MAC 115 - POLI - Primeiro Semestre de 1997

AVISOS

1. Cronograma para Correção de EPs:
   - Segundo EP: disponível a partir de 28/4 à tarde. Devolver em 11/5.
   - Exercício Complementar: disponível a partir de 11/5 à tarde. devolver em 18/5. (A correção deste exercício será **muito** fácil).
   - Terceiro EP: disponível 27/5 à tarde. Devolver em 8/6.
   - Quarto EP: disponível a partir de 19/6 à tarde. Devolver em 29/6.
2. Segunda prova: 6/6 (sábado), compareçam na secretaria do Departamento de Ciência da Computação às 9 horas. A prova termina a 1 hora.

Critério de Correção para o Segundo EP

1 Observações Gerais

- **(Notas numa planilha)** Como no EP anterior, as notas devem ser passadas ou numa planilha (de preferência a mesma) ou anotadas numa tabela, a critério da preferência do professor.
- **(Número de EPs impressos no CCE)** Conte quantos dos EPs foram impressos no CCE (que tem aquele papel "timbrado") para que possamos ter uma estimativa do uso das salas Pró-Aluno do CCE.
- **(Indicar erros encontrados)** Os erros encontrados devem ser indicados na listagem do programa, junto com o valor descontado. Anote também todas as observações que quiser fazer. A nota final do EP deve estar num lugar visível.
- **(Casos difíceis)** Nos casos difíceis de avaliar, o monitor deve consultar o professor da turma.
- **(Em caso de cola)** Será muito difícil dizer se houve cópia entre EP's, mas se houver indícios fortes de que algum programa foi copiado, então zero no exercício-programa! A decisão final porém é do professor. Nos EP's também deve ser anotada a nota que o aluno tiraria no EP caso não houvesse cola. Desconfie de EP's com diferenças gritantes de estilo de um exercício para outro dentro do mesmo EP.
  Cheque se o número USP que o aluno usa no código do programa é o do próprio aluno, isto é, se o valor que ele coloca na variável y (relativa ao sorteio) é o próprio número do aluno. É possível que alguns alunos copiem sem se dar conta deste "identificador" inserido no código. Uma discrepância não indica necessariamente cola, já que alguns alunos variaram o NUSP para observar o comportamento aleatório dos dados, e talvez não tenham voltado para seu próprio NUSP. Mas separe tais EPs, para que possamos eventualmente contrastar com um outro EP com mesmo número USP.
- **(Funções)** Muitos alunos começaram a fazer seu programa antes que o conceito de *funções* tivesse sido dado em aula. Desta forma, não desconte ou critique a repetição de trechos como por exemplo o sorteio. (Acreditamos que a maioria dos alunos usarão função neste exercício-programa.)
- **(Identação)** Desconte 1,5 ponto dos alunos que não fizerem sequer uma identação mínima de seus programas. Desconte 1.0 dos alunos que fizerem uma identação ruim, isto é, com muita discrepância e inconsistência, tornando o código difícil de ler. Observe que pode haver diferentes estilos de indentação (se

indento os inícios e fins de bloco ou não, se ponho na mesma linha ou não, etc). O importante é verificar se o estilo é coerente e se os comandos internos são alinhados mais à direita que os comandos externos. Um único comando interno na mesma linha NÃO é necessáriamente má indentação. Por exemplo:

```
if (x < 0) x = -x;
while (x < n) x++;
if (x > 4) {
    x++;
    printf ("%d", n);
}
else x--;
```

- **(Saída da execução na impressora)** Desconte 1,5 pontos se o aluno não tiver entregue uma listagem com um exemplo de execução do programa. A saída da execução pode ser feita usando-se um arquivo de saída (conforme descrito em folha anexa ao EP), ou imprimindo-se a janela de saída (há muitos ambientes que tornam isto possível trivialmente).
- **(Uso da função sin(x))**
  Se o aluno não calcular o `seno(x)` usando a fórmula que fornecemos (ao invés disso simplesmente usando a função `sin(x)` disponível na biblioteca), desconte 4 pontos.
- **(Erro de sintaxe)** Programa com erro de sintaxe deve receber zero;
- **(Variável não inicializada)** Caso detecte uso de variável não inicializada (uma variável cujo valor esteja sendo usado sem que antes tenha sido lido ou atribuído) desconte 1.0 ponto;

2 Erros Comuns

- **(Inicialização da variável `y` ou `passado`)** Inicializar a variável `y` ou `passado` (relativa ao sorteio dos dados, conforme descrito no enunciado do EP) dentro do loop de repetição de apostas. Esta variável só deve ser inicializada com o valor 0. $< nusp >$ uma vez, no início da execução.
  Descontar 2.5 pontos;
- **(Função para sortear um número)** Outro erro relacionado à variável y pode ocorrer com alunos que façam uma função para sortear um número, e fornecem o `y` ou `passado` como argumento por valor (e não por referência usando "`double *y`", afinal eles só estão vendo este assunto agora). Como a rotina alterara o `y` ou `passado`, mas o `y` ou `passado` foi passado por valor, a variável do programa nunca é atualizada. Alguns alunos usam `y` ou `passado` como variável global, e aí está ok (mas escreva na listagem que isto não é uma 'boa' prática de programação).
  Se um aluno estiver mudando o `y` ou `passado` dentro de uma função em que `y` ou `passado` foi passador por valor, desconte 2.0 (e explique que ele deveria ter percebido que o sorteio dos dados ficou comprometido, sorteando sempre a mesma coisa)
- **(Cálculo do seno I)** No cálculo da fórmula fornecida para `seno(x)`, alguns alunos guardam a parte do denominador (um fatorial) em uma variável. Se esta variável para armazenar os fatoriais (3!, 5!, 7!, 9!, etc) tiver sido declarada como `int`, a variável estoura bem cedo (pois por exemplo 8! é 40320). Neste caso, explique que a variável deveria ter sido declarada como `float` ou `double`, e desconte 0.5 ponto; (Uma maneira melhor de fazer o cálculo do seno, usando a séria de potências dada, é calcular o próximo termo da série aproveitando o valor do termo anterior.)
- **(Cálculo do seno II)** Ainda no cálculo do seno, olhe o trecho do programa (função ou as linhas do programa principal com esta parte) e verifique que a lógica está correta.
- **(Histograma)** Não é necessário que o histograma seja sofisticada, mas se a saída estiver bem bagunçada e difícil de entender (asteriscos não começando na mesma coluna, etc), desconte 1.0.

Verifique também se a quantidade de número sorteados sendo feita é a necessária (alguns alunos se enganam da seguinte forma: para fazer `n` sorteios, eles começam com um contador de 0 e vão até `n` inclusive, portanto fazendo `n+1` sorteios. Analogamente, alguns alunos começam o contador com 1 e repetem enquanto o contador for `< n`, executando apenas `n-1` vezes). Para erros deste tipo, desconte 0.5.

3 Versão Urkasky e Jogo do Bixo

(Consistência dos dados) Não estaremos descontando valor por "consistência de dados de entrada", ou seja, o aluno não precisa validar o valor da aposta (ou o tipo da aposta, no caso das versões que oferecem várias possibilidades).

(Conversa programa × usuário) Em algumas versões do segundo exercício-programa foi pedido que o aluno perguntasse ao usuário se este deseja ou não continuar apostando. É indiferente se o aluno usou `getc`, ou `getch`, ou `getchar` ou `scanf` para obter a resposta do usuário. Aceite também soluções que pedem para o usuário digitar 0 para continuar e 1 para parar.

`(Loop no laço principal)` Ainda neste controle sobre continuar ou não a execução do jogo, alguns alunos lêm a opção do usuário em uma variável "resposta" e escrevem o seguinte:

```
while (resposta != ´N´ || resposta != ´n´) { ...}
```

ou ainda

```
do { ... } while (resposta != ´N´ || resposta != ´n´)
```

Obviamente, em ambos os casos o loop nunca pararia.
Se você encontrar algo assim, o aluno provavelmente interrompeu com control-break (ou equivalente) a execução a fim de conseguir um arquivo de saída. Desconte 2,0 pontos neste caso.

3.1 Versão Urkasky

Um erro relativamente frequente que os alunos desta versão podem fazer se refere a usar o operador de divisão "/" com dois argumentos inteiros, e esperar que a conta tivesse um resultado fracionário. Por exemplo,

```
num_asteriscos = (240 / N )  * num_deu_1;
```

ou

```
num_asteriscos = (240 * num_deu_1) / N;
```

Por exemplo, se o dado foi lançado 1000 vezes (N = 1000) e o valor 1 apareceu 200 vezes (num_deu_1 = 200) o primeiro cálculo dará (240/1000)*200 = 0*200 = 0. O aluno deveria ter feito um "casting" de forma a garatir que o "/" seria feito com reais (por exemplo (240.0/N) * num_deu_1 ou ((double) 240 / N) * num_deu_1). Este erro ficaria evidente para qualquer N > 240. Neste caso, desconte 0.7 ponto.
No caso do segundo exemplo acima, o erro só se mostraria em casos mais extremos, em que por exemplo N = 100 e num_deu_1 = 4 (isto é que é dado viciado!). Neste caso, desconte 0.5 ponto.

3.2 Versão Jogo do Bixo

Esta versão têm quatro tipos de jogos (MILHAR, CENTENA, TERNO e GRUPO) e o teste de honetidade do jogo (TESTAR). Verifique se a saída do programa testou os quatro tipos de jogos e o teste de honestidade. Se o aluno esqueceu de testar (na execução que resultou na saída impressa) um dos casos, desconte 1,0 ponto. Se ele não testou por não ter feito a parte, desconte um total de 2,0 pontos (mas isto provavelmente não acontecerá ...)

Não é necessário que a saída siga o formato do exemplo apresentado no enunciado, mas é importante que a interação com o usuário seja clara. O usuário deve digitar vários tipos de informação (tipo do jogo, valor da aposta, e aí dependendo do tipo da aposta um ou mais números). Desconte 1.0 ponto se o programa não guia o usuário sobre que valor a digitar.

Um critério construtivo: para efeito de nota, cada um dos tipos de jogos e o teste de honestidade valem 2,0 pontos. Assim se um aluno fez corretamente quatro destes cinco 'módulos' do programa e não perdeu pontos com algum dos erros descritos então este alunos deverá tirar nota 8,0.

Um critério destrutivo: se algum dos 5 módulos não estiver funcionando desconte de 1,0 ponto a 2,0 pontos dependendo da gravidade do erro (gravidade = quanto trabalho você teria para consertar o erro).

Alguns alunos podem ter interpretado alguns detalhes dos jogos de maneiras diferentes. Se for este o caso, não é necessário descontar nota alguma. (Note que 'interpretar alguns detalhes de maneira diferente' não é o mesmo que fazer algo que não tem nada a haver com o enunciado, ou algo que esteja claramente errado.)

4 Versão Las Begas

Nessa versão, o aluno não deve implementar um jogo onde o usuário faz as apostas. Foi pedido apenas que o programa do aluno simule o apostador e a banca várias vezes para fazer certas estatísticas. Ou seja, não há interação com o usuário. Dê um bônus de no máximo 1,0 ponto, caso o aluno, além do que foi pedido no enunciado, tenha implementado uma opção do usuário ser o apostador.

Possíveis problemas desta versão:

- **Usuário é o apostador**: Se o aluno implementou o jogo mas não o que foi pedido no enunciado, desconte 4,0 pontos.
- **Usuário digita o valor de teto**: Se o aluno leu o valor de teto e simulou 100 vezes apenas para este valor, desconte 2.0 pontos.
- **teto = 0.0**: Não desconte nada se o aluno não simula para `teto = 0.0`.
- **Empate**: A banca ganha quando tem os mesmos pontos e menos cartas, ou seja, ela não pede mais cartas quando está com a pontuação igual a do apostador, mas o apostador tem mais cartas. Se o aluno ignora isso e nem tem contadores para o número de cartas do apostador e da banca, desconte 1,5 pontos. Se o aluno tem os contadores, mas dá cartas para a banca até que ela tenha mais pontos que o apostador e usa os contadores apenas para decidir quem ganhou no fim, desconte 0,5 ponto.
- **Reinicializar**: muitos alunos esquecem de reinicializar algumas variáveis, e isso pode afetar a saída de vários jeitos. Desconte até 2,0 pontos dependendo de quanto isso afeta a saída, ou seja, de quão fácil era notar que havia algum problema.