



DEEC
DEPARTAMENTO DE ENGENHARIA
ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Mestrado Integrado em Engenharia
Electrotécnica e de Computadores
(MEEC)

PROGRAMAÇÃO

ENUNCIADO DO PROJECTO



POKER *ISTARS*

Versão pública 1.0 (02/Março/2020)

2019/2020
2º Semestre

Conteúdo

1	Introdução	2
2	O problema – POKER<i>ist</i>ARS	2
3	O programa “POKER<i>ist</i>ARS”	6
3.1	Execução do programa	6
3.1.1	Modo linha de comando	7
3.1.2	Modo ficheiro de baralho	8
3.1.3	Modo ficheiro de baralho extra	10
3.1.4	Modo ficheiro de “shuffling”	12
3.1.5	Dados finais	13
3.2	Formato de entrada	13
3.3	Formato de saída	15
4	Primeira fase de submissões	17
5	Avaliação do Projecto	18
5.1	Funcionamento	19
5.2	Código	20
5.3	Relatório	20
5.4	Critérios de Avaliação	20
6	Código de Honestidade Académica	21

1 Introdução

O trabalho que se descreve neste enunciado possui duas componentes, que correspondem às duas fases de avaliação de projecto para a disciplina de Programação. A descrição geral do projecto que se propõe diz respeito ao trabalho a desenvolver para a última fase de avaliação. O trabalho a realizar para a primeira fase de avaliação assenta no mesmo problema, mas consiste no desenvolvimento de algumas funcionalidades que, apesar de não determinarem em absoluto a solução final, podem ser usadas posteriormente para ajudar na sua resolução. Assim, os alunos deverão encarar a primeira fase de avaliação como uma primeira etapa no trabalho de concepção e desenvolvimento da sua solução final.

A entrega do código fonte em ambas as fases é feita através de um sistema automático de submissão que verificará a sua correcção e testará a execução do programa em algumas instâncias do problema.

2 O problema – POKERistARS

Neste projecto pretende-se desenvolver um programa que seja capaz identificar o valor de uma mão de poker em diferentes modos de jogo. Uma mão de poker é constituída por 5 cartas de um baralho de 52 cartas de jogar. Dado um conjunto de 5 cartas torna-se necessário saber qual o seu valor, existindo um total de 10 configurações base, aqui descritas por ordem crescente de valor:

1. “High card” (ou “High hand”) – que se obtém quando todas as cartas são diferentes na pontuação, não sendo os cinco valores consecutivos, e não sejam as cinco do mesmo naipe;
2. “Pair” – que se obtém com duas cartas de igual valor, sendo as restantes diferentes destas e entre si;
3. “Two pair” – que se obtém com dois conjuntos de duas cartas de igual valor, diferentes entre si e em que a quinta carta difere de qualquer dos dois pares;
4. “Three of a kind” – que se obtém com 3 cartas de igual valor, sendo as restantes 2 diferentes destas e entre si;
5. “Straight” – que se obtém com 5 cartas de valor consecutivo, sequência, que não sejam todas do mesmo naipe;
6. “Flush” – que se obtém com 5 cartas do mesmo naipe que não sejam uma sequência;
7. “Full house” – que se obtém com 3 cartas de um valor e 2 de outro;
8. “Four of a kind” – que se obtém com todas as 4 cartas de igual valor dos 4 naipes;
9. “Straight flush” – quando é uma sequência do mesmo naipe (*i.e.*, 7, 6, 5, 4, 3);
10. “Royal flush” – é um “straight flush” com as 5 cartas mais altas: ás, rei, dama, valete e 10;

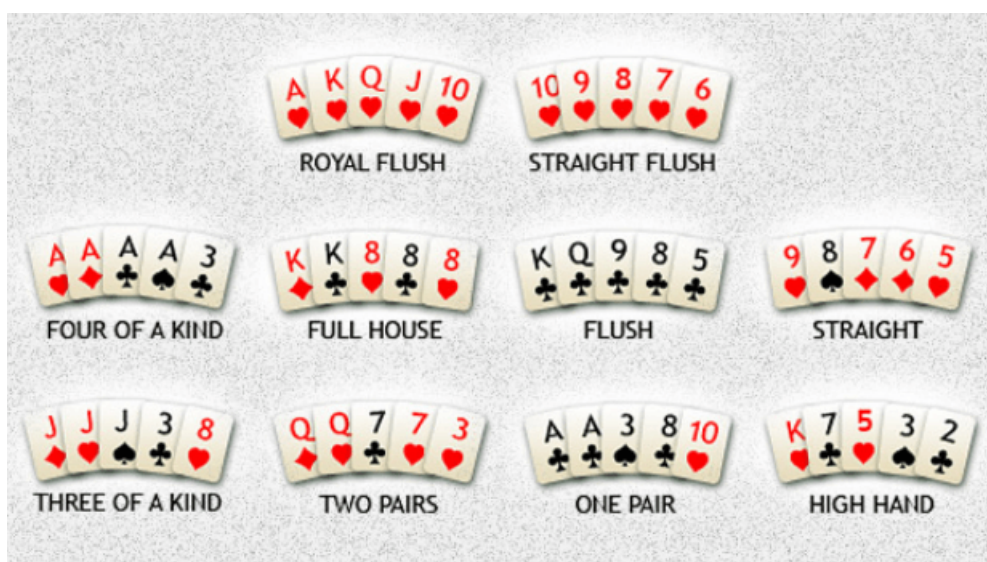


Figura 1: Exemplos das 10 configurações possíveis.

Na Figura 1 apresenta-se uma ilustração com exemplos das 10 configurações acima descritas.

Cada carta pode ser representada por dois caracteres: o primeiro carácter representa o valor; o segundo carácter representa o naipe. Os valores possíveis são: A, 2, 3, 4, 5, 6, 7, 8, 9, T, J, Q, K. Os naipes são: C(opas), E(spadas), O(uros), P(aus).

Quando se confrontam duas mãos de poker vence a que tiver o valor mais alto daquela lista. Por exemplo, “Two pair” vence sempre “Pair” quaisquer que sejam as cartas do “Pair”. Num jogo de poker cada jogador recebe duas cartas que mantém escondidas e na mesa são colocadas cinco cartas que vão sendo viradas para cima de acordo com a seguinte sequência: primeiro entra o “flop”, que é um conjunto de três cartas viradas para cima; o “turn” em que se mostra a quarta carta virada para cima; finalmente, o “river” em que se apresenta a quinta e última carta virada para cima. Cada jogador que decida ir a jogo, e nele permanecer à medida que se vão mostrando as cinco cartas da mesa, e se fazem as rondas de apostas, tem como mão o melhor conjunto de cinco cartas das sete que vê: as duas que recebeu em conjunto com as cinco da mesa. Assim, é possível que diferentes jogadores possuam mãos de igual valor, e é possível que algumas das cartas que compõem a melhor mão de cada um sejam comuns (há sempre, pelo menos, uma carta comum em quaisquer duas mãos que resultem deste processo de distribuição de cartas), por serem as da mesa. No limite, a melhor mão pode ser constituída pelas cinco cartas da mesa, caso em que ninguém vence, declarando-se empate entre os jogadores que permanecem na mesa no final.

Num confronto de duas mãos, se ambas correspondem ao mesmo valor (de 1 a 10), vence a que tiver as cartas mais altas. Ver descrição abaixo:

- se ambas as mãos forem “High card”, desempata-se da maior carta para a menor até haver diferença, vencendo a mão com a mais alta carta que desempata;
 - a melhor “High card” possível é, por exemplo, AE KP QP JC 90¹;
 - a pior “High card” possível é, por exemplo, 70 5C 40 3C 20;

¹Se aqui estivesse um dez em vez do nove a mão seria uma sequência de ás.

- se uma mão for JE 9P 6P 4C 3O e a segunda for JP 9O 6E 5C 2E, vence a segunda mão porque ambas as mãos têm as três mais altas cartas de igual valor, mas a quarta carta da segunda mão é o cinco de copas e a quarta carta da primeira é um quatro de copas, apesar de a quinta carta da primeira mão ser mais alta que a quinta da segunda.
- se ambas as mãos tiverem “Pair” ganha o par de maior valor, mas se for do mesmo valor, usam-se as restantes três cartas para desempatar, vencendo a mão com a mais alta carta que desempata;
 - o melhor “Pair” possível é, por exemplo, AE AO KE QE JC;
 - o pior “Pair” possível é, por exemplo, 5P 4P 3O 2C 2E;
 - se uma mão for TO TC 9O 8C 5O e a segunda for TP TC 9O 7E 6E, vence a primeira, porque o par é de igual valor em ambas e a terceira carta também tem o mesmo valor mas a quarta carta da primeira mão é o oito de copas superior ao sete de espadas da segunda mão.
- num “Two Pair” ganha a mão que tiver o par de maior valor mais alto ou o segundo par mais alto quando o primeiro for igual ou vence a mão em que a quinta carta é mais alta, quando os dois pares são de igual valor;
 - o melhor “Two Pair” possível é, por exemplo, AE AP KP KO QC;
 - o pior “Two Pair” possível é, por exemplo, 4O 3C 3O 2C 2O
 - se uma mão for JC JO 7E 7C 5P e a segunda for JC JO 8E 8C 5P, vence a segunda porque o seu segundo par é de maior valor que o segundo par da primeira mão.
- se ambas as mãos forem “Three of a kind”, vence o trio de maior valor, mas se os dois trios forem de igual valor, vence a mão com a mais alta carta que desempata comparando as restantes duas cartas;
 - o melhor “Three of a kind” possível é, por exemplo, AC AO AE KC QP;
 - o pior “Three of a kind” possível é, por exemplo, 4C 3E 2O 2C 2P;
 - se uma mão for JC 8O 7C 7E 7P e a segunda for JC 9P 7C 7E 7O, vence a segunda mão porque o nove de paus bate o oito de ouros da primeira mão.
- se ambas as mãos forem “Straight” vence a sequência que tiver a carta mais alta;
 - a melhor “Straight” possível é, por exemplo, AC KO QE JC TP;
 - a pior “Straight” possível é, por exemplo, 5C 4O 3E 2C AP;
- se ambas as mãos forem “Flush” vence a mão que tiver a mais alta das cinco cartas que desempata;
 - o melhor “Flush” possível é, por exemplo, AC KC QC JC 9C²;
 - o pior “Flush” possível é, por exemplo, 7O 5O 4O 3O 2O

²Se aqui estivesse o dez de copas, em vez do nove, a mão seria um “Royal flush”.

- se uma mão for KP TP 9P 7P 6P e a segunda for KP TP 9P 8P 4P, vence a segunda porque, apesar de as três cartas mais altas serem as mesmas, a quarta carta da segunda mão bate a quarta carta da primeira.
- se ambas as mãos forem “Full house” ganha a mão que tiver o trio mais alto ou o par mais alto quando os trios forem iguais;
 - o melhor “Full House” possível é, por exemplo, AC AE AO KC KP;
 - o pior “Full House” possível é, por exemplo, 3C 3O 2E 2C 2P;
- se ambas as mãos forem “Four of a kind” vence quem tiver as quatro cartas de valor mais alto, e se houver empate vence quem tiver a quinta carta mais alta;
 - o melhor “Four of a kind” possível é, por exemplo, AC AE AO AP KO;
 - o pior “Four of a kind” possível é, por exemplo, 3C 2C 2E 2O 2P;
- se ambas as mãos forem “Straight flush”, vence a mão com a mais alta carta da sequência;
 - o melhor “Straight flush” possível é, por exemplo, KO QO JO TO 9O;
 - o pior “Straight flush” possível é, por exemplo, 5P 4P 3P 2P AP;
- se ambas as mãos forem “Royal flush” nenhuma é vencedora.

Para todos os casos acima descritos é possível que não haja mão vencedora, declarando-se empate – neste caso usa-se o termo “**split**” para indicar que o pote em disputa é dividido por ambas as mãos. De todos os valores, o ás é a única carta que pode ser a mais alta de uma sequência (de ás a dez) ou a mais baixa (de 5 a ás). Apenas em sequências de 5 como carta mais alta o ás vale como 1. Em todas as restantes mãos que o tenham, o ás é a carta mais alta.

O que se pretende neste projecto é desenvolver uma aplicação em linguagem C que seja capaz de resolver automaticamente diferentes problemas inspirados nos princípios básicos da determinação do valor da mão de poker de um dado conjunto de 5 cartas.

Nas secções seguintes descreve-se os diferentes formatos de utilização do programa a desenvolver; nos casos aplicáveis, quais as configurações dos ficheiros de entrada e saída; quais as regras de avaliação; e faz-se referência ao *Código de Conduta Académica*, que se espera ser zelosamente cumprido por todos os alunos que se submetam a esta avaliação.

Aconselha-se todos os alunos a lerem com a maior atenção todos os aspectos aqui descritos. Será obrigatória a adesão sem variações nem tonalidades a todas as especificações aqui descritas. A falha em cumprir alguma(s) especificação(ões) e/ou regra(s) constante(s) neste enunciado acarretará necessariamente alguma forma de desvalorização do trabalho apresentado.

Por esta razão, tão cedo quanto possível e para evitar contratempos tardios, deverão os alunos esclarecer com o corpo docente qualquer aspecto que esteja menos claro neste enunciado, ou qualquer dúvida que surja após leitura atenta e cuidada.

3 O programa “POKER*ist*ARS”

O programa a desenvolver deverá ser capaz de interpretar mãos de poker em diferentes cenários e produzir soluções para cada um deles.

3.1 Execução do programa

Haverá duas fases de submissão do projecto. O que se descreve nas próximas secções diz respeito às especificações da versão final do projecto. Na secção 4 detalham-se as especificações relativas à primeira fase de submissões.

O programa POKER*ist*ARS deverá ser invocado na linha de comandos da seguinte forma:

```
prog$ ./pokeristars [OPTION] ... [FILE]
```

`pokeristars` designa o nome do ficheiro executável contendo o programa POKER*ist*ARS;

[OPTION] designa a possibilidade de o programa ser invocado com diferentes opções de funcionamento;

[FILE] designa a possibilidade de o programa ter de operar sobre algum ficheiro;

As opções de funcionamento, identificadas sempre como strings começadas com o carácter ‘-’, que deverão estar operacionais são:

- c – identifica modo linha de comando;
- d1 – identifica modo ficheiro de baralho #1;
- d2 – identifica modo ficheiro de baralho #2;
- d3 – identifica modo ficheiro de baralho #3;
- d4 – identifica modo ficheiro de baralho #4;
- dx – identifica modo ficheiro de baralho extra;
- s1 – identifica modo ficheiro de *shuffling* #1;
- o – identifica modo de escrita de resultados em ficheiro.

Por cada invocação do programa apenas uma das 7 primeiras opções acima indicadas poderá ser usada. A opção -o, para escrever a saída em ficheiro, não é válida quando se usa a opção -c. Ou seja, o modo linha de comando produz sempre saída para `stdout`.

3.1.1 Modo linha de comando

A opção `-c` significa que as cartas a analisar serão indicadas na linha de comando. Após a opção dever-se-á apresentar 5, 7, 9 ou 10 strings de dois caracteres cada uma. Cada string identifica uma carta.

Quando o número de strings for 5 pretende-se identificar qual a mão de poker presente nas 5 cartas, devendo o programa escrever para `stdout` um número inteiro de acordo com a mão identificada: 1, 2, ..., 10. O inteiro 1 corresponde a “High card” e o inteiro 10 corresponde a “Royal flush”.

Quando o número de strings for 10 pretende-se interpretá-las como duas mãos de dois jogadores. Neste caso o programa deverá escrever como resultado um de 3 inteiros: 0 para empate, 1 indica que as primeiras 5 cartas vencem as segundas 5, 2 indica o contrário.

Quando o número de strings for 7 pretende-se interpretá-las como 2 cartas na mão de um jogador solitário seguidas de 5 cartas da mesa. O objectivo neste caso é indicar qual o melhor conjunto de 5 cartas que se pode construir com as 7 cartas apresentadas. Neste caso o programa deverá produzir como resultado as 5 cartas seguidas do inteiro da mão a que correspondem. Por exemplo, se as 7 cartas fossem KE 30 70 5E 6C 7P TC o programa deveria escrever KE TC 70 7P 6C 2, que identifica a mão como sendo um “Pair”. A mão tem um par de setes e as restantes cartas são as mais altas de entre as 5 que não formam o par e é apresentada por ordem decrescente de valor e alfabética crescente quando os valores são iguais.

Quando o número de strings for 9 pretende-se interpretá-las como 2 cartas na mão do primeiro jogador, seguidas de 2 cartas na mão do segundo, após o que se indicam as 5 cartas na mesa. O objectivo neste caso é escrever as duas mãos de cada jogador e o resultado do embate (ver exemplo abaixo). A diferença neste caso prende-se com a necessidade de determinar qual a melhor mão de 5 cartas que cada jogador pode compôr associando as suas 2 cartas com as 5 cartas da mesa. O programa deverá indicar qual a mão de cada jogador, no mesmo formato do caso de apenas 7 cartas e no final deverá acrescentar mais um inteiro que deverá tomar um de três valores, tal como no caso em que existem 10 cartas.

Quando, em modo linha de comando, o número de strings identificadoras das cartas for diferente de 5, 7, 9 ou 10, o programa deverá produzir -1 como resultado, indicando dessa forma que a entrada é ilegal. O mesmo deverá fazer se alguma das strings for ilegal. Ou seja, caso tenha comprimento diferente de 2 e/ou algum carácter na string não fizer parte do alfabeto acima definido na ordem indicada. Por exemplo, 2E identifica o dois de espadas mas E2 é uma carta ilegal, assim como 1P, 6e ou 10C. Também é ilegal um conjunto de cartas em que alguma seja repetida, excepto exclusivamente no caso em que se processam 10 cartas. Neste caso, cada uma das mãos pode ter cartas que a outra também possua até ao valor máximo de cartas (5), mas cada uma das duas mãos não pode ter cartas repetidas.

Abaixo apresentam-se alguns exemplos de utilização do programa em modo linha de comando:

```
prog$ ./pokeristars -c 3E 7P 6P 4C 50
```

```
5
```

```
prog$ ./pokeristars -c 3E 7P 6P 4C
```

```
-1
```



```
prog$ ./pokeristars -c 3E 7P 6P 4C 5D 7C AP 4E 4P
```

```
AP 7C 7P 4E 4P 3 AP 7C 4C 4E 4P 4 2
```

```
prog$ ./pokeristars -c 3E 7P 6P 4C 5D 7C 3E
```

```
-1
```

No primeiro exemplo a mão apresentada corresponde a uma sequência, com o sete de paus como carta mais alta, que ocupa o quinto lugar na lista; no segundo exemplo falta uma carta; no terceiro exemplo a melhor mão do primeiro jogador é AP 7C 7P 4E 4P que corresponde a “Two pair” com setes e quatros, mas a melhor mão do segundo jogador é AP 7C 4C 4E 4P que corresponde a “Three of a kind” de quatros. Note-se que neste último exemplo ambos os jogadores usaram 4 das 5 cartas da mesa para compôr a sua melhor mão. No quarto exemplo a carta três de espadas surge duas vezes: uma na mão e outra na mesa.

3.1.2 Modo ficheiro de baralho

A opção `-di`, com $i = 1, 2, 3, 4$, deverá ser seguida de um ficheiro `<nome>.deck`, em que `<nome>` pode ser qualquer string de qualquer tamanho, mas a extensão não é opcional. É obrigatório que o ficheiro passado como argumento possua a extensão indicada.

O ficheiro passado como argumento possui um ou mais conjuntos de 52 strings representando um ou mais baralhos completos aleatoriamente ordenados.

Cada baralho presente no ficheiro deverá ser processado de acordo com uma de quatro variantes possíveis, conforme o valor de i :

1. O baralho deverá ser processado em grupos de 5 cartas;
2. O baralho deverá ser processado em grupos de 7 cartas;
3. O baralho deverá ser processado em grupos de 9 cartas;
4. O baralho deverá ser processado em grupos de 10 cartas;

Em cada uma das quatro variantes o programa terá de ler um conjunto de cartas necessárias para a variante em processamento. Se não conseguir ler um conjunto completo de cartas termina o processamento.

Como o número de cartas de cada baralho não é múltiplo de nenhum dos números indicados acima, as cartas sobrantes não devem ser processadas. Ou seja, não devem ser tratadas como um conjunto incompleto. Assim, quando o número de cartas ainda não usadas não chega para constituir um conjunto completo, o programa deverá passar ao baralho seguinte, se houver, saltando por cima das cartas não usadas.

Também se pode dar o caso de alguma das cartas no ficheiro ser ilegal. Qualquer conjunto de cartas que possua uma carta ilegal ou indevidamente repetida tem sempre como resultado `-1` para esse conjunto, mas o processamento de um ficheiro só termina quando não houver mais cartas para ler.

O output para cada uma das quatro variantes segue o perfil definido para o modo linha de comando.

Abaixo apresenta-se dois exemplos possíveis de invocação deste modo.

```
prog$ ./pokeristars -d1 alguns.deck
```

3

4

...

-1

...

7

...

```
prog$ ./pokeristars -d3 alguns.deck -o resultados.out
```

```
prog$
```

No primeiro exemplo, processa-se o ficheiro de nome `alguns.deck` na variante 1 e escreve-se o resultado do processamento de cada conjunto de 5 cartas lido para `stdout`, um por linha. No segundo exemplo processa-se o mesmo ficheiro na variante 3 e o resultado para cada conjunto de 9 cartas é escrito numa linha do ficheiro de nome `resultados.out`.

Sempre que se termina um baralho, deverá ser produzida uma linha em branco como separador do processamento do baralho actual do baralho seguinte, se o houver, ou da apresentação das mesmas estatísticas já referidas no modo ficheiro de mãos.

No final da execução, após concluir o processamento de todos os baralhos, o programa deverá terminar com a escrita de estatísticas do ficheiro processado. Na secção 3.1.5 descreve-se qual a informação que deve ser recolhida para exibição no final da execução do programa. Só se apresenta um conjunto de dados estatísticos por ficheiro e sempre no final do processamento de todos os baralhos que ele contenha. Os dados a apresentar são sempre relativos a todo o ficheiro.

Em todos os modos que tenham ficheiro de entrada, o output só pode ser escrito para ficheiro de saída se este for dado como argumento após a opção `-o`. Na falta de tal especificação, o output será sempre enviado para `stdout`, como foi descrito nesta secção. A opção de `-o` só pode ser usada como terceiro argumento e apenas se o primeiro for diferente de `-c`. Eis o que deverá acontecer com erros de utilização da opção `-o`:

```
prog$ ./pokeristars -o resultado.out -d1 alguns.deck
```

```
-1
```

```
prog$ ./pokeristars -d2 outros.deck -o
```

```
-1
```

```
prog$ ./pokeristars -c AE 30 K0 KP 6C -o guarda.aqui
```

```
-1
```

```
prog$
```

No primeiro exemplo a opção `-o` aparece onde deveria estar a opção de processamento (uma das 7 primeiras da lista acima). No segundo exemplo a opção `-o` está no sítio certo mas falta o nome do ficheiro de saída. No terceiro há erro porque o modo de linha de comando não pode aceitar a opção de escrita em ficheiro.

3.1.3 Modo ficheiro de baralho extra

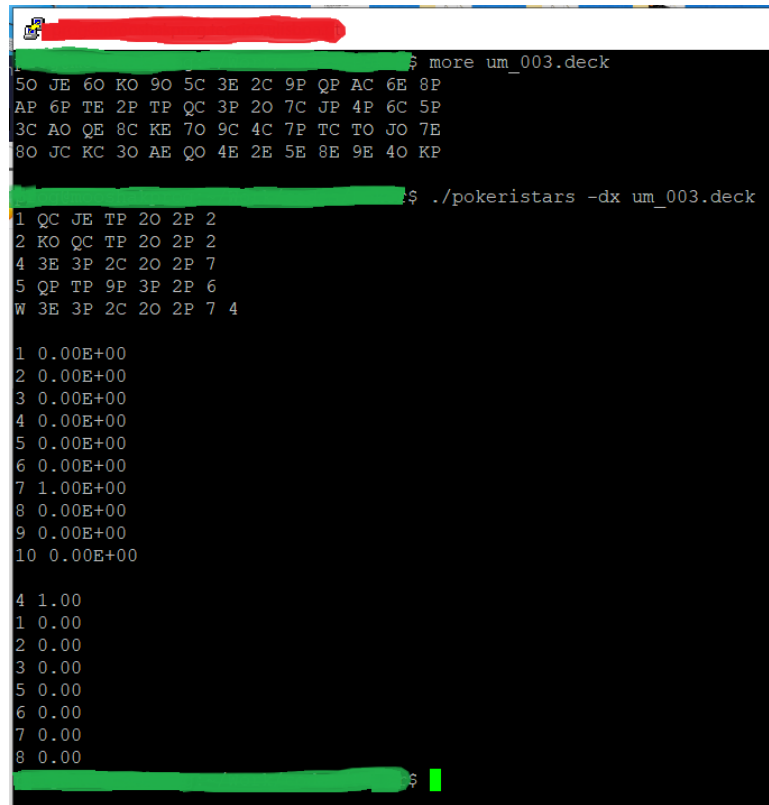
A opção `-dx` deverá ser seguida de um ficheiro `<nome>.deck`, em que `<nome>` pode ser qualquer string de qualquer tamanho, mas a extensão não é opcional. É obrigatório que o ficheiro passado como argumento possua a extensão indicada.

Este é o modo de funcionamento que mais de perto se aproxima de um torneio de poker. Cada ficheiro de entrada possui vários baralhos completos, mas cada baralho apenas serve para se jogar uma ronda em que intervêm, no máximo, oito jogadores, numerados de 1 a 8. O modo de funcionamento é o seguinte:

1. No início todos os oito jogadores estão activos;
2. Cada baralho só pode ser usado numa ronda, pelo que as cartas não usadas numa ronda deverão ser ignoradas na ronda seguinte, que usa um baralho novo;
3. Em cada ronda, cada jogador activo recebe duas cartas para a sua mão. As primeiras duas para o jogador activo de menor número, as seguintes para o jogador activo seguinte na numeração, etc.;
4. Em seguida são mostradas de uma só vez as cinco cartas da mesa;
5. A partir deste momento é possível determinar o valor da mão de poker que cada jogador possui, agregando às suas duas as cinco que estão na mesa;
6. O jogador i é obrigado a ir a jogo se a sua mão for igual ou superior, em valor, a i ou se fez “**fold**” consecutivamente nas últimas quatro rondas; por exemplo, o jogador 3 não pode fazer “**fold**” se a sua mão for igual ou superior a “Two pair”, porque essa é a mão de valor 3;
7. Só depois de se saber quantos jogadores ficam no jogo se pode determinar quem ganha a ronda e regista-se a mão vencedora, podendo haver empate(s);
8. Uma vitória atribui um ponto ao vencedor, e num empate atribui-se a cada jogador empatado o ponto dividido pelo número de empatados;
9. Jogadores inactivos ou que façam “**fold**” não recebem qualquer fracção de ponto da ronda;
10. Nenhum jogador pode ir a jogo mais que duas vezes consecutivas, ficando inactivo durante uma só ronda;
11. Ao fim de duas idas a jogo, sem “**fold**”, o jogador não recebe o par de cartas na ronda seguinte, por estar inactivo;
12. Nenhum jogador pode fazer “**fold**” mais que quatro vezes consecutivas;
13. Ao fim de quatro “**folds**” consecutivos o jogador é obrigado a ir a jogo com a mão que receber, qualquer que ela seja;
14. Sempre que um jogador não faz “**fold**” conta para a sequência de idas a jogo e limpa a sequência de “**folds**”;
15. Sempre que um jogador faz “**fold**” conta para a sequência de “**folds**” e limpa a sequência de idas a jogo;

16. O facto de um jogador não receber cartas, por estar inactivo, não conta como “fold”, nem como ida a jogo, sem “fold”.

Em cada ronda, o programa deverá escrever uma linha por cada jogador que for a jogo com a indicação do jogador e da mão que possui. No final escreve a linha com a mão vencedora e que jogadores a tiveram, por ordem crescente do seu identificador.



```
$ more um_003.deck
50 JE 60 KO 90 5C 3E 2C 9P QP AC 6E 8P
AP 6P TE 2P TP QC 3P 2O 7C JP 4P 6C 5P
3C AO QE 8C KE 7O 9C 4C 7P TC TO JO 7E
8O JC KC 3O AE QO 4E 2E 5E 8E 9E 4O KP

$ ./pokerstars -dx um_003.deck
1 QC JE TP 2O 2P 2
2 KO QC TP 2O 2P 2
4 3E 3P 2C 2O 2P 7
5 QP TP 9P 3P 2P 6
W 3E 3P 2C 2O 2P 7 4

1 0.00E+00
2 0.00E+00
3 0.00E+00
4 0.00E+00
5 0.00E+00
6 0.00E+00
7 1.00E+00
8 0.00E+00
9 0.00E+00
10 0.00E+00

4 1.00
1 0.00
2 0.00
3 0.00
5 0.00
6 0.00
7 0.00
8 0.00
```

Figura 2: Exemplo de execução do modo `-dx` num ficheiro com 1 baralho.

Na Figura 2 ilustra-se o formato de saída para uma execução de apenas um baralho. Antes da invocação do programa mostra-se o conteúdo do ficheiro de entrada. Neste exemplo todos os oito jogadores estão activos, mas apenas os jogadores 1, 2, 4 e 5 vão a jogo. O jogador 1 tem “Pair” (valor $2 \geq 1$), assim como o jogador 2. O jogador 4 tem “Full House” (valor $7 \geq 4$) e o jogador 5 tem “Flush” (valor $6 \geq 5$). A mão vencedora, assinalada com o carácter ‘W’ é a mão do jogador 4.

Uma situação de empate acontece normalmente com cartas diferentes. Para garantir que o output é sempre o mesmo, em situação de empate dever-se-á escrever a mão possuída pelo jogador de menor índice identificador, de todos os que tiverem empatado.

Se o ficheiro de entrada possuir mais que um baralho, o ficheiro de saída possuirá a ronda completa de cada baralho: jogadores que vão a jogo, as cartas da sua mão e o valor, seguido da mão vencedora e jogadores que a possuíram, do menor para o maior identificador. Duas rondas consecutivas deverão ser separadas entre si de uma linha em branco. Sublinha-se que se pode dar o caso de todos os jogadores activos fazerem “fold”. Nessa situação não é produzida qualquer saída, porque não há jogadores em jogo e não há mão vencedora. Para efeitos estatísticos só se contam mãos e rondas que tenham tido, pelo menos, um dos jogadores activos que não tenha feito “fold”.

Na remota, mas não improvável, hipótese de todos os oito jogadores estarem inactivos, o programa deverá saltar o baralho que seria usado, representando dessa forma que houve uma ronda. No entanto, nessa ronda não há distribuição de cartas nem atribuição de pontos.

No final de todas as rondas, quando já não houver mais baralho para jogar, o programa deverá produzir as estatísticas relativas às frequências relativas das mãos ganhadoras, após o que apresenta o quadro classificativo dos jogadores por ordem decrescente da pontuação obtida. Na secção 3.3 descreve-se o formato de produção desta informação.

Este modo de funcionamento é o mais interessante, mas também o mais complexo de todos os modos deste projecto. Face a isto não é expectável que todos os grupos sejam capazes de o implementar correctamente. Dada a sua complexidade, o número de testes deste tipo será limitado a um intervalo percentual de todos os testes da fase final que se traduza numa contribuição para a nota final de 1.0 a 1.5 valores.

Sugere-se que os grupos dêem prioridade à implementação dos restantes modos antes de se aventurarem na implementação deste.

3.1.4 Modo ficheiro de “shuffling”

A opção `-s1` deverá ser seguida de um ficheiro `<nome>.shuffle`, em que `<nome>` pode ser qualquer string de qualquer tamanho, mas a extensão não é opcional. É obrigatório que o ficheiro passado como argumento possua a extensão indicada.

O ficheiro passado como argumento possui uma sequência, de tamanho indeterminado, de inteiros só constituída pelos dígitos 1, 2 e 3. Após a sequência de inteiros apresentam-se 52 strings com dois caracteres, representando um baralho completo. Um qualquer ficheiro de extensão `shuffle` pode conter mais que um conjunto deste tipo: sequência de dígitos seguida de um baralho completo.

A sequência de inteiros indica o modo de mistura das cartas que a seguem. Quando o inteiro vale 1, o baralho é partido em duas metades iguais (26 cartas cada), após o que se mistura alternando cartas da primeira metade com cartas da segunda, uma carta de cada vez. Quando o inteiro vale 2, o baralho é partido em duas metades iguais e, após a inversão da ordem das cartas apenas da segunda metade, aplica-se o mesmo procedimento de mistura, alternando entre uma carta do primeiro e uma carta do segundo conjunto. O inteiro 3 serve para a operação que, normalmente, se designa por corte, i.e., a primeira metade do baralho é passada, sem alteração da posição das suas cartas, para o final, ficando atrás da segunda metade, que também não é alterada.

O programa deverá produzir para `stdout` o baralho resultante da sequência de misturas. A utilização da opção `-o` faz com que o resultado seja enviado para o ficheiro indicado após esta opção. Cada baralho baralhado deve ser escrito no formato de 13 cartas por linha, cada carta separada da anterior com apenas um espaço em branco. Havendo mais que um conjunto por ficheiro, cada conjunto deverá ser apresentado separado por uma linha em branco do que o antecede.

Na implementação deste modo é obrigatório que os grupos façam uso de listas para representação do baralho que se pretende misturar. Sendo possível resolver este problema sem o recurso a listas, a sua não utilização acarreta penalização na avaliação global do projecto submetido.

3.1.5 Dados finais

Nos modos de baralho anteriormente descritos é necessário que o programa produza como saída final, depois de processar os vários conjuntos de cartas existentes no ficheiro de entrada, um conjunto de dados estatísticos e de síntese.

- Dados estatísticos – dizem respeito à frequência relativa de cada uma das 10 mãos possíveis; também se deve incluir nesta informação a frequência relativa das mãos ilegais (que tenham tido -1);
- Dados de síntese – dizem respeito ao número total de empates e vitórias de cada jogador; quando há mãos ilegais deve registar-se um empate.

Se, por hipótese, tiverem sido produzidas 100 mãos no total (em modo de dois jogadores deve contabilizar-se sempre duas mãos em cada conjunto que tenha uma ou mais cartas ilegais), e 3 for o total final de mãos ilegais, então a frequência relativa observada para este tipo de mão foi de 3%. Se tiverem sido registadas 45 mãos com “High card”, a frequência relativa deste tipo de mão será 45%.

Os dados estatísticos são produzidos para qualquer das quatro variantes do modo `-di`. Os dados de síntese são produzidos apenas para as variantes 3 e 4.

A contagem das mãos ilegais para as variantes 1 e 2 é trivial. Havendo uma carta ilegal o conjunto todo conta como uma mão ilegal. Nas variantes 3 e 4, qualquer que seja a localização da carta, ou cartas, ilegal, o conjunto todo conta como duas mãos ilegais.

Uma carta é ilegal se não fizer parte do baralho, como foi definido, ou se estiver repetida nas variantes 1, 2 e 3. Em variante 4 podem haver cartas repetidas, mas se algum dos dois jogadores tiver cartas repetidas essas cartas são ilegais.

Na secção 3.3 detalha-se qual o formato de saída da informação que aqui se descreveu.

3.2 Formato de entrada

Para este projecto existirão dois tipos de ficheiro de entrada: ficheiros de extensão `deck` e ficheiros de extensão `shuffle`. Qualquer outro ficheiro que seja passado como ficheiro de leitura e não possua uma destas duas extensões é ilegal, assim como é ilegal passar ficheiros incompatíveis na extensão com o modo pedido na primeira opção. Isto é, quando a opção for `-di` apenas se aceitam ficheiros de extensão `deck` e os ficheiros de extensão `shuffle` só são aceites para processamento se a opção for `-s1`.

Um ficheiro de extensão `deck` contém sequências de cartas que podem ser usadas numa das quatro variantes anteriormente definidas. Por exemplo, um ficheiro deste tipo poderia ter o seguinte conteúdo:

```
2P JE 90 6C 2C
Q0 T0 7E AP 9C
...
```

mas também poderia surgir do seguinte modo:

2P	JE		
	90		
6C		2C	
	Q0	TO	7E
			AP
9C			
...			

Sublinha-se aqui que os dados de cada problema não têm necessariamente que estar “arrumados” como se ilustra no primeiro exemplo. O segundo exemplo ilustra isso mesmo. Os procedimentos de leitura a adoptar pelos alunos deverão ser robustos a qualquer tipo de desarrumação, sendo que existirá sempre, pelo menos, um espaço em branco³ após cada carta.

Um ficheiro de extensão **deck** possui sempre, pelo menos, um baralho completo ordenado aleatoriamente, podendo possuir mais que um. Todos os baralhos presentes nestes ficheiros são sempre completos, ou seja há sempre 52 cartas. Ilustra-se um ficheiro de extensão **deck** abaixo onde se colocaram 13 cartas em cada linha apenas por economia de espaço, mas não existe qualquer garantia nem necessidade de que todos os ficheiros deste tipo tenham de estar assim construídos.

AE	2E	3E	4E	5E	6E	7E	8E	9E	TE	JE	QE	KE
AC	2C	3C	4C	5C	6C	7C	8C	9C	TC	JC	QC	KC
AP	2P	3P	4P	5P	6P	7P	8P	9P	TP	JP	QP	KP
A0	20	30	40	50	60	70	80	90	TO	JO	Q0	K0

Um ficheiro de extensão **shuffle** possui sempre, pelo menos, uma sequência de mistura e um baralho completo. Abaixo apresenta-se um possível ficheiro deste tipo, também apresentado em forma mais compacta apenas por economia de espaço.

1	1	1	1	2	2	1	3	1	1	2	2	2
AE	2E	3E	4E	5E	6E	7E	8E	9E	TE	JE	QE	KE
AC	2C	3C	4C	5C	6C	7C	8C	9C	TC	JC	QC	KC
AP	2P	3P	4P	5P	6P	7P	8P	9P	TP	JP	QP	KP
A0	20	30	40	50	60	70	80	90	TO	JO	Q0	K0

1	2											
AE	2E	3E	4E	5E	6E	7E	8E	9E	TE	JE	QE	KE
AC	2C	3C	4C	5C	6C	7C	8C	9C	TC	JC	QC	KC
AP	2P	3P	4P	5P	6P	7P	8P	9P	TP	JP	QP	KP
A0	20	30	40	50	60	70	80	90	TO	JO	Q0	K0

³Ou mudança de linha, ou tab.

3.3 Formato de saída

Excepto para as opções `-s1` e `-dx`, o formato de saída do programa é sempre o mesmo: um único inteiro de 1 a 10 por cada conjunto de cinco cartas, ou -1 se o conjunto tiver cartas ilegais; um único inteiro de 0 a 2 para conjuntos de dez cartas, ou -1 se o conjunto for ilegal; cinco strings representando a melhor mão, seguidas de um inteiro de 1 a 10 por cada conjunto de sete cartas, ou -1 se o conjunto for ilegal; cinco strings, seguidas de um inteiro de 1 a 10, mais cinco strings seguidas de um inteiro de 1 a 10 e ainda mais um inteiro de 0 a 2 para conjuntos de 9 cartas, ou -1 se o conjunto for ilegal.

O resultado de cada conjunto deve ocupar uma única linha e os espaços em branco só devem ser usados entre dados na mesma linha. Existem situações em que um conjunto de sete cartas produz a mesma mão de poker com cartas diferentes. Por exemplo, se as sete cartas forem 3E 7P 30 6P 4C 3P 50 a mão é “Straight” de 7 e existem três cartas de valor 3. Logo, qualquer das três cartas de valor 3 pode ser usada para compôr a mão. Para efeitos da avaliação, é necessário que os todos os programas produzam sempre a mesma saída quando os dados de entrada são iguais. Assim, sempre que haja escolha de cartas possível é obrigatório escolher a(s) carta(s) por ordem alfabética crescente do naipe. Ver exemplo abaixo:

```
prog$ ./pokeristars -c 3E 7P 30 6P 4C 3P 50
```

7P 6P 50 4C 3E 5 – saída correcta

```
prog$ ./pokeristars -c 3E 7P 30 6P 4C 3P 50
```

7P 6P 50 4C 30 5 – saída incorrecta

```
prog$ ./pokeristars -c 3E 4P 30 6P 4C 3P 40
```

4C 40 4P 3E 30 7 – saída correcta

```
prog$ ./pokeristars -c 3E 4P 30 6P 4C 3P 40
```

4C 40 4P 3E 3P 7 – saída incorrecta

Se não for especificado ficheiro de saída, ou se se estiver a usar a opção `-c`, todos os resultados são escritos em `stdout`.

Para o modo de ficheiro de baralho é necessário produzir dados estatísticos com o seguinte formato:

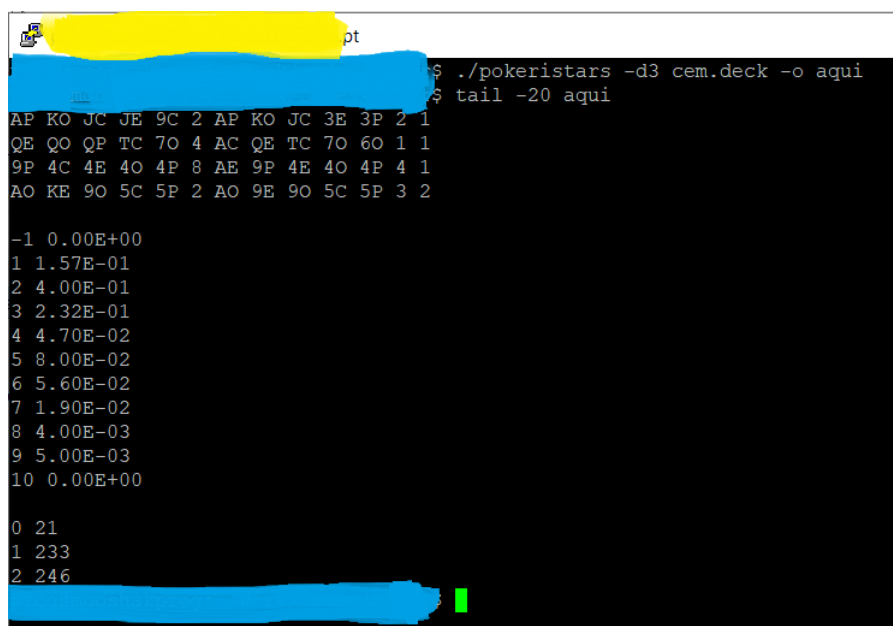
```
-1 2.21E-02
1 5.02E-01
2 3.01E-02
3 1.07E-02
...
...
10 0.00E00
```


Uma linha por tipo de mão, seguida da frequência relativa, apresentada com apenas 2 dígitos decimais e em notação científica. Entre o tipo de mão e a sua frequência relativa deverá estar apenas um espaço em branco.

Após estas 11 primeiras linhas e uma linha vazia, se a variante tiver envolvido dois jogadores, apresenta-se o total de empates (inclui as mãos ilegais), vitórias do jogador 1 e vitórias do jogador 2, com dois inteiros por linha separados entre si com apenas um espaço em branco. Em qualquer dos casos, o último número escrito deverá ser seguido de mudança de linha.

```
0 6
1 30
2 44
```

Nestes dois exemplos, se assumirmos pertencerem a um mesmo ficheiro, houve 2.21% de mãos ilegais, 50.2% de “High card”, 3.01% de “Pair”, 1.07% de “Two pair”, . . . , e 0% de “Royal flush”. Houve um total de 6 empates, 30 vitórias do jogador 1 e 44 vitórias do jogador 2. Na Figura 3 mostra-se parte da saída produzida para ficheiro.



```

$ ./pokeristars -d3 cem.deck -o aqui
$ tail -20 aqui
AP KO JC JE 9C 2 AP KO JC 3E 3P 2 1
QE QO QP TC 7O 4 AC QE TC 7O 6O 1 1
9P 4C 4E 4O 4P 8 AE 9P 4E 4O 4P 4 1
AO KE 9O 5C 5P 2 AO 9E 9O 5C 5P 3 2

-1 0.00E+00
1 1.57E-01
2 4.00E-01
3 2.32E-01
4 4.70E-02
5 8.00E-02
6 5.60E-02
7 1.90E-02
8 4.00E-03
9 5.00E-03
10 0.00E+00

0 21
1 233
2 246

```

Figura 3: Exemplo da saída em modo -d3 num ficheiro com 100 baralhos.

Para a opção -dx as estatísticas das mãos ganhadoras deverão seguir o mesmo formato das estatísticas anteriores, excepto pelo facto de não haver mãos ilegais, pelo que a linha a elas associada deverá ser omitida⁴. Após as estatísticas das mãos ganhadoras (uma mão de empate também conta como ganhadora, mesmo que tenha havido empate nos oito jogadores), deve-se produzir uma linha vazia e depois dessa escreve-se o quadro classificativo, como no exemplo abaixo, usando apenas e sempre dois dígitos decimais.

⁴Para o modo de baralho extra, pode-se assumir que se usarão sempre ficheiros contendo baralhos legítimos.

```
7 6.50
4 2.33
1 1.33
2 1.33
3 0.50
5 0.00
6 0.00
8 0.00
```

No exemplo acima o jogador 7 pode ter ganho 6 rondas e tido um empate com outro, mas também pode ter tido mais rondas de empate sempre com apenas um outro jogador. Os jogadores 5, 6 e 8 nunca tiveram mão ganhadora.

Quando os jogadores têm a mesma pontuação, deverão ser apresentados por ordem crescente do seu índice.

Quando a opção usada for `-s1` o output deverá ser o baralho resultante do processo de mistura. Cada baralho deverá ser escrito, seja em `stdout` seja em ficheiro, no formato de treze cartas por linha e se o ficheiro de entrada possuir mais que um baralho, dever-se-á colocar uma linha vazia a separar baralhos consecutivos. A última linha de saída deverá ser sempre vazia.

Todo e qualquer erro de invocação do programa obriga a que se produza sempre para `stdout` o resultado -1. Ao longo deste enunciado têm sido referidos alguns dos erros possíveis de invocação, faltando apenas indicar que quando se fornecem ficheiros de entrada para processamento, deverá essa invocação ser interpretada como contendo erro se o ficheiro não existir, possua ou não a extensão adequada para o modo solicitado. Também é erro de invocação usar alguma opção diferente das que aqui se descreveram.

Durante a fase de desenvolvimento do projecto é natural que os grupos avancem implementando os diferentes modos de funcionamento progressivamente. Assim, será boa ideia que, para modos ainda não implementados, o programa produza como saída o mesmo que produz quando existam erros de invocação. No entanto, podem os grupos optar por produzir como saída alguma outra informação diversa de -1. Por exemplo, escrevendo coisas como `Ainda não implementado`, mas na `“to do list”`., `Coming soon...` `to a theater near you.`, ou ainda 0.

Finalmente, falta indicar que a função `int main(...)` quando termina a sua execução tem de devolver o inteiro zero (`EXIT_SUCCESS`, seja por `return` ou por `exit(.)`). O site de submissões automáticas interpreta qualquer retorno diferente de zero como “Erro de Execução”, atribuindo 0 pontos a cada teste em que tal aconteça.

4 Primeira fase de submissões

Nesta secção explicitam-se os objectivos, especificações e funcionalidades que deverão estar operacionais na data da primeira fase de submissões.

O que os grupos deverão ter a funcionar correctamente aquando da primeira submissão é o modo de linha de comando completo como descrito na secção 3.1.1.

Para além disso deverá o programa ser robusto a erros de invocação. Isto é, apenas é aceitável o modo linha de comando, com opção `-c` e apenas para cinco, sete, nove ou dez strings, todas com dois caracteres cada, após a definição da opção.

Qualquer outro formato de invocação, com outra opção que não seja `-c` e que tenha um número de strings que não seja 5, 7, 9 nem 10 deverá produzir -1 como resultado. Também deverá produzir resultado -1 quando as strings passadas como argumento são ilegais ou ilegalmente repetidas, de acordo com as regras referidas no modo linha de comando.

Também a produção de resultados é sempre feita para `stdout` de acordo com o formato de saída já descrito anteriormente.

Os grupos podem e devem tentar ter mais funcionalidades operacionais à data da primeira fase de submissões. Quanto mais adiantado estiver o trabalho, melhor será para o cumprimento do prazo do projecto completo. No entanto, apenas se avaliarão de forma automática as funcionalidades que estão descritas nesta secção.

5 Avaliação do Projecto

O projecto está dimensionado para ser feito por grupos de dois alunos, não se aceitando grupos de dimensão superior nem inferior. Para os alunos que frequentam o laboratório, o grupo de projecto não tem de ser o mesmo do laboratório, mas é aconselhável que assim seja.

Do ponto de vista do planeamento, os alunos deverão ter em consideração que o tempo de execução e a memória usada serão tidos em conta na avaliação do projecto submetido. Por essas razões, a representação dos dados necessários à resolução dos problemas deverá ser criteriosamente escolhida tendo em conta o espaço necessário, mas também a sua adequação às operações necessárias sobre eles.

Serão admissíveis para avaliação versões do programa que não possuam todas as funcionalidades, seja no que à primeira fase de submissões diz respeito, como para a fase final. Naturalmente que um menor número de funcionalidades operacionais acarretará penalização na avaliação final.

O projecto será submetido a testes automáticos aquando da sua submissão electrónica. Isto significa que a saída do programa para `stdout` ou para ficheiro deve cumprir rigorosamente as regras especificadas, caso contrário os testes falham e o projecto será considerado não cumprindo as especificações, com a correspondente penalização na nota. Assim, chama-se a atenção que qualquer mensagem de debug que seja escrita juntamente com a saída do programa resultará na falha dos testes automáticos. Sugere-se que mensagens de debug sejam enviadas para `stderr`, ou sejam desactivadas no código submetido.

Quando os grupos de projecto estiverem constituídos, os alunos devem obrigatoriamente inscrever-se no sistema Fenix, no grupo de Projecto correspondente, que será criado oportunamente.

A avaliação do projecto decorre em quatro ou cinco instantes distintos. O primeiro instante coincide com a primeira submissão electrónica, onde os projectos serão avaliados automaticamente com base na sua capacidade de cumprir as especificações e funcionalidades definidas na Secção 4. Para esta fase existe apenas uma data limite de submissão (veja a Tabela 1) e não há qualquer entrega de relatório. Após a conclusão da primeira fase, cada grupo receberá um relatório com uma avaliação do trabalho submetido, o que constitui o segundo ponto de avaliação. O terceiro instante de avaliação corresponde à submissão electrónica do código na sua versão final e à entrega do relatório. O relatório em formato PDF deverá ser incluído no ficheiro de submissão da fase final.

Num quarto instante há uma proposta enviada pelo corpo docente que pode conter a indicação de convocatória para a discussão e defesa do trabalho ou uma proposta de nota

Tabela 1: Datas importantes do Projecto

Data	Documentos a Entregar
até 6 de Março de 2020	Enunciado do projecto disponibilizado na página da disciplina.
até 27 de Março de 2020 (18h00)	Inscrição dos grupos no sistema Fenix.
17 de Abril de 2020, (18h00)	Conclusão da primeira submissão.
22 de Maio de 2020, (18h00)	Conclusão da entrega do projecto final com relatório anexo.
	Submissões posteriores às 18h00 de 22 de Maio têm penalização de 20 valores.
até 29 de Junho de 2020	Eventual discussão do trabalho (data combinada com cada grupo).

para a componente de projecto. Caso os alunos aceitem a nota proposta pelo docente avaliador, a discussão não é necessária e a nota torna-se final. Se, pelo contrário, os alunos decidirem recorrer da nota proposta, será marcada uma discussão de recurso em data posterior. O quinto instante acontece apenas caso haja marcação de uma discussão, seja por convocatória do docente, seja por solicitação dos alunos. Nestas circunstâncias, a discussão é obrigatoriamente feita a todo o grupo, sem prejuízo de as classificações dos elementos do grupo poderem vir a ser distintas.

As datas de entrega referentes aos vários passos da avaliação do projecto estão indicadas na Tabela 1.

As submissões electrónicas estarão disponíveis em datas e condições a indicar posteriormente na página da disciplina e serão aceites trabalhos entregues até aos instantes finais indicados.

Os alunos não devem esperar qualquer extensão nos prazos de entrega, pelo que devem organizar o seu tempo de forma a estarem em condições de entregar a versão final antes do prazo final de submissões.

Note-se que, na versão final, o projecto só é considerado entregue aquando da entrega do relatório. As submissões electrónicas do código não são suficientes para concretizar a entrega.

5.1 Funcionamento

A verificação do funcionamento do código a desenvolver no âmbito do projecto será exclusivamente efectuada nas máquinas do laboratório da disciplina, embora o desenvolvimento possa ser efectuada em qualquer plataforma ou sistema que os alunos escolham.

Esta regra será estritamente seguida, não se aceitando quaisquer excepções. Por esta razão, é essencial que os alunos, independentemente do local e ambiente em que desenvolvam os seus trabalhos, os verifiquem no laboratório antes de os submeterem, de forma a evitar problemas de última hora. Uma vez que os laboratórios estão abertos e disponíveis para os alunos em largos períodos fora do horário das aulas, este facto não deverá causar qualquer tipo de problemas.

5.2 Código

Não deve ser entregue código nem relatório em papel.

Na primeira fase de submissões é admissível que as submissões se façam através de um único ficheiro `*.c`, dispensando-se a divisão em módulos e a existência de uma `Makefile`.

Para a fase final, o código deve ser estruturado de forma lógica em vários ficheiros (`*.c` e `*.h`). As funções devem ter um cabeçalho curto mas explicativo e o código deve estar correctamente indentado e com comentários que facilitem a sua legibilidade.

A submissão final, por via electrónica, deverá conter o código do programa (ficheiros `.h` e `.c`), uma `Makefile` para gerar o executável e o relatório final em formato PDF. Todos os ficheiros (`*.c`, `*.h`, `Makefile`) e relatório devem ser comprimidos num único ficheiro, de acordo com as instruções que serão publicadas no Fénix quando se abrir o processo de submissões.

Sobre a utilização de comentários em código, convém referir que os mesmos não devem ser entendidos pelos alunos como enfeites que se colocam se e só se o programa já está completo e passa todos os testes, para que os professores não nos venham depois amolar o juízo. Se for este o caso, os comentários são praticamente inúteis, tendo o valor de uma iluminação em árvore de Natal. É giro mas só enfeita, não tendo qualquer utilidade prática.

Os comentários no código existem para guiar o desenvolvimento e para traduzir para linguagem humana aspectos importantes e/ou mais intrincados do código que está a ser desenvolvido em equipa, sem que a equipa tenha que estar toda a olhar para o computador ao mesmo tempo que se acrescenta código.

Quando bem feitos devem permitir uma fácil passagem de testemunho aos colegas de trabalho e também um fácil recomeço quando se está algum tempo sem produzir código.

5.3 Relatório

Os relatórios devem ser entregues na altura indicada na Tabela 1. O relatório entregue deverá permitir saber a identidade dos seus autores e forma de contacto: identificação do grupo, número, nome e e-mail dos alunos.

O relatório do projecto deverá ser claro e conciso e deverá permitir que se fique a saber como o grupo desenhou e implementou a solução apresentada. Ou seja, uma leitura do relatório deverá dispensar a necessidade de se inspeccionar o código para que fiquem claras as opções tomadas, justificações das mesmas e respectivas implementações.

No relatório deverá ser dada ênfase às funções e/ou funcionalidades mais importantes e à forma como se estruturaram os dados para facilitar a produção de soluções e recolha de dados estatísticos. O texto não deverá exceder as 5 páginas e deverá incluir em anexo alguns fluxogramas, minimamente comentados, que permitam perceber a arquitectura geral da implementação e das funções consideradas mais importantes.

O documento final, excluindo a capa, não deverá exceder as 8 páginas em formato A4 com tamanho de letra não inferior a 12 pt.

5.4 Critérios de Avaliação

Os projectos submetidos serão avaliados de acordo com a seguinte grelha:

- Testes passados na primeira submissão electrónica – 30%
- Apreciação do código submetido na primeira fase – 10%

- Testes passados na última submissão electrónica – 40%
- Apreciação do código submetido na fase final – 10%
- Relatório escrito – 10%

Tanto na primeira como na submissão electrónica final, cada projeto será testado com vários problemas de diferentes graus de complexidade, onde se avaliará a capacidade de produzir soluções correctas dentro de limites de tempo e memória. Para o limite de tempo, cada um dos testes terá de ser resolvido em menos de 60 segundos. Para o limite de memória, cada um dos testes não poderá exceder 100MB como pico de memória usada. Cada teste resolvido dentro dos orçamentos temporal e de memória que produza soluções correctas recebe um ponto.

Um teste considera-se errado se, pelo menos, um dos problemas do ficheiro de entrada correspondente for incorrectamente resolvido.

Se o corpo docente entender necessário, face à complexidade dos problemas a resolver, poderão os limites de tempo e/ou memória ser alterados.

Caso o desempenho de alguma submissão electrónica não seja suficientemente conclusivo, poderá ser sujeita a testes adicionais fora do contexto da submissão electrónica. O desempenho nesses testes adicionais poderá contribuir para subir ou baixar a pontuação obtida na submissão electrónica.

No que à avaliação do relatório diz respeito, os elementos de avaliação incluem: apreciação da abordagem geral ao problema e respectiva implementação; clareza e suficiência do texto, na sua capacidade de descrever e justificar com precisão o que está feito; e qualidade do texto escrito e estruturação do relatório.

Pela análise da grelha de avaliação aqui descrita, deverá ficar claro que a ênfase da avaliação se coloca na capacidade de um programa resolver correctamente os problemas a que for submetido. Ou seja, o código de uma submissão até pode ser muito bonito e bem estruturado e o grupo até pode ter dispendido muitas horas no seu desenvolvimento. No entanto, se esse código não resolver um número substancial de testes na submissão electrónica dificilmente terá uma nota positiva.

6 Código de Honestidade Académica

Espera-se que os alunos conheçam e respeitem o Código de Honestidade Académica que rege esta disciplina e que pode ser consultado na página da cadeira. O projecto é para ser planeado e executado por grupos de dois alunos e é nessa base que será avaliado. Quaisquer associações de grupos ou outras, que eventualmente venham a ocorrer, serão obviamente interpretadas como violação do Código de Honestidade Académica e terão como consequência a anulação do projecto aos elementos envolvidos.

Lembramos igualmente que a verificação de potenciais violações a este código é feita de forma automática com recurso a sofisticados métodos de comparação de código, que envolvem não apenas a comparação directa do código mas também da estrutura do mesmo. Esta verificação é feita com recurso ao software disponibilizado em

<http://moss.stanford.edu/>