Programação I – 2017/2018

Enunciado do Projeto – *Blackjack*

Introdução

O *Blackjack* (conhecido também por *21*) é um jogo de cartas que é jogado entre um *dealer*, que representa o casino, e um ou mais jogadores. Para cada jogador, o objetivo é simples: juntar uma sequência de cartas (uma mão), de modo a que o seu valor conjunto seja maior que o do *dealer*, sem ultrapassar 21. Cada jogada envolve uma aposta por parte do jogador.

Neste projeto vai ter que construir um programa que realize uma versão simplificada do *Blackjack* que permita realizar um jogo entre o *dealer* e um jogador humano.

As regras do jogo

Valor das cartas

As cartas com figuras (Valete, Dama, Rei) valem 10 pontos.

O Ás pode valer 1 ou 11 pontos, consoante a situação, isto é, consoante a mão em que se insere. O Ás valerá 1 apenas quando o valor 11 fizer a mão ultrapassar os 21 pontos.

Todas as outras cartas (do 2 ao 10) têm o valor correspondente ao seu número.

O desenrolar do jogo

O jogador inicia o jogo com uma certa quantidade de fichas em sua posse e um valor de aposta definido para cada ronda. Por omissão, o jogador começa com 100 fichas, sendo 10 o valor de cada aposta.

Cada ronda do jogo é jogada com um baralho novo de 52 cartas.

Cada jogo decorre depois de acordo com os seguintes passos:

1. São distribuídas duas cartas a cada um (jogador e *dealer*). Essas cartas são a *mão inicial*. As cartas das mãos (do jogador e do *dealer*) ficam todas visíveis, exceto a segunda carta do *dealer*.
2. É a vez do jogador. Existem dois cenários possíveis:
    a. O jogador tem *blackjack* (Um Ás e uma carta de valor 10).
    Neste caso, a sua jogada termina automaticamente (o jogo continua a partir do ponto 4).
    b. Caso não tenha *blackjack*, vai ter que decidir a sua próxima ação.
3. Decisão do jogador. Não tendo *blackjack*, o jogador tem que decidir entre duas ações possíveis (“HIT” ou “STAND”) :
    a. “HIT” – pede mais uma carta. Três situações podem ocorrer:
        i. Se o valor da nova carta fizer a mão exceder 21, o jogador perde automaticamente a ronda (situação designada por “BUST”). (o jogo continua no ponto 9).
        ii. Se o valor da mão igualar os 21 pontos, a sua jogada termina, passando a vez ao *dealer*.
        iii. Se o valor da mão ficar inferior a 21, o jogador pode novamente decidir (“HIT” ou “STAND”), o que poderá sempre voltar a fazer enquanto não se verificar uma das duas situações anteriores.
    b. “STAND” – indica que não pretende mais cartas. A partir daí, já não poderá fazer qualquer alteração à sua mão, passando a vez ao *dealer*.
4. É a vez do *dealer*. Após uma decisão de “STAND” por parte do jogador, ou este ter feito *blackjack*, toma a vez o *dealer*.

5. Se o jogador terminou com um *blackjack* (21 pontos em duas cartas), e se o *dealer,* com a sua carta visível ainda puder fazer um *blackjack* (se a carta visível for Ás ou valer 10 pontos), o *dealer* verifica se, com a sua segunda carta, ele próprio também tem *blackjack*: se tiver, empata a ronda; se não tiver, perde (segue o jogo para o ponto 9).
6. Caso não se verifique a situação do ponto anterior, a segunda carta do *dealer* é revelada.
7. Decisão do *dealer*. Enquanto não atingir os 21 pontos, as decisões possíveis são as mesmas ("HIT" ou "STAND") sendo a decisão concreta determinada pela seguinte regra:
    a. Se a mão do *dealer* valer menos de 17 pontos, a decisão é "HIT";
    b. Se a mão do *dealer* valer 17 ou mais pontos, a decisão é "STAND".
8. Tal como para o jogador, o *dealer* também poderá rebentar após uma decisão de "HIT", perdendo automaticamente o jogo.
9. Distribuição dos ganhos. Considerando que o jogador apostou **x**, tendo um saldo de **s.**
    a. Caso o jogador perca ("BUST" ou o *dealer* consiga um número superior de pontos), perde o valor da aposta. O seu saldo passa a ser **s-x**.
    b. Caso o jogador ganhe (conseguiu um número superior de pontos ou o *dealer* "rebentou"), recebe o dobro do que apostou (recebe **2x**). O seu saldo passa a ser **s+x**.
    c. No caso de o jogador ganhar com *blackjack*, em vez de **2x**, recebe **5x/2**. Ou seja, o seu saldo passa a ser **s+3x/2**.
    d. Caso empatem, o jogador recupera o valor apostado. O seu saldo mantém-se inalterado.
10. O jogo regressa a 1, para uma nova ronda, a não ser que o jogador decida terminar.

Informação complementar sobre as regras de jogo pode ser encontrada aqui.

Objetivo do Trabalho

O objetivo do trabalho é desenvolver um programa que permita realizar um jogo de *Blackjack* entre o computador (*dealer*) e um jogador humano que fornecerá as suas jogadas interactivamente. O funcionamento deverá seguir as regras do jogo indicadas acima.

Funcionamento do Programa

Para além das indicações dadas nesta secção, **veja com cuidado os exemplos de execução fornecidos em anexo**. O programa deverá funcionar de acordo com o seguinte:

1. Pedir o nome do jogador (uma *string*).
2. Pedir o valor do montante total com que o jogador quer iniciar o jogo (um *float* correspondente ao valor das suas fichas). Se for dada uma resposta inválida, deverá assumir o valor por omissão (100.0).
3. Pedir o valor de cada aposta (um inteiro). Este valor é igual em todas as rondas do jogo. Se for dada uma resposta inválida, assumir o valor por omissão (10).
4. Realizar o jogo de acordo com as regras e os valores indicados acima. Deverá começar por mostrar o saldo inicial, e o valor da aposta a ser utilizado durante o jogo (igual em todas as rondas).
5. Em cada ronda é utilizado um novo baralho, cuja sequência de cartas deverá ser lida de um ficheiro 'baralho_<i>.txt', em que <i> deverá ser substituído pelo número da ronda[1]. Os ficheiros com os baralhos a utilizar são disponibilizados no *moodle*.

[1] Assim, por exemplo, na primeira ronda, o baralho deverá ser lido do ficheiro com o nome 'baralho_1.txt', na décima ronda "baralho_10.txt", etc.

a. Em primeiro lugar, carregar então o conjunto de cartas já baralhadas do ficheiro. As cartas deverão posteriormente ser utilizadas pela mesma ordem que estão no ficheiro.
b. Mostrar as cartas iniciais de cada um (a primeira e terceira cartas são para o jogador, a segunda e quarta são para o *dealer*). A segunda carta do dealer fica, para já, escondida. Deverá ser mostrado o valor da mão do jogador.
c. Caso o jogador tenha *blackjack*, deverá ser apresentada uma mensagem indicando esse facto, terminando imediatamente a vez do jogador.
d. Caso o jogador não tenha *blackjack*, pedir ao utilizador a sua decisão, informando o utilizador sobre as hipóteses possíveis. O utilizador deverá fornecer a sua decisão escrevendo uma das *strings* "STAND" ou "HIT", em maiúsculas ou minúsculas.
e. Caso o utilizador introduza uma *string* inválida (diferente das duas referidas), deverá ser assumido "STAND".
f. O utilizador deverá ser sempre informado da ação considerada.
g. Caso o utilizador tenha escolhido "HIT", deverá ser mostrada a sua mão atualizada (mencionando o novo valor), com a indicação "BUST", caso tenha rebentado. Caso não tenha rebentado, nem atingido os 21 pontos, repete a partir de 'd'.
h. Quando o utilizador escolher "STAND", a sua jogada termina, passando a ser executada a jogada do *dealer*.
i. Revelar a mão do *dealer*. Exceção: Se o jogador ganhou com *blackjack*, ou rebentou, a segunda carta do *dealer* não é revelada.
j. Mostrar a sequência de decisões do *dealer* (e a respetiva evolução da mão) enquanto este não fizer "STAND".
k. Mostrar o resultado final e ganhos correspondentes. O ganho (a diferença entre o saldo anterior e o saldo após a ronda, deverá ser apresentado arredondado às decimas).
l. Verificar se o jogador ainda tem dinheiro disponível e, se for o caso, perguntar se quer jogar mais uma ronda ("QUIT" para terminar). Se sim, voltar a 'a'. Deverá interpretar como 'sim' qualquer resposta diferente de "QUIT".

6. No final, mostrar as estatísticas do jogo, na perspetiva do jogador humano: número de rondas realizadas, saldos inicial e final, número de rondas ganhas/perdidas/empatadas, número de vitórias com *blackjack* (ver formato dos resultados nos exemplos de execução).

Programação

Tendo liberdade em alguns aspetos, a sua implementação deverá respeitar as seguintes indicações.

Representação / Estruturas de dados

Cada carta deverá ser representada por um par (<face>,<naipe>), no qual ambos os elementos são strings. As faces possíveis são: 'A', '2', '3', ... '10', 'J', 'Q', 'K'. Os naipes são: 'O', 'P', 'C', 'E'. Por exemplo:

- O Ás de espadas é representado pelo par ('A','E')
- A Dama de paus, pelo par ('Q','P')

Um baralho completo é uma lista das 52 cartas existentes.

Uma mão é também uma lista de cartas.

Leitura dos baralhos de ficheiros

Como referido acima, o baralho a utilizar na ronda <i> deverá ser lido do ficheiro 'baralho_<i>.txt', devendo posteriormente as cartas ser utilizadas pela ordem em são lidas deste.

Cada ficheiro tem 52 linhas, todas diferentes, uma para cada carta. Em cada linha, o primeiro valor representa a face da carta, o segundo, separado por um espaço, o naipe. Por exemplo, um dos baralhos, tem como primeiras linhas:

```
10 P
Q P
2 E
5 E
A C
...
```

Funções a definir

Para além de outras que considere necessárias, deverá definir as seguintes funções básicas (consulte a *docstring* destas funções num documento anexo):

- **valor(mao)** – função que, dada uma mão, devolva o inteiro correspondente ao seu valor.
- **bust(mao)** – função que, dada uma mão, determine se esta "rebentou" (resultado booleano).
- **blackjack(mao)** – função booleana que, dada uma mão, determina se esta corresponde a um *blackjack* (21 pontos em duas cartas).
- **ler_baralho(n)** – função que, dado o número da ronda, devolve o baralho guardado no ficheiro correspondente (uma lista de cartas).
- **decisao_dealer(mao)** – função que, dada uma mão, devolve a decisão do *dealer*: "HIT" ou "STAND" (ambas em maiúsculas).
- **ronda(i,jogador,aposta)** – executa a ronda **i**, para o **jogador** (o seu nome), sendo dado também o valor da **aposta**. Esta função deverá devolver um par **(resultado, ganho)**, onde o **resultado** é "vitória blackjack", "vitória", "derrota" ou "empate" e **ganho** é um valor positivo ou negativo que corresponde ao que o jogador ganhou ou perdeu nessa ronda.
- **jogar()** – função que executa um jogo completo, ou seja, deverá concretizar o funcionamento do jogo a partir do ponto 1 (secção "Funcionamento do Programa", acima).

O vosso programa deve manter o estado do jogo e fazer as jogadas indicadas pelo jogador, até terminar. Deverá garantir que as jogadas indicadas são válidas, conforme as regras, e garantir a execução correta do jogo.

Exemplos de Execução

Em anexo a este enunciado é disponibilizado um documento com alguns exemplos de execução que ilustram o modo como o programa deve funcionar.

Nestes exemplos, o funcionamento do programa considera já as extensões que a seguir se descrevem.

Extensões ao jogo

Fornecer sugestão ("HINT")

Incorporar no programa a possibilidade de o jogador pedir uma sugestão para o ajudar a escolher a ação a executar. Assim, aquando do passo 5d acima, para além das ações de "STAND" o "HIT", o jogador tem a possibilidade de escrever "HINT". Neste caso, deverá ser apresentada a probabilidade de o jogador rebentar caso escolha pedir mais uma carta ("HIT").

Para concretizar esta opção, defina uma função:

- **calcula_hint(mao_jogador, mao_dealer)** – devolve a probabilidade do jogador rebentar se fizer "HIT" sobre a mão dada, com base nas cartas que estão visíveis (as suas e primeira carta da mão do dealer). O valor retornado deverá ser aproximado à terceira casa decimal

Escolha da regra do Casino

A forma do *dealer* decidir pode ser refinada.

A regra de decisão geral (sempre válida) é:

- Se a mão vale menos de 17 pontos, a decisão é "HIT";
- Se a mão vale mais de 17 pontos, a decisão é "STAND".

Quando a mão vale exatamente 17 pontos, existem dois modos possíveis de tomar a decisão. Estas duas alternativas são designadas por "Soft 17" (S17) e "Hard 17" (H17):

- S17 – o *dealer* decide "STAND" sempre que tem 17 pontos. (corresponde à que foi referida para o funcionamento do jogo).
- H17 – o dealer decide "STAND" apenas se os 17 pontos forem "hard". Na presença de uma mão com 17 pontos "soft", o *dealer* decide "HIT".

O valor de uma mão diz-se "soft" quando o adicionar mais uma carta à mão nunca fará o jogador "rebentar". No caso de uma mão valer 17 pontos, a distinção só existe se a mão tiver pelo menos um Ás (uma mão de 17 pontos sem nenhum Ás é sempre "hard").

Por exemplo: Uma mão com um **Ás** e um **6** vale 17, mas é "soft", pois o Ás está a valer 11 pontos e, caso a carta seguinte valha 10, o Ás pode passar a 1 ponto ficando, nesse caso, a mão a valer na mesma 17 pontos (Ás(1) + 6 + 10), mas agora "hard".

Sintetizando, um valor de 17 para a mão é "soft" se esta tem um Ás a valer 11 pontos; é "hard" se todos os Ases da mão estão a valer 1 ponto.

Estender o programa com a possibilidade de o utilizador escolher a regra de casino a utilizar. Assim, antes das rondas começarem (após o ponto 3 acima), deverá ser perguntado ao utilizador com que regras de casino se vai realizar o jogo ('S17' ou 'H17'). Se for dada uma resposta inválida, assumir 'S17'.

Para concretizar esta opção:

- defina uma função booleana **soft(mao),** que devolve *True,* se a mão fornecida for *soft*, e *False*, caso contrário.
- Acrescente à função ronda um parâmetro que indica qual a regra a utilizar ('s17' ou 'h17').

Entrega do Projeto

Deve ser feita no Moodle.

Cada grupo deve submeter um único ficheiro com a sua solução. O nome do ficheiro deve ser BJXXX.py, onde XXX é o número do grupo.

A submissão deve ser feita por um (e apenas um) qualquer membro do grupo antes das 20h do dia 20 de Dezembro de 2017.

Até ao limite do prazo, podem entregar nova versão, a qual substitui a anterior. Nesse caso, deverá ser o mesmo aluno a fazer a submissão.

Lembre-se de comentar adequadamente o seu programa de forma a facilitar a compreensão do código. Inclua as docstrings de todas as funções que desenvolver.

Lembre-se ainda que o seu programa vai ser não só executado por um computador, mas também lido e avaliado por pessoas. Convém que os nomes das variáveis e a organização do programa sejam pensados para facilitar essa tarefa, como é devido a qualquer programa, mesmo que não fosse para avaliação. Na dúvida, consulte um docente.

Teste isoladamente as suas funções antes de as integrar no programa. Só então deve testar o programa. Tente que os seus testes cubram a grande variedade de casos que podem ocorrer.

Logo nas primeiras linhas do ficheiro tem de estar incluído (obviamente, comentado) o número do grupo e o nome e número de todos os elementos do grupo. Por exemplo:

```
#####################################
###########Projecto de Programação 1
#### GRUPO 888
#### Aureliano Buendía fc88008
#### Meursault fc80880
#### Euchrid Eucrow fc88088
#####################################
```

Aproveite para indicar aqui se entrega o projeto apenas com a programação do jogo ou se programou também as extensões. Deste modo:

```
#####################################
########## JOGO -- sim
########## HINT -- sim
########## REGRAS -- sim
#####################################
```

A cotação do projeto será 3 valores para o caso-base e 0.5 valores para cada uma das extensões, num total de 4 valores.

A correção e elegância do código, bem como o uso de boas práticas de programação serão valorizadas na avaliação.

Evitem a todo o custo copiar código dos projetos uns dos outros. Não é ético, é ilegal, vão ser apanhados de certeza, vão chumbar a esta cadeira e ainda pode haver consequências legais muito graves.