Universidad Central de Venezuela
Facultad de Ciencias
Escuela de Computación
Lenguajes de Programación (6204)
Semestre 1-2024

Proyecto #1 – Haskell
Torneo del Poder

En Dragon Ball Super, el Torneo del Poder o Torneo de la Fuerza es una competencia de supervivencia entre universos. Como fanático de Dragon Ball y experto en programación funcional, has decidido crear un simulador del torneo que permita a los usuarios recrear algunas de las batallas más icónicas de la serie.

Parte 1 – Creación de Personajes

Cada personaje contará con un nombre (String), un nivel de poder (Int), una salud inicial (Int) y una lista de técnicas y/o habilidades de lucha iniciales (List[Technique]), cada técnica contará con el nombre (String) y el multiplicador de daño (Int).

Para ello debes usar las siguientes estructuras de datos:

data Character = Character String Int Int [Technique] deriving (Show, Eq)

data Technique = Technique String Int deriving (Show, Eq)

data Response = Wrong String | Success Character deriving (Show, Eq)

Dada la información anterior, escriba la función

createCharacter :: String -> Int -> Int -> [Technique] -> Response

Ejemplos:

1) createCharacter (“Piccolo” 6 50000 [Technique “Makankosappo” 3500, Technique “Special Beam Canno” 2000])

Retorno: Success (Character “Piccolo” 6 50000 [Technique “Special Beam Canno” 2000, Technique “Makankosappo” 3500])

2) createCharacter (“Krillin” 3 0 [Technique “” 3000])

Retorno: Wrong “Invalid input data”

3) createCharacter (“Goku” 10 120000 [Technique “Kamehameha” 6000, Technique “Ultra Instinct” 50000, Technique “Instant Transmission” 4000])

Retorno: Success (Character “Goku” 10 120000 [Technique “Instant Transmission” 4000, Technique “Kamehameha” 6000, Technique “Ultra Instinct” 50000])

Notas:

- Las técnicas deben ser ordenadas de menor a mayor en base a su multiplicador de daño, creando una función llamada **sortTechniques :: [Technique] -> [Technique]**.
- Investigar y documentar el uso de la expresión **deriving (Show, Eq).**
- Se debe validar cada campo. Todos deben ser distintos de null, los enteros deben ser mayores a 0 y la lista de técnicas debe contener al menos una posición (donde cada posición debe tener un nombre válido y un daño mayor a 0). En caso de que no se cumpla alguna de las condiciones anteriores el resultado de la función será el string genérico **“Invalid input data”**.

Parte 2 – Técnicas de Lucha

Los personajes en Dragon Ball pueden aprender nuevas técnicas, para ello debes crear una función que tome como entrada un personaje (Character), una técnica de lucha (Technique) y que devuelva el personaje con la nueva técnica aprendida o un mensaje de error según sea el caso (Either String Character).

learnTechnique :: Character -> Technique -> Response

Casos de prueba:

1) learnTechnique (Character “Hit” 9 100000 [Technique “Time Skip” 20000]) (Technique “Pure Progress” 5000)

Retorno: Success (Character “Hit” 9 100000 [Technique “Pure Progress” 5000 , Technique “Time Skip” 20000])

2) learnTechnique (Character “Trunks” 0 0 [Technique “” 0]) (Technique “Masenko” 4000)

Retorno: Wrong “Invalid character”

3) learnTechnique (Character “Jiren” 11 150000 [Technique “Power Impact” 10000]) (Technique “” 0)

Retorno: Wrong “Invalid technique”

4) learnTechnique (Character “Vegeta” 8 110000 [Technique “Final Flash” 6000, Technique “Big Bang Attack” 7000]) (Technique “Final Flash” 6000)

Retorno: Wrong “Technique already learned”

Notas:

- Verificar que los datos de entradas sean válidos y que la nueva técnica no exista en la lista del personaje.
- Para las validaciones del personaje y técnica pueden reutilizarse las funciones de la parte anterior.
- La nueva técnica debe agregarse de forma ordenada.

Parte 3 – Batallas

Las peleas o batallas son lo que más abunda en la serie, para ello escriba una función que simule la batalla entre dos personajes. Como entrada se recibirán dos personajes (Character) y un valor String que indicará cuál técnica deben utilizar para la batalla, esperando como resultado una tupla (String, Int) que indicará el nombre del vencedor y su salud restante.

battle :: Character -> Character -> String -> (String, Int)

La batalla debe realizarse por turnos. Por cada turno, ambos personajes atacan a su oponente, el daño infligido por la técnica seleccionada se calcula multiplicando el nivel de poder del personaje por el multiplicador de daño de la técnica. Se debe aplicar la siguiente fórmula:

health1 = health1 – (powerLevel2 * damageMultiplier2)

health2 = health2 – (powerLevel1 * damageMultiplier1)

Casos de prueba:

1) battle (Character “Hit” 9 100000 [Technique “Pure Progress” 5000 , Technique “Time Skip” 20000]) (Character “Vegeta” 8 110000 [Technique “Final Flash” 6000, Technique “Big Bang Attack” 7000]) “last”

Retorno: ("Hit", 44000)

2) battle (Character “Hit” 9 100000 [Technique “Pure Progress” 5000 , Technique “Time Skip” 20000]) (Character “Vegeta” 8 110000 [Technique “Final Flash” 6000, Technique “Big Bang Attack” 7000]) “first”

Retorno: ("Tie", 0)

3) battle (Character "Mr. Satan" 1 10000 [Technique "Punch" 550]) (Character "Monaka" 1 10000 [Technique "Punch" 1000]) “first”

Retorno: ("Monaka", 4500)

Notas:

- Para esta parte no se solicitan validaciones, pero se deja a su criterio si desea implementarlas. Las pruebas a realizar serán siempre con el escenario ideal, es decir, personajes y técnicas válidas, así como también el String que indica cuál técnica utilizar estará en el rango de { “first”, “last” }.
- La batalla debe continuar hasta que uno de los personajes se quede sin puntos de vida (health <=0).
- Si ambos personajes en una determinada ronda poseen health <=0, el resultado esperado en la tupla sería el de empate **(Tie, 0)**.

Parte 4 – Torneo del Poder

Crea una función que simule el Torneo del Poder. La función debe permitir organizar batallas entre los personajes hasta que solo quede uno o ninguno. Se recibe como entrada una lista de personajes (List[Character]) y como respuesta se espera una tupla (String, Int) que indique el nombre del ganador y salud restante.

tournamentOfPower :: [Character] -> (String, Int)

Ejemplos:

Definiendo lista de personajes.

goku = Character "Goku" 10 100000 [Technique "Kamehameha" 1000]

vegeta = Character "Vegeta" 9 90000 [Technique "Final Flash" 900]

jiren = Character "Jiren" 8 80000 [Technique "Power Impact" 800]

toppo = Character "Toppo" 7 70000 [Technique "Justice Flash" 700]

gohan = Character "Gohan" 8 60000 [Technique "Masenko" 600]

dyspo = Character "Dyspo" 8 60000 [Technique "Light Bullet" 600]

hit = Character "Hit" 5 50000 [Technique "Time-Skip" 500]

cabba = Character "Cabba" 4 40000 [Technique "Galick Cannon" 400]

frost = Character "Frost" 3 30000 [Technique "Chaos Beam" 300]

botamo = Character "Botamo" 2 20000 [Technique "Botamo Shield" 200]

1) tournamentOfPower [toppo, goku, dyspo, botamo, hit, gohan, cabba, vegeta, jiren, frost]

Retorno: ("Goku", 36900)

2) tournamentOfPower [vegeta, frost, cabba]

Retorno: ("Vegeta", 78400)

3) tournamentOfPower [gohan, dyspo]

Retorno: ("Tie", 0)

4) tournamentOfPower [jiren]

Retorno: ("Jiren", 80000)

Notas:

- Reutilizar la función **battle** de la parte anterior.
- Asumir que los personajes usan su última técnica (la más poderosa) y que el orden de las batallas se inicia desde la posición 0 de la lista de dos en dos.
- Puede llegar a existir el resultado de empate **(Tie, 0)**.

Puntuación:

- Parte 1 y 2, 5 ptos c/u. Total 10 ptos.
- Parte 3, 10 ptos.
- Parte 4, 3 ptos extras.

Consideraciones para la entrega

- Sólo pueden utilizar operaciones del Preludio de Haskell.
- No está permitido el uso de palabras reservadas como: **do**, **if**, **else**, **then**. Utilizar composición de funciones y funciones de orden superior.
- Las funciones solicitadas deben estar definidas dentro de un único archivo llamado **`Proyecto1.hs`** y las funciones deben tener el mismo nombre y número de argumentos descritos en el enunciado.
- Las respuestas de las funciones deben ser iguales a las solicitadas en el enunciado, de no ser así el script de pruebas arrojará error y se calificará como incorrecto.

- Utilice comentarios en Haskell para separar las soluciones de las preguntas y documentar lo que considere necesario. En Haskell hay dos tipos de comentarios: línea (`--`) y bloque (`{-` y `-}`). Incluya un comentario de encabezado con sus datos en el archivo.
- No está permitido utilizar soluciones de terceros (incluyendo soluciones generadas por LLMs). Cualquier material consultado para ideas de solución o documentación debe ser referenciado, indicando para qué sirvió. Esto lo pueden documentar en un archivo en formato *markdown* (**`README.md`**) o directamente como comentarios en **`Proyecto1.hs`**. No están permitidos documentos en ningún otro formato.
- El proyecto es **estrictamente en parejas** y la fecha de entrega será el **viernes 31/05/2024**, hasta las 11:59 pm (VET).
- El proyecto se debe adjuntar por e-mail a la dirección `ldpucv@gmail.com` usando como asunto: `[Proyecto Haskell] <Apellido(s)> <Nombre(s)>, <Cédula1> - <Apellido(s)> <Nombre(s)>, <Cédula2>`.

José Yvimas, Mayo 2024