Universidade Federal de Minas Gerais **Algoritmos e Estruturas de Dados II**
Instituto de Ciências Exatas **1º Semestre de 2022**
Departamento de Ciência da Computação **Prof. Gisele L. Pappa e Wagner Meira Jr**

Trabalho Prático 1

Poker Face

15 pontos. Entrega: 31/05/22

1. Especificação

Em geral, os alunos de ED são na sua maioria apreciadores de Pôquer - o famoso jogo de azar que alimenta o imaginário de muitos com promessas de riqueza rápida e, é tão interessante que, a proposta de ver pessoas jogando por si só já é suficiente para manter programas de TV com altos índices de audiência.
Infelizmente, apesar de ser um jogo que envolve movimentação de capital, nem todos os alunos estão dispostos a compartilhar o "dinheiro da merenda" dessa forma. Pensando nisso, você - profundo apreciador da centenária arte do Pôquer, conhecedor das técnicas avançadas de programação, de C, C++, de tipos abstratos de dados, de alocação dinâmica de variáveis, de processos simples de ordenação e outros métodos valiosos - resolve achar uma solução computacional para esse problema.
Para manter a emoção do jogo, você resolve criar um sistema onde qualquer um pode jogar, mesmo aqueles que não sabem quase nada das regras do jogo. O seu sistema basicamente cuida da etapa de **decisão,** onde vê-se **quem ganhou**, e do **montante de dinheiro virtual** que cada **jogador terá ao final das diversas rodadas**. Dessa forma, os jogadores podem fisicamente fazer suas jogadas, blefarem, realizarem apostas e o seu sistema só precisa saber quais cartas e quanto foi a aposta total de cada participante no momento da decisão de cada rodada para determinar quem foi o vencedor.

Você deve observar as seguintes definições, regras, testes de sanidade e formatos de entrada e saída para o seu algoritmo. Leia completa e atentamente os requisitos abaixo.

1.1. Definições

- Todo jogador tem um nome - que é um conjunto de letras e espaços, ou seja, pode ser um nome composto;
- Todo jogador começa com o mesmo valor em unidades de dinheiro;
- Uma carta é uma composição de um número e um naipe;
- Os números pertencem ao intervalo fechado de 1 a 13, representando do ás ao rei. Salvo exceções, vale a ordem crescente para determinar o valor;

- Os naipes são representados pelas letras maiúsculas: P, E, C, O - respectivamente, paus, espadas, copas e ouros. Não há uma ordem para determinar o valor;
- Uma mão tem sempre 5 cartas;
- O pingo é o valor que cada jogador, a cada rodada, deve obrigatoriamente colocar no pote;
- O pote é o montante de dinheiro arrecadado ao final de uma rodada;
- O pote é composto do pingo de todos os jogadores mais a aposta dos participantes da rodada;
- A cada rodada usa-se 1 baralho completo embaralhado;
- Os possíveis **resultados de uma rodada** são: um jogador ganhou e os demais perderam, dois a quatro jogadores empataram e os demais perderam;
- Todos os jogadores subtraem do seu montante virtual sua contribuição para o pote;
- O(s) jogador(es) vencedor(es) de uma rodada adiciona(m) a razão do valor integral do montante do pote pelo número de vencedores ao seu montante virtual;
- O valor mínimo do pingo é de 50 unidades de dinheiro;
- Toda aposta deve ser não nula e múltipla de 50 unidades de dinheiro.

1.2. Regras

- Todos os jogadores apostam na primeira rodada;
- Todo jogador é obrigado a participar com o pingo a cada pote (rodada), logo, mesmo jogadores que não participam da rodada estão contribuindo com o pote;
- A carta "Ás" pode assumir duas importâncias, a depender da jogada: carta menos valiosa, ao participar de uma sequência baixa, e carta mais valiosa, nos demais casos;
- Uma mão pode ser classificada em uma das 10 possíveis jogadas, na ordem decrescente de valor:

Royal Straight Flush [RSF]: São 5 cartas seguidas do mesmo naipe do 10 até ao Ás;

Straight Flush [SF]: São 5 cartas seguidas do mesmo naipe que não seja do 10 até ao Ás.

Four of a kind [FK]: São 4 cartas iguais, e em caso de empate ganha o jogador com a Quadra mais alta, caso permaneça empate ganha aquele que possuir a carta mais alta;

Full House [FH]: Uma tripla e um par. Em caso de empate ganha o jogador com a trinca mais alta, caso permaneça o empate ganha aquele que possuir o maior par;

Flush [F]: São 5 cartas do mesmo naipe sem serem seguidas. Caso dois jogadores possuam Flush ganha aquele que possuir a carta mais alta;

Straight [S]: São 5 cartas seguidas sem importar o naipe. Em caso de empate ganha aquele que possuir a carta mais alta;

Three of a kind [TK]: São 3 cartas iguais mais duas cartas diferentes, em caso de empate ganha aquele com a maior tripla, caso permaneça o empate ganha aquele que possuir a carta mais alta;

Two Pairs [TP]: São 2 pares de cartas. Em caso de empate ganha aquele que possuir o maior par maior, caso permaneça o empate ganha aquele que possuir o maior par menor, caso permaneça o empate ganha aquele que possuir a carta mais alta;

One Pair [OP]: São 2 cartas iguais e três diferentes. Em caso de empate ganha aquele que possuir o maior par, caso permaneça o empate ganha aquele que possuir a carta mais alta;

***High Card [HC]**:* Se nenhum jogador tiver uma das jogadas acima, ganha aquele que possuir a carta mais alta.

- De maneira genérica, as mãos mais valiosas que estão empatadas dentro de uma rodada devem seguir os seguintes critérios de desempate, na ordem: a jogada de cartas mais altas vence e as cartas mais altas que não participam da jogada vence;
- Se, apesar dos critérios de desempate, as mãos mais valiosas continuarem empatadas, deve-se dividir o pote entre os vencedores.

1.3 Testes de Sanidade

Uma falha em um teste de sanidade invalida a rodada, ou seja, deve ser ignorada no que diz respeito à formação do pote. Isto ocorre quando um jogador sem dinheiro contribuir com o pingo ou apostar.
OBS: Não deve ser realizado teste de sanidade para requisitos publicados nas definições, por exemplo, não há mão com menos de 5 cartas.

2. Formato de Entrada e Saída

2.1 Entrada

A entrada deverá ser passada através de um arquivo de nome *entrada.txt*.
A entrada é composta por representações de uma partida (ou conjunto de rodadas), de uma rodada (ou conjunto de jogadas) e de uma jogada (um jogador, sua aposta e mão).
A primeira linha da entrada é um par de valores inteiros que indica o *número de rodadas* ***n*** e o *dinheiro inicial dos participantes* ***di***.
A linha seguinte inicia um bloco de linhas que representa uma rodada. Haverão ***n*** blocos deste tipo.
Para cada rodada existe um bloco de linhas onde a primeira linha é um par de valores inteiros que indica o *número de jogadores* ***j*** e o *valor do pingo* ***p***.
As linha seguintes representam um jogador/jogada, haverão ***j*** linhas deste tipo.
Para cada jogada existe uma linha composta de: *nome do jogador*, *valor da aposta*, *sequência de cartas da mão*.

Genericamente:
Entrada = [Partida]
Partida = [*número de rodadas, dinheiro inicial*][Rodada_1, Rodada_2, …, Rodada_n]
Rodada = [*número de jogadores, pingo*][Jogada_1, Jogada_2, …, Jogada_j]
Jogada = [*nome do jogador*, *aposta*, *mão*]

Exemplo de entrada
3 1000
5 50
Giovanni 100 6O 3P 10E 11O 1O
John 200 3P 4E 3E 13C 13O
Thiago 100 12O 7P 12C 1O 13C
Gisele 300 12E 10C 11C 9C 13E
Wagner 50 5P 12P 5E 2E 1P
2 50
Wagner 200 2P 13E 9E 12C 2O
Gisele 350 11P 9P 2E 6E 4P
3 100
Thiago 250 1O 4P 1E 3O 8O
Gisele 100 9C 8C 8C 2C 6C
Giovanni 150 4P 12P 8E 12E 2P

2.2. Saída

A saída deverá ser retornada através de um arquivo de nome *saida.txt*.
A saída é composta por representações de resultado de rodada e pelo resultado final.
Para cada rodada existe um bloco de linhas onde a primeira linha é um par de valores inteiros e um conjunto de caracteres que indicam, respectivamente, o *número de vencedores* ***nv***, o *valor ganho por cada* ***v*** e a *classificação* ***c*** da jogada dos vencedores (dada por um dos códigos apresentados anteriormente: RSF, SF, FH, ...).
As ***nv*** linhas seguintes representam o *nome do(s) jogador(es)* vencedores da rodada.
Ao final de todas as rodadas deve-se exibir para cada jogador seu *nome* e seu montante de dinheiro, ordenados decrescentemente pelo montante.
O resultado de rodada inválida tem **0** vencedores, **0** valor ganho por cada e classificação **I** (inválida).

Genericamente:
Saída = [Resultado de Rodada$_1$, R. de Rodada$_2$, …, R. de Rodada$_n$][####][Resultado Final]
Resultado de Rodada = [*número de vencedores*, *valor ganho*, *classificação da jogada*][Vencedor de Rodada$_1$, V. de Rodada$_2$, …, V. de Rodada$_{nv}$]
Vencedor de Rodada = [*nome do jogador*]
Resultado Final = [*resultado de jogador$_{1º}$*, *resultado de jogador$_{2º}$*, …, *resultado de jogador$_{nº}$*]

*** Exemplo de Saída (referente a entrada de exemplo)**
1 1000 S
Gisele
1 200 OP
Wagner
1 1000 F
Gisele

####
Gisele 2050
Clodoveu 1350
John Holiver 600
Giovanni 550
Thiago 450

3. Desafio (OPCIONAL - 1.5 pontos extra)

Deverão ser implementadas heurísticas (vide referência) para detectar os determinados comportamentos dos jogadores:

- Qual jogador blefa mais?
- Qual o melhor jogador? (O conceito de melhor deve ser definido pelo aluno e deve ser diferente de "aquele que ganhou mais dinheiro")

As decisões de implementação referentes a heurística devem ser descritas em detalhes na documentação, assim como a análise de complexidade dos algoritmos.
A heurística deve ser implementada dentro de um módulo do seu programa.
A saída deverá ser retornada através de um arquivo de nome *saidaDesafio.txt*. O formato fica ao critério do aluno, e deve estar devidamente explicada na documentação.

4. Entregáveis

Você deve utilizar a linguagem C ou C++ para o desenvolvimento do seu sistema. O uso de estruturas pré-implementadas pelas bibliotecas-padrão da linguagem ou terceiros é **terminantemente vetado**. Caso seja necessário, use as estruturas que **você** implementou nos Trabalhos Práticos anteriores para criar **suas próprias implementações** para todas as classes, estruturas, e algoritmos.

Você **DEVE utilizar** a estrutura de projeto abaixo junto ao *Makefile* :

```
- TP
   |- src
   |- bin
   |- obj
   |- include
   Makefile
```

A pasta **TP** é a raiz do projeto; a pasta **bin** deve estar vazia; src deve armazenar arquivos de código (*.c, *.cpp ou *.cc); a pasta include, os cabeçalhos (*headers*) do projeto, com extensão *.h, por fim a pasta **obj** deve estar vazia. O **Makefile** deve estar na **raiz do projeto**. A execução

do **Makefile** deve gerar os códigos objeto *.o no diretório **obj,** e o executável do TP no diretório **bin**.

3. 1 Documentação

A documentação do trabalho deve ser entregue em formato **pdf**. A documentação deve conter os itens descritos a seguir :

Título, nome, e matrícula.
Introdução: Contém a apresentação do contexto, problema, e qual solução será empregada.
Método: Descrição da implementação, detalhando as estruturas de dados, tipos abstratos de dados (ou classes) e funções (ou métodos) implementados.
Análise de Complexidade: Contém a análise da complexidade de tempo e espaço dos procedimentos implementados, formalizada pela notação assintótica.
Estratégias de Robustez: Contém a descrição, justificativa e implementação dos mecanismos de programação defensiva e tolerância a falhas implementados.
Análise Experimental: Apresenta os experimentos realizados em termos de desempenho computacional e localidade de referência, assim como as análises dos resultados.
Conclusões: A Conclusão deve conter uma frase inicial sobre o que foi feito no trabalho. Posteriormente deve-se sumarizar o que foi aprendido.
Bibliografia: Contém fontes utilizadas para realização do trabalho. A citação deve estar em formato científico apropriado que deve ser escolhido por você.
Instruções para compilação e execução: Esta seção deve ser colocada em um apêndice ao fim do documento e em uma página exclusiva (não divide página com outras seções).

3.2 Submissão

Todos os arquivos relacionados ao trabalho devem ser submetidos na atividade designada para tal no Moodle. A entrega deve ser feita **em um único arquivo** com extensão **.zip**, com nomenclatura nome_sobrenome_matricula.zip}, onde nome, sobrenome e matrícula devem ser substituídos por suas informações pessoais. O arquivo **.zip** deve conter a sua documentação no formato **.pdf** e a estrutura de projeto descrita no início da Seção 3.

5. Avaliação

O trabalho será avaliado de acordo com:

- A Corretude na execução dos casos de teste - (50% da nota total)
- Apresentação da análise de complexidade das implementações - (15% da nota total)
- Estrutura e conteúdo exigidos para a documentação - (20% da nota total)
- Indentação, comentários do código fonte e uso de boas práticas - (10% da nota total)
- Cumprimento total da especificação - (5% da nota total)

Se o programa submetido não compilar[1], seu trabalho não será avaliado e sua nota será 0.
Trabalhos entregues com atrasos sofrerão penalização de 2^{d-1} pontos, com d = dias de atraso.

6. Considerações Finais

1. Comece a fazer esse trabalho prático o quanto antes, enquanto o prazo de entrega está tão distante quanto jamais estará.
2. Leia **atentamente** o documento de especificação, pois o descumprimento de quaisquer requisitos obrigatórios aqui descritos causará penalizações na nota final.
3. Certifique-se de garantir que seu arquivo foi submetido corretamente no sistema.
4. Plágio é CRIME. Trabalho onde o plágio for identificado serão **automaticamente anulados** e as medidas administrativas cabíveis serão tomadas (em relação a todos os envolvidos). Discussões a respeito do trabalho entre colegas são permitidas. É permitido consultar fontes externas, desde que exclusivamente para fins didáticos e devidamente registradas na sessão de bibliografia da documentação. **Cópia e compartilhamento de código não são permitidos**.

FaQ (Frequently asked Questions)

1. **Posso utilizar alguma estrutura de dados do tipo Queue, Stack, Vector, List, e etc..., do C++? NÃO**
2. Posso utilizar o tipo String? SIM.
3. Posso utilizar o tipo String para simular minhas estruturas de dados? NÃO
4. Posso utilizar alguma biblioteca para tratar exceções? SIM.
5. Posso utilizar alguma biblioteca para gerenciar memória? SIM.
6. As análises e apresentação dos resultados são importantes na documentação? SIM.
7. Os meus princípios de programação ligados a C++ e relacionados a engenharia de software serão avaliados? NÃO
8. Posso fazer o trabalho em dupla ou em grupo? NÃO
9. Posso trocar informações com os colegas sobre a teoria? SIM.
10. Posso fazer o trabalho no Windows, Linux, ou MacOS? SIM.
11. Posso utilizar IDEs, Visual Studio, Code Blocks, Visual Code, Eclipse? SIM.

*** Referências**

http://pt.wikipedia.org/wiki/Pôquer
http://en.wikipedia.org/wiki/Poker
http://pt.wikipedia.org/wiki/Anexo:Tabela_de_jogadas_do_pôquer
http://pt.wikipedia.org/wiki/Heurística#Conceito_simplificado

[1] Entende-se por compilar aquele programa que, independente de erros no Makefile ou relacionados a problemas na configuração do ambiente, funcione e atenda aos requisitos especificados neste documento em um ambiente Linux.