**MAC0110 – Introdução à Ciência da Computação**
**Bacharelado em Ciência da Computação**
**IME – Primeiro Semestre de 2015**

Primeiro Exercício-Programa 2 (EP2)
Professor: André Fujita

Data de entrega: até 23:55 do dia 03 de maio de 2015.

## The Ultimate Janken-Pon

O Janken-Pon, ou popularmente conhecido no Brasil como o jogo da pedra, papel e tesoura é um jogo de mãos para duas ou mais pessoas, que não requer equipamentos, mas apenas uma habilidade descomunal com as mãos.

No Janken-Pon, os jogadores devem simultaneamente esticar a mão, na qual cada um forma um símbolo que significa pedra (mão fechada), papel (mão aberta) ou tesoura (um "V" de vitória ou aquele símbolo hippie de paz e amor). Então os jogadores comparam os símbolos para decidir o vencedor da seguinte forma:

- Pedra ganha da tesoura
- Tesoura ganha do papel
- Papel ganha da pedra

Para adquirir profundos conhecimentos sobre essa arte milenar, consulte a página do wikipedia: http://pt.wikipedia.org/wiki/Pedra,_papel_e_tesoura

Neste exercício-programa, vamos (vocês alunos ☺) implementar um jogador de Janken-Po "inteligente". O jogo consistirá de várias partidas, sendo que cada partida será executada pelo seu EP versus o jogador humano. Ganhará aquele que vencer o maior número de partidas.

Você terá que implementar 3 modos de jogo: fácil, médio e difícil. O modo *fácil* consiste em gerar números aleatórios a partir de uma distribuição uniforme em que a chance do computador jogar pedra, papel ou tesoura é a mesma. Gere um número aleatório usando o método das congruências lineares (veja o Apêndice) e escolha o símbolo conforme a regra:

- se o número estiver no intervalo [0; 1/3[; escolha pedra;
- se o número estiver no intervalo [1/3; 2/3[; escolha tesoura;
- se o número estiver no intervalo [2/3; 3/3[; escolha papel.

O modo *médio* consiste em usar probabilidades condicionais. Baseado em $k$ jogadas anteriores, calcule a probabilidade da próxima jogada do jogador ser pedra, tesoura ou papel dada a última jogada dele. E faça a jogada para ganhar (óbvio!). Será que existe algum $k$ ótimo? Ou será melhor usar todas as jogadas anteriores? Faça a escolha do $k$ e justifique no código.

O modo *difícil* é por sua conta. Pense num algoritmo melhor que os dois acima (e que ganha da monitora :-P) e implemente. Se quiser, procure na internet. Porém, não vale copiar o código. Implemente você mesmo a ideia e escreva a referência no código como comentário.

O programa deve ler do teclado se o jogador deseja jogar no modo fácil, médio ou difícil através dos números 1, 2 e 3, respectivamente.

A jogada do ser humano será lida pelo teclado, com os números 1, 2 e 3 indicando pedra, tesoura e papel, respectivamente. A cada jogada, imprima a

jogada do computador e quem venceu o lance. O número zero indicará a parada de jogo. Nesse caso, imprima o número de vitórias, derrotas e empates do jogador, a porcentagem e indique quem foi o vitorioso geral.

**Curiosidades sobre esta arte**:
http://www.techtimes.com/articles/6438/20140503/winning-rock-paper-scissors-game-chinese-scientists.htm

http://motherboard.vice.com/read/game-theory-rock-paper-scissors

http://www.tofugu.com/2012/07/06/japans-most-dangerous-game-rock-paper-scissors/

http://dragonball.wikia.com/wiki/Rock,_Scissors_%27N%27_Paper

**Apêndice**:

*Método das congruências lineares*

Dado um número inicial $x_0$, conhecido como semente, o próximo número da sequência é dado por $x_1 = (ax_0 + b) mod(m)$. Em geral, o número $x_{i+1}$ é obtido a partir do número $x_i$ pela fórmula $x_{i+1} = (ax_i + b) mod(m)$.

Por exemplo, para $a = 7, b = 1, m = 13$ e $x_0 = 3$, a sequência de números gerada é: 9, 12, 7, 11, 0, 1, 8, 5, 10, 6, 4, 3, 9, ...

Neste EP, vocês podem usar como parâmetros, os valores $a = 22695477$, $b = 1$ e $m = 2^{32}$. Estes parâmetros são os mesmos usados pelo Borland C/C++ (um ambiente de programação da linguagem C).

Observações:

- Você pode implementar várias estratégias "vencedoras". Se essas outras estratégias forem realmente interessantes, haverá bonificação na nota (em outras palavras, sua nota pode ser acima de 10).
- Somente comandos ensinados em aula poderão ser usados no EP.
- O cabeçalho do EP 2 deve ser:

```
/* ********************************************************************** */
/* Nome: [coloque aqui seu nome]                                          */
/* Numero USP: [coloque aqui seu numero USP]                              */
/* Exercício-programa 2                                                   */
/* ********************************************************************** */
```

- EPs que não compilam receberão nota ZERO. O comando a ser usado na compilação do monitor será:
  gcc –Wall –ansi –pedantic –O2 –o ep1 ep1.c
  Certifique-se que seu EP compila no sistema operacional Linux com o comando acima. Mensagens de "warnings" serão penalizados na nota.
- Não serão aceitos EPs atrasados. Será considerado como EP não entregue.
- Você deve entregar somente o arquivo contendo o código fonte: *.c
  Outros arquivos que não sejam .c entregues "por engano" receberão nota ZERO.

- Seu programa NÃO precisa checar consistência de dados.
- O EP deve ser feito de forma INDIVIDUAL. Você pode conversar e discutir a solução com seus colegas, mas em hipótese alguma você deve mostrar e/ou ver o código dos outros. Qualquer problema com o código do EP deve ser tratado com o monitor da disciplina.
- EPs copiados parcialmente ou totalmente da internet ou de qualquer outra fonte será considerado plágio. EPs que tentem "mascarar" a cópia também serão considerados plágio.
- EPs com plágio receberão nota ZERO, o aluno será REPROVADO e seu nome será encaminhado a Comissão de Graduação.