BCC202 – Estruturas de Dados I

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

LISTA

- **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!

Trabalhando com listas circulares

Implemente um **Tipo Abstrato de Dado (TAD)** Lista utilizando vetores mas que trabalhe como uma lista circular. Ademais, considerando a sua implementação, o tamanho máximo do vetor deve ser 10.

Uma lista circular é um tipo de lista encadeada (mas que nesta prática será feita com vetores para facilitar) em que o último nó está conectado de volta ao primeiro nó, formando um ciclo. Essa estrutura tem algumas características especiais que a diferenciam de outras listas. A Figura 1 apresenta um exemplo gráfico de uma lista circular.

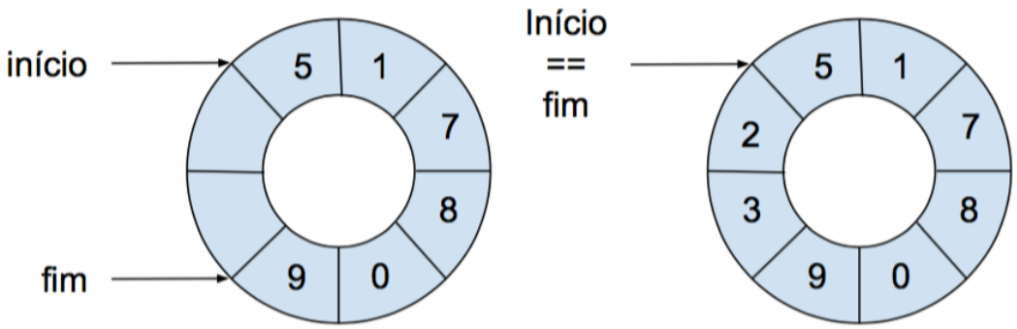

Figura 1: Exemplo gráfico de uma lista de circular.

Entre as suas características principais, está o fato de que ao invés do do último nó apontar para nada, ele aponta para o primeiro nó, criando uma conexão circular. Além disso, ela pode ser percorrida indefinidamente em um *loop*, começando de qualquer posição. Ela se torna principalmente útil quando temos problemas que envolvem operações que requerem ciclos (como rodízios, filas circulares, ou jogos baseados em turnos).

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. A sua lista será somente de inteiros e com um vetor estático.

Note que **não** se deve utilizar algum algoritmo de ordenação para resolver este problema. Basta ir selecionando qual o próximo duende mais velho e seguir escalando os times.

Especificação da Entrada e da saída

A entrada é composta por códigos, descritos a seguir, que identificam as operações a serem realizadas e os seus dados complementares, quando cabível. Para facilitar, neste momento, considere que somente uma variável do tipo lista será criada e manipulada.

Dica: antes de executar as operações a seguir, a variável do tipo Lista* deve estar devidamente alocada.

As operações são:

1 para inserir, de forma ordenada, um elemento na lista; é informado também o elemento que será inserido.

2 para retirar um elemento específico da lista; é informado também o elemento que será retirado.

3 para retirar o primeiro elemento da lista.

4 para verificar se um elemento pertence à lista, seguido pelo elemento em questão.

5 para imprimir todos os elementos da lista.

6 para imprimir onde está a posição inicial da lista circular.

-1 para encerrar o programa.

A saídas serão em conformidade com as operações que maninpulam um diretório especificadas na entrada.
Exemplo de entrada e saída para o problema:

Entrada	Saída
1 5 1 4 1 6 1 1 5 2 4 2 1 5 3 4 5 -1	1 4 5 6 5 6 Nao Pertence

Diretivas de Compilação

```
$ gcc -c lista.c -Wall
$ gcc -c pratica.c -Wall
$ gcc lista.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`.