Trabalho Prático 1 (10 pontos)

"Popopopopopopopo... poker Face"

*** Especificação**

Em geral, os alunos que chegam na graduação de AEDS2 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 - aluno maroto, profundo apreciador da centenária arte do Pôquer, conhecedor das técnicas avançadas de programação, de 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.

*** 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 normal 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 múltipla de 50 unidades de dinheiro.;

*** 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;
- Somente tem direito a disputar o pote os jogadores cuja aposta seja igual à maior aposta;
- 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, devem dividir o pote entre os vencedores.

*** Testes de Sanidade**

Uma falha em qualquer dos testes de sanidade invalida a rodada, ou seja, deve ser ignorada no que diz respeito à formação do pote. Isto ocorre nos seguintes casos:
- Houverem cartas iguais em duas ou mais mãos em uma mesma rodada;
- A contribuição de um jogador ao pote for superior ao seu montante de dinheiro.

OBS: Esses são os únicos teste de sanidade necessários.

*** 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*, *sequencia 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 50 6O 3P 10E 11O 1C
John Holiver 200 3C 4E 3E 13P 13O
Thiago 100 12O 7P 12C 1O 13C
Gisele 300 12E 10C 11C 9C 13E
Clodoveu 50 5P 12P 5E 2E 1P
2 50
Clodoveu 250 2P 13E 9E 12C 2O
Gisele 250 11P 9P 2E 6E 4P
3 100
Thiago 250 1O 4P 1E 3O 8O
Gisele 250 9C 8C 3C 2C 6C
Giovanni 250 1P 1C 12E 12O 2P
```

*** 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)**

1 950 S
Gisele
1 750 OP
Clodoveu
1 1250 F
Gisele
####
Gisele 2200
Clodoveu 1250
John Holiver 600
Giovanni 500
Thiago 450

*** Desafio**

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 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, todavia deve estar devidamente explicada na documentação.

*** O que deve ser entregue**

- Código fonte do programa em C (todos os arquivos .c e .h), bem identado e comentado;
- Arquivo executável;
- Documentação do trabalho. Entre outras coisas, a documentação deve conter:
    1. Introdução: descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa.
    2. Implementação: descrição sobre a implementação do programa. Deve ser detalhada a estrutura de dados utilizada (de preferência com diagramas ilustrativos), o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, compilador utilizado, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado.
    3. Estudo de Complexidade: estudo da complexidade do tempo de execução dos procedimentos implementados e do programa como um todo (notação O).
    4. Testes: descrição dos testes realizados e listagem da saída (não edite os resultados).
    5. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
    6. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet, se for o caso. Atente que a cópia de qualquer trecho de código da Internet deverá estar devidamente citada e extremamente bem motivada.

  Obs1: Apesar desse trabalho ser simples, a documentação pedida segue o formato da documentação que deverá ser entregue nos próximos trabalhos. Um exemplo de documentação está disponível no learnloop.

*** Considerações importantes**

O trabalho tem o valor de 10 pontos. A pontuação é uma função da sua implementação e da sua documentação, logo, não se esqueça de fazer uma documentação criteriosa.
A data de entrega do trabalho é dia **17/04/2011, até às 23:59**. Isso significa que, submissões a partir de 00:01:00 do dia 18/04/2011 serão consideradas com 1 dia de atraso (*hard deadline!*).

Preste atenção nos seguinte pontos. Eles serão importantes na correção de seu trabalho:

- A especificação pode sofrer alterações caso surja necessidade. Fique atento às aulas e ao learnloop. É de suma importância que logo após o lançamento da especificacão, suas dúvidas sobre a mesma sejam postadas o mais rápido possível nos auxiliando a identificar requisitos não especificados corretamente;
- Dê preferência ao uso de Linux. Seu trabalho deverá ser **compilável** em Linux para ser devidamente corrigível pelos monitores. O executável pode ser voltado para qualquer plataforma, contudo, deve estar especificado no nome do arquivo (e.g.: tp1_win.exe);
- O trabalho deve ser implementado em C e não deve ser usada nenhuma biblioteca fora

do padrão ANSI-C;

- Comece a fazer este trabalho logo, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar;
- O trabalho é individual;
- Caso haja suspeita de que seu trabalho foi copiado, comprado, doado, etc você será avaliado de acordo e poderá ser convidado ou se convidar a explicar-se sobre os fatores que geraram a suspeita;
- Programe de forma organizada modularizando o código de forma adequada. Comente seu código extensivamente;
- Procure dar nome significativo para suas variáveis;
- Para cada *malloc* deve haver um *free*. Vazamento de memória em excesso será um critério de penalização de sua nota. Utilize *valgrind* (http://valgrind.org) para lhe auxiliar nesta tarefa;
- A submissão será feita pelo Learnloop. Faça um zip ou similar com todos os arquivos, inclusive documentação;
- **Penalização por atraso: (2^d – 1) pontos, onde *d* é o número de dias de atraso;**

*** 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
OBS: Existem outras referências no learnloop.