Algoritmos e Estruturas de Dados
1º Teste
30 de Outubro 2019

Pergunta I (4,5 valores)

Para cada uma das seguintes alíneas, indique o estado das estruturas de dados (desenhando), após a execução da sequência de instruções dadas. No desenho deve incluir as variáveis de instância da respectiva classe.

(A) Lista simplesmente ligada (com *cabeça*, *cauda* e *número de elementos*)

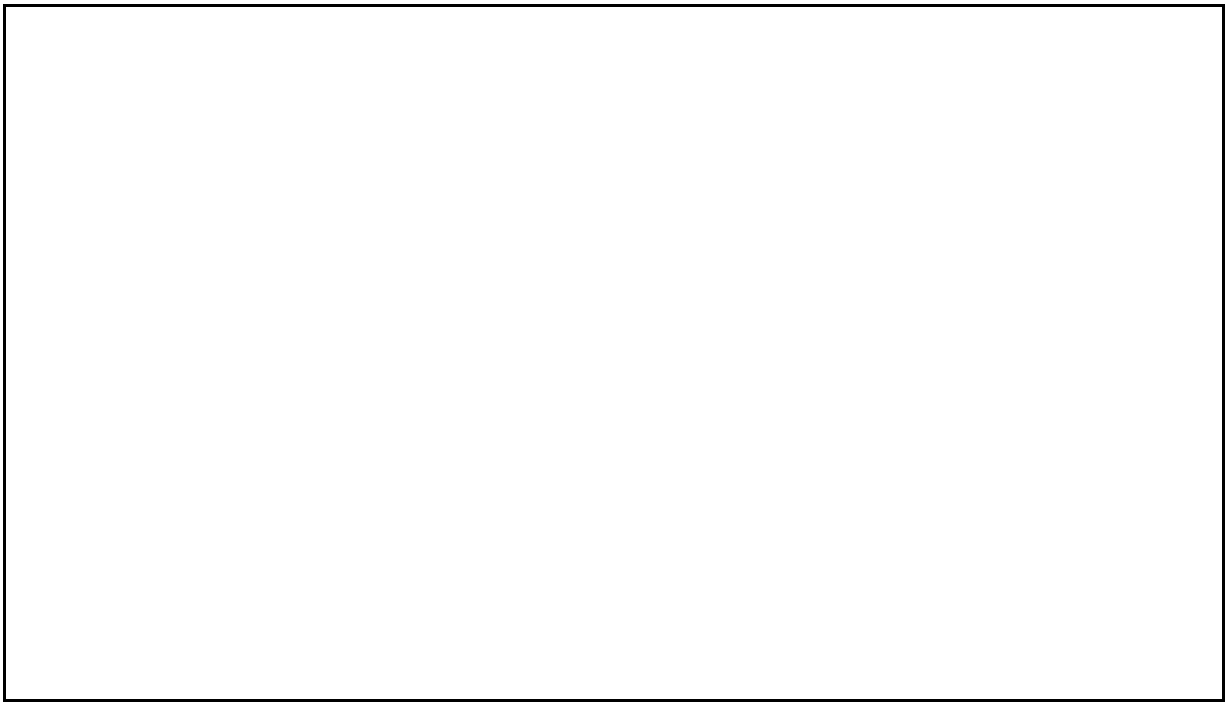
```
List<Integer> l = new SinglyLinkedList<Integer>();
l.addLast(12);
l.addFirst(15);
l.add(3,17);
l.remove(4);
l.add(0,45)
l.remove(1)
l.add(1,60);
```

(B) Fila implementada com vetor circular (com *início* e *fim* - índices do vetor)

```
Queue<Integer> q= new QueueInArray<Integer>(10);
q.enqueue(5)
q.enqueue(3)
q.dequeue()
q.enqueue(2)
q.enqueue(8)
q.dequeue()
q.dequeue()
q.enqueue(9)
```

(C) Tabela de dispersão aberta (tamanho do vetor 11, assuma que o hashcode do elemento é a própria chave inteira)

```
Map<Integer,String> m= new SepChainHashTable<Integer,String>(8);
m.insert(11,”Ana”);
m.insert (10, “Luis”);
m.insert(88,”Margarida”);
m.insert(4,”Joao”);
m.insert(11,”Maria”);
m.insert(15,”Luis”);
m.insert(1, “Susana”);
m.remove(26);
```



Pergunta II (4 valores)

A classe SportsClub tem uma variável de instância *eventos*, que é um `Map` em que a chave é uma data do tipo `Date` e o valor é uma Lista de Eventos (eventos que ocorrem nessa data). As datas existentes neste `Map` são todas referentes ao ano corrente (2019). Para além disso existem clubes desportivos que tem um grande número de eventos, normalmente em todos os dias do ano tem pelo menos 1 evento.

Foi necessário adicionar uma funcionalidade/método "OrderedIterator" a esta classe, que criasse um iterador de datas de eventos, que as permitisse percorrer de forma ordenada (ordem cronológica).

```
public Iterator<Date> OrderedIterator();
```

Lembre-se que o TAD `Map` tem iteradores que lhe permitem percorrer todos os elementos/chaves/valores (de forma não ordenada). A interface `Date` está em anexo.

Tendo em conta as características dadas, indique o(s) nome(s) do(s) algoritmo(s) de ordenação que usaria para percorrer por ordem cronológica as datas em que há eventos (método `OrderedIterator)`. Explique como faria e justifique a sua escolha.

Pergunta III (5 valores)
O método "`middleNode`", na classe "`DoublyLinkedList`", devolve o nó que está no meio da lista duplamente ligada. Nesta classe as únicas variáveis de instância são a *cabeça* e *cauda* da lista (ver fragmento de código abaixo). Logo, caso necessite usar algum método da classe, deve implementá-lo.

O nó do meio duma lista com um número *n* elementos é o *k*-ésimo nó, com $k=n/2$ se *n* par, ou $k=(n+1)/2$ se *n* ímpar. Caso a lista esteja vazia devolve null.

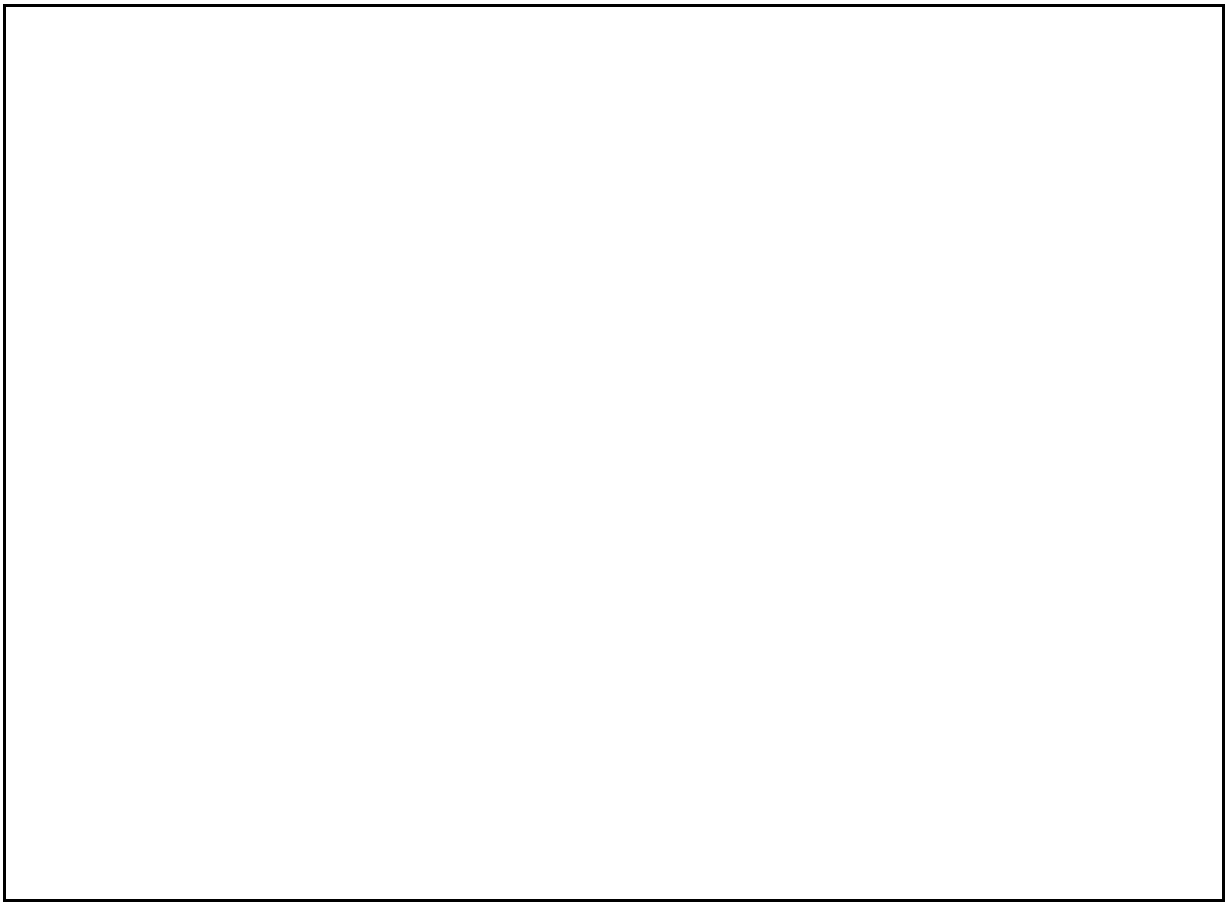
Implemente o método "`middleNode`". A sua implementação deve ser eficiente (O(*n*) - linear) e **cumprir** com $T(n)= c * n + b$, com *c* e *b* constantes, em que $c <= 1$ e $b >= 0$.

```
package dataStructures;

public class DoublyLinkedList<E> implements TwoWayList<E>  {
    // Node at the head of the list.
    protected DListNode<E> head;

    // Node at the tail of the list.
    protected DListNode<E> tail;

    protected DListNode<E> middleNode(){
```



```
    }
        …
}
```

Pergunta IV (6,5 valores)

Título: Documento de texto organizado em capítulos.

Cada capítulo tem uma sequência de linhas. Cada linha é uma sequência de palavras.

Uma linha com uma única palavra “Capítulo” indica o início do próximo capítulo. A linha imediatamente a seguir é o nome do capítulo.

Cada página do documento contém no máximo 25 linhas. Cada capítulo começa numa página nova.

Na criação do documento é dado um objeto que implementa a interface “Reservada”, a qual tem um método “ePalavraReservada” para indicar se uma dada palavra é considerada “palavra reservada” neste documento.

Neste documento é possível realizar as seguintes funcionalidades:

(1) inserir linha no final do documento;

(2) Consultar o número de capítulos existentes até ao momento;

(3) Listar o índice do documento da seguinte forma:

1. NomeCapitulo_1 página_inicial_capitulo_1
2. NomeCapitulo_2 página_inicial_capitulo_2

...

n. NomeCapitulo_n página_inicial_capitulo_n

(4) Listar as linhas (texto) de um dado capítulo. Esta listagem é dada pela ordem de inserção das linhas;

(5) Listar o índice das palavras reservadas da seguinte forma:

palavra_1 - sequência de linhas onde aparece
palavra_2 - sequência de linhas onde aparece
…
palavra_k - sequência de linhas onde aparece

Esta listagem é ordenada por ordem alfabética.

(A) Indique os tipos abstractos de dados do domínio do problema que usaria na sua solução. Indique o que representa cada um.

(B) Indique as classes que usaria no domínio do problema, indicando as variáveis de instância que usaria para implementar de forma adequada. Justifique todos os TADs (interfaces) do pacote dataStructures que utilizou. **Não pode usar interfaces do pacote java.util.**
Não se preocupe com as estruturas de dados que implementam os TADs do pacote dataStructures. Assuma que todas as interfaces estão implementadas de forma eficiente.

(C) Explique como implementaria cada funcionalidade dada, indicando as variáveis de instância e operações/métodos utilizados.

(continuação pergunta _____)

Anexos

Interface Date (pergunta II)

```
public interface Date extends Comparable<Date>{
	/*Returns the day of the date */
	int getDay();
	/*Returns the month of the date */
	int getMonth();
	/*Returns the year of the date */
	int getYear();
}
```

Classe DListNode<E>

```
package dataStructures;

class DListNode<E>{
	 // Element stored in the node.
	 protected E element;
	 // (Pointer to) the next node.
	 protected DListNode<E> next;
	 // (Pointer to) the previous node.
	 private DListNode<E> previous;

	 public DListNode( E elem, DListNode<E> thePrev, DListNode<E> theNext ){...}
	 public DListNode( E theElement ){ ...}
	 public E getElement( ){... }
	 public DListNode<E> getNext( ){...}
	 public DListNode<E> getPrevious( ){...}
	 public void setElement( E newElement ){...}
	 public void setNext( DListNode<E> newNext ){... }
	 public void setPrevious( DListNode<E> newPrevious ){... }
}
```

Interfaces do package dataStructures