cubrir lo mismo con menos tests. Muy bien en general!	1,5	Entre CACC y RACC se nota una diferencia grande, para la próxima intenten comunicarse mejor!	5