Actividad 1 - Nano Jack

UNSAM Programa

Verano 2020 - adaptado de EXACTAS Programa

Vamos a escribir un programa en `Python` que *juegue* a `Nano Jack`, un juego de cartas basado en el famoso *Black Jack*, pero con reglas un tanto relajadas.

En el juego `Nano Jack` se utiliza un mazo de cartas por cada jugador que participa en el juego, *combinados* y *mezclados*. Cada jugador retira una a una cartas del mazo y va sumando sus valores. Si las cartas *suman* 21 gana, si se pasa pierde.

El objetivo de esta actividad es escribir un programa en `Python` que *simule* varios jugadores jugando al `Nano Jack`. **Envíen la tarea a:** `unsamprograma@unsam.edu.ar`

En nuestro juego, un mazo de cartas será representado por una lista de números enteros entre 1 y 13, repetida una vez por cada palo (es decir 4 veces cada una). Vamos a escribir funciones en `Python` que, combinadas, permitan jugar al `Nano Jack`. Además de estas funciones se necesita un programa que las llame en cierto orden, y con los parámetros adecuados para simular un juego :

1. `generar_mazos(n)`: recibe n, un número entero positivo que representa la cantidad de mazos con los que se jugará (que dependerá del número de jugadores). Devuelve una lista conteniendo los valores de las cartas de los n mazos mezclados de manera aleatoria.

   Ayuda: Se puede utilizar la función de `Python` `random.shuffle(a)` del módulo `random` para mezclar las cartas.

2. `jugar(m)`: recibe un mazo m (que es una lista de números enteros entre 1 y 13) y retira cartas una a una hasta que la suma de los valores de las cartas retiradas sea mayor o igual a 21. Devuelve la suma obtenida (aún si ésta fuera mayor a 21). Si no es posible retirar cartas de m (mazo vacío) la suma de los valores de las cartas retiradas es 0.

   Para que se puedan combinar luego varios jugadores, las cartas retiradas son quitadas del mazo, es decir, al terminar esta función, el mazo m deberá tener menos cartas que antes (tantas menos como fueron retiradas para obtener el resultado de la función).

   Ayuda: `una_lista.pop(0)` devuelve el primer elemento de `una_lista`, retirándolo de `una_lista`.

3. `jugar_varios(m,j)`: recibe un mazo de cartas `m` y un número de jugadores `j`. Usando la función definida en el punto anterior (`jugar`) para cada uno de los j jugadores, devuelve una lista con la suma de los valores de las cartas obtenidas por cada jugador.

   Ayuda: llamar sucesivamente a la función `jugar` con el mazo restante.

4. (*Optativo*) `ver_quien_gano(resultados)`: recibe la lista de enteros (`resultados`) devuelta por la función `jugar_varios`. Se considera ganador al jugador que haya sumado 21. Devuelve una lista que indica, para cada jugador, si gano (con un 1) o si no (con un 0). Puede haber varios ganadores.

5. (*Optativo*) `experimentar(rep,n)`: recibe dos número enteros positivos: *rep* y otro n. Juega `Nano Jack` *rep* veces con n jugadores. Devuelve una lista con n valores donde cada valor representa cuántas veces ganó el jugador correspondiente (el elemento 0 guardará cuántas partidas ganó el primer jugador, el elemento 1 guardará cuántas el siguiente y así sucesivamente).

Nota: en caso de que existan funciones de `Python` que resuelvan alguno de los ítems pedidos total o parcialmente, no se permite utilizarlas (salvo indicación contraria).