Instituto de Computação da UNICAMP

Disciplina MC102: Algoritmos e Programação de Computadores - Turmas EF

Laboratório Nº 08. Peso 1.

Prazo de entrega: **17/04/2015 às 23:59:59**

PROFESSOR: Alexandre Xavier Falcão
MONITORES: João do Monte Gomes Duarte
MONITORES: Jadisha Yarif Ramírez Cornejo
MONITORES: Takeo Akabane
MONITORES: Eduardo Spagnol Rossi
MONITORES: Guilherme Augusto Sakai Yoshike

Ruína do Jogador

Nesse laboratório, iremos explorar funções em C. Para isso, iremos modelar uma máquina caça níquel e simular duas estratégias diferentes de apostas para tal máquina.

Um caça-níquel é uma máquina de jogo de chance, probabilística, que oferece ao jogador uma oportunidade para ele aumentar, se tiver sorte, ou diminuir, se tiver azar, o dinheiro que apostou inicialmente contra a máquina. Para permitir que uma pessoa jogue, um caça-níquel requer um depósito inicial — um cacife **C**. Um jogo nessa máquina é composto de jogadas sucessivas de uma moeda; cada jogada da moeda produz exclusivamente uma cara ou uma coroa. A probabilidade de se obter cara é 0.5, a probabilidade de se obter coroa também é 0.5. O caça-níquel é quem realiza as jogadas da moeda. Em cada jogada i, o jogador aposta um valor A_i contra a máquina. Por definição, o jogador ganha se a moeda atirada pelo caça-níquel produzir cara, caso contrário, ele perde a jogada. O jogo termina quando o jogador para de jogar porque atingiu o seu objetivo, digamos um valor **O**, ou quando o jogador perdeu todo o dinheiro que apostou contra a máquina.

As duas estratégias a serem simuladas são as seguintes:

- Sequencial: nessa estratégia o jogador sempre aposta uma moeda até ele alcançar seu objetivo O ou perder todo o seu cacife C.
- Martingale: nessa estratégia o jogador começa apostando uma moeda (R$ 1,00). Caso vença, na próxima jogada ele novamente aposta somente uma moeda. Caso o jogador perca a aposta, na próxima jogada ele aposta o dobro do apostado na jogada anterior, e segue apostando o dobro até o momento em que ele ganha a aposta, tendo um lucro de uma moeda, e começando então o processo novamente partindo de uma aposta de uma moeda até o jogador alcançar seu objetivo ou perder todo seu dinheiro.

Como exemplo, para a estratégia de Martingale podemos ter um jogo em que o cacife C seja R$ 8,00 e um objetivo O de R$ 10,00. Para esse jogo, podemos ter a seguinte sequência de jogadas: na primeira jogada A_1, o jogador aposta R$ 1,00 tendo como o resultado da moeda "cara", atualizando o saldo do jogador para R$ 9,00. Na próxima jogada A_2, o jogador aposta novamente R$ 1,00 mas dessa vez ele perde a jogada, ficando com R$8,00. Na próxima jogada A_3, o jogador deve apostar R$2,00, e o resultado novamente é negativo, deixando ele com um cacife de R$6,00. Em A_4, a aposta será R$4,00, e o caça níquel acaba dando cara, fazendo com que o seu cacife pule para R$10,00, alcançando seu objetivo. Note que caso na jogada A_4 o jogador perdesse, ele necessitaria fazer uma jogada A_5 com o valor de aposta R$8,00 seguindo a estratégia de Martingale, porém como seu cacife seria somente R$2,00, ele acabaria apostando todo o seu cacife restante.

Já para a estratégia Sequencial, para o mesmo jogo com cacife C de R$ 8,00 e objetivo O de R$ 10,00, o jogador sempre apostará R$1,00 independente do resultado da jogada anterior. Logo, para as mesmas jogadas A_1, A_2, A_3, A_4 o cacife irá para R$9,00, diminuirá para R$8,00, para R$7,00 e novamente para R$8,00, sendo que o jogador necessitaria continuar jogando para tentar chegar a seu objetivo R$10,00.

Especificação

O objetivo deste laboratorio é simular um conjunto de jogos e extrair estatísticas sobre eles. Uma maneira para calcularmos aproximadamente essas probabilidades é escrever um programa que realiza simulações de apostas de maneira automática. Quanto maior o número de simulações destes jogos, mais preciso será o cálculo da probabilidade de se atingir o objetivo.

Escreva um programa em C, que receba como entrada os seguintes dados:

1. O cacife inicial C.
2. O objetivo O.
3. O número N de jogos a serem simulados.

O programa deve escrever na saída padrão as seguintes mensagens:

- `"Sequencial: X, Y"`, onde X é a porcentagem do número total de jogos (séries de apostas) em que o jogador atingiu o seu objetivo utilizando a estratégia sequencial e Y o número médio de jogadas realizados por jogo. A porcentagem deve ser truncada para um número **inteiro** entre 0 e 100 e a média deve ser truncada para um número **inteiro**.
- `"Martingale: P, Q"`, onde P é a porcentagem do número total de jogos (séries de apostas) em que o jogador atingiu o seu objetivo utilizando a estratégia Martingale e Q o número médio de jogadas realizados por jogo. Novamente, a porcentagem deve ser truncada para um número **inteiro** entre 0 e 100 e a média deve ser truncada para um número **inteiro**.

Para a simulação de um jogo, você **deve** escrever uma função em C que recebe como parâmetros o cacife inicial C e o objetivo O. As suas funções devem simular uma sequência de jogadas realizadas pelo jogador utilizando as estratégias Sequancial e Martingale até ele atingir o objetivo ou até perder todo seu cacife. Considerando que funções em C podem apenas retornar um único valor, retorne o número de jogadas feitas pelo jogador. Este número de jogadas terá sinal positivo, caso ele atinja seu objetivo ou sinal negativo, caso ele perca todo o seu capital. Seu código deve conter a seguinte função:

```
/* Entrada: cacife inicial e o objetivo do jogador
   Saida: caso o jogador atinja seu objetivo utilizando a estratégia sequencial, retorna o numero de jogadas realizadas,
          caso contrario, retorna esse valor negado */
int Sequencial(int cacife, int objetivo);

/* Entrada: cacife inicial e o objetivo do jogador
   Saida: caso o jogador atinja seu objetivo utilizando a estratégia Martingale, retorna o numero de jogadas realizadas,
          caso contrario, retorna esse valor negado */
int Martingale(int cacife, int objetivo);
```

Além disso obrigatoriamente você **deve** utilizar a seguinte função para determinar o resultado de cada aposta:

```
/* Retorna 1 se o jogador ganhou a aposta, 0 caso contrario */
int ganhou(){
    // Codigo que gera uma sequencia pseudo-randomica,
    static unsigned long long int seed = 123456789;
    seed = (1103515245 * seed + 12345) % 2147483648;
    // Teste da probabilidade de ganhar a aposta
    return seed >= 2147483648/2;
}
```

Considere a função `ganhou()` como sendo a função utilizada pelo caça-níquel para jogar a moeda.

IMPORTANTE: realize todos os cálculos relativos a estratégia Sequencial primeiro para só então realizar os cálculos para a estratégia Martingale. Isso é necessário para que o resultado da função `ganhou()` no seu código seja sempre o mesmo que o da solução codificada no SuSy.

Exemplo

Se o programa realizasse uma simulaçãon com N=3, jogos X,Y e Z, com os seguintes dados de entrada: um cacife C de R$ 2,00 e um objetivo O de R$ 5,00. Um possível resultado para a simulação da estratégia Sequencial poderia ser expresso assim:

- X: ganhou, perdeu, ganhou, ganhou, ganhou
- Y: ganhou, perdeu, perdeu, perdeu
- Z: perdeu, ganhou, perdeu, ganhou, ganhou, ganhou, ganhou

Note que o jogo X parou após a realização de cinco apostas porque o jogador atingiu o seu objetivo. O jogo Y parou por que o jogador perdeu todo o seu dinheiro ao final de quatro apostas. O jogo Z parou por que o jogador atingiu o seu objetivo ao final da sétima aposta (jogada). Nesse exemplo, os valores retornados pela função **Sequencial(2, 5)** seriam os seguintes:

- Para o jogo X: **5** (valor positivo pois o jogador atingiu seu objetivo)
- Para o jogo Y:**-4** (valor **negativo** pois o jogador perdeu todo o seu dinheiro)
- Para o jogo Z: **7** (valor positivo pois o jogador atingiu seu objetivo)

Para esse exemplo, o programa deveria imprimir os seguintes valores:

- A porcentagem do número total de séries de apostas em que o jogador atingiu o seu objetivo: (2.0/3.0)*100 = 66 (truncada para um número inteiro)
- O número médio de apostas realizados por jogo: (5+4+7)/3 = 5 (truncada para um número inteiro)

Já para a estratégia Martingale, supondo que os jogos tivessem os mesmos resultados, o jogo X terminaria na quarta jogada, o jogo Y na terceira e o jogo Z na sétima jogada. Logo, a função **Martingale(2, 5)** retornaria em sequência 4, -3, 7 na sua simulação.

Observações

- Você pode supor que objetivo sempre é maior que o cacife inicial.
- Você pode supor que o número N de jogos a serem simulados é no máximo 1000000.
- Você **deve** quebrar a linha após imprimir cada resultado.
- Seu programa não precisa escrever nenhuma mensagem antes de ler o número pela entrada padrão.

Exemplos de execução

Exemplo 1:

```
500 550 1000
Sequencial: 92, 23231
Martingale: 91, 102
```

Exemplo 2:

```
1 2 1000000
Sequencial: 50, 1
Martingale: 50, 1
```

Exemplo 3:

```
50 66 2000
Sequencial: 76, 789
```

```
Martingale: 76, 30
```

Nota: Textos em `azul` denotam dados de entrada do programa.
Textos em `vermelho` denotam dados de saída do programa.

Observações gerais:

- O nome do arquivo submetido deve ser `jogador.c`
- O número máximo de submissões é 30;
- Não se esqueça de incluir no início do programa uma breve descrição dos objetivos, da entrada e da saída.
- Não se esqueça de **indentar e comentar** seu programa adequadamente. Isso **será** cobrado nas provas.
- Para a realização dos testes automáticos, a compilação se dará da seguinte forma: `gcc -std=c99 -pedantic -Wall -lm -o jogador *.c`.

Critérios importantes

Independentemente dos resultados dos testes do SuSy, não serão consideradas submissões que não aderirem aos critérios abaixo.

- O únicos header aceito para inclusão é `stdio.h`.
