BCC202 – Estruturas de Dados I

Departamento de Computação - Universidade Federal de Ouro Preto - MG
Professor: **Pedro Silva** (`www.decom.ufop.br/`)

MÉTODOS DE ORDENAÇÃO (*Bubble Sort*)

- **Submissão com data e hora de entrega disponíveis na plataforma da disciplina**.

- Procedimento para a entrega:.

1. Siga atentamente quanto ao formato da entrada e saída de seu programa, exemplificados no enunciado.
2. Durante a correção, os programas serão submetidos a vários casos de testes, com características variadas.
3. A avaliação considerará o tempo de execução e o percentual de respostas corretas.
4. Eventualmente, serão realizadas entrevistas sobre as práticas para complementar a avaliação.
5. Considere que os dados serão fornecidos pela entrada padrão. Não utilize abertura de arquivos pelo seu programa. Se necessário, utilize o redirecionamento de entrada.
6. Os códigos fonte serão submetidos a uma ferramenta de detecção de plágios em software.
7. Códigos cuja autoria não seja do aluno, com alto nível de similaridade em relação a outros trabalhos, ou que não puder ser explicado, acarretará na perda da nota.
8. Códigos ou funções prontas específicos de algoritmos para solução dos problemas elencados não são aceitos.
9. Não serão considerados algoritmos parcialmente implementados.

- Bom trabalho!

Quem vai ganhar o jogo?

Andréa, Carlos e Marcelo são muito amigos e passam todos os finais de semana à beira da piscina. Enquanto Andréa se bronzeia ao sol, os dois ficam jogando Bolhas. Andréa, uma cientista da computação muito esperta, já disse a eles que não entende por que passam tanto tempo jogando um jogo tão primário.

Usando o computador portátil dela, os dois geram um inteiro aleatório n e uma sequência de inteiros, também aleatória, que é uma permutação de $1, 2, \ldots, n$. O jogo então começa, cada jogador faz um movimento, e a jogada passa para o outro jogador. Marcelo é sempre o primeiro a começar a jogar. Um movimento de um jogador consiste na escolha de um par de elementos consecutivos da sequência que estejam fora de ordem e em inverter a ordem dos dois elementos. Por exemplo, dada a sequência $1, 5, 3, 4, 2$, o jogador pode inverter as posições de 5 e 3 ou de 4 e 2, mas não pode inverter as posições de 3 e 4, nem de 5 e 2. Continuando com o exemplo, se o jogador decide inverter as posições de 5 e 3 então a nova sequência será $1, 3, 5, 4, 2$.

Mais cedo ou mais tarde, a sequência ficará ordenada. Perde o jogador impossibilitado de fazer um movimento. Andréa, com algum desdém, sempre diz que seria mais simples jogar cara ou coroa, com o mesmo efeito. Sua missão, caso decida aceitá-la, é determinar quem ganha o jogo, dada a sequência inicial. Se a quantidade de movimentos é par, então Carlos ganha. Caso contrário, ganha Marcelo.

Considerações

- Não altere o nome dos arquivos.
- O arquivo `.zip` deve conter na sua raiz somente os arquivos-fonte.
- Há vários casos de teste. Você terá acesso (entrada e saída) de casos específicos para realizar os seus testes localmente.

O código-fonte deve ser modularizado corretamente conforme os arquivos de protótipo fornecidos. Um vetor dinâmico de inteiros deve ser alocado e posteriormente desalocado para armazenar os números. O número de movimentos deve ser calculado utilizando algum método de ordenação, porém, cada caso de teste deve ser resolvido em até 1 segundo!

Especificação da Entrada e da saída

Cada caso de testes consiste de várias rodadas do jogo. Os dados de cada rodada estão em uma única linha, consistindo de números inteiros separados por um espaço em branco. Cada linha contém um inteiro n arbitrário, seguido da sequência inicial $P = [x_1, x_2, \ldots, x_n]$ de n inteiros distintos, em que $1 \leq x_i \leq n$ para $1 \leq i \leq n$. O final da entrada é indicado por uma linha que contém apenas o número zero.

Para cada rodada da entrada seu programa deve imprimir uma única linha, com o nome do vencedor, igual a Carlos ou Marcelo, sem espaços em branco.

Entrada	Saída
5 1 5 3 4 2	Marcelo
5 5 1 3 4 2	Carlos
5 1 2 3 4 5	Carlos
6 3 5 2 1 4 6	Carlos
5 5 4 3 2 1	Carlos
6 6 5 4 3 2 1	Marcelo
0	

Diretivas de Compilação

```
$ gcc -c ordenacao.c -Wall
$ gcc -c pratica.c -Wall
$ gcc ordenacao.o pratica.o -o exe
```

Avaliação de *leaks* de memória

Uma forma de avaliar se não há *leaks* de memória é usando a ferramenta valgrind. Um exemplo de uso é:

```
gcc -g -o exe *.c -Wall; valgrind -leak-check=yes -s ./exe < casoteste.in
```

Espera-se uma saída com o fim semelhante a:

```
==38409== ERROR SUMMARY: 0 errors from 0 contexts (suppressed:  0 from 0)
```

Para instalar no Linux, basta usar: `sudo apt install valgrind`.