Instituto Federal da Bahia – IFBA

Curso: Bacharelado em Sistemas de Informação - BSI

Disciplina: Estruturas de Dados Tipo: 1 Data: 31/10/2023

Docente: Cláudio Rodolfo Sousa de Oliveira

Discente:_____

PROVA INDIVIDUAL DA UNIDADE 2

Instruções

1. *A prova vale 6,0 pontos e cada questão vale 2,0 Pontos.*
2. *Tenha como base de desenvolvimento, os códigos criados em sala durante as aulas, usando a Linguagem de Programação Java.*
3. *Crie uma pasta com seu nome completo e coloque os arquivos de resolução dentro dela. Como a prova é prática, ao término chame o professor para extrair seus arquivos.*
4. *Atenção para não sobrescrever as questões. Recomenda-se inicialmente já salvar cada uma delas em um arquivo diferente para evitar acontecer este problema.*

QUESTÕES

1. Implemente o método *"`void sort()`"* para ordenar de forma crescente os elementos de uma Fila Dinâmica Duplamente Encadeada que armazena números inteiros não negativos.

2. Implemente o método "`void insertBegin(Object data)`" de uma Lista Dinâmica Duplamente Encadeada que sempre insere um novo dado no início da Estrutura de Dados.

3. Implemente um método *"`boolean isPalindrome(String data)`"* para verificação de palíndromos utilizando Pilha Dinâmica.

Boa Prova!

Instituto Federal da Bahia – IFBA

Curso: Bacharelado em Sistemas de Informação - BSI

Disciplina: Estruturas de Dados Tipo: 2 Data: 31/10/2023

Docente: Cláudio Rodolfo Sousa de Oliveira

Discente:_____

PROVA INDIVIDUAL DA UNIDADE 2

Instruções

1. *A prova vale 6,0 pontos e cada questão vale 2,0 Pontos.*
2. *Tenha como base de desenvolvimento, os códigos criados em sala durante as aulas, usando a Linguagem de Programação Java.*
3. *Crie uma pasta com seu nome completo e coloque os arquivos de resolução dentro dela. Como a prova é prática, ao término chame o professor para extrair seus arquivos.*
4. *Atenção para não sobrescrever as questões. Recomenda-se inicialmente já salvar cada uma delas em um arquivo diferente para evitar acontecer este problema.*

QUESTÕES

1. Implemente o método "`Object removeEnd()`" de uma Fila Dinâmica Duplamente Encadeada que sempre remove um dado do final da Estrutura de Dados.

2. Implemente o método "`void sort()`" que ordene de forma decrescente os elementos de uma Lista Dinâmica Duplamente Encadeada que armazena números reais.

3. Implemente um método "`String decToBin(String data)`" que retorne a representação Binária de números Decimais, utilizando Pilha Dinâmica.

Boa Prova!

```java
////////TIPO 1
//Questão 1
//Ordena - Algoritmo
//Para 1 único elemento considera-se que já está ordenado
//É necessário um laço mais externo que a cada iteração, faça com que o maior elemento em sua faixa de elementos, vá para sua posição correta (último, penúltimo, antepenúltimo, etc.)
//Laço mais interno que compara nodos adjacentes, e troca os seus conteúdos caso estejam decrescente.
public void sort() {
    if (quantidade > 1) {
        for (int i = 0; i < (quantidade-1); i++) {
            Nodo atual = ponteiroInicio;
            Nodo proximo = atual.getProximo();
            for (j = 0; j < (quantidade-1)+i; j++) {
                if (atual.getDado() > proximo.getDado()) {
                    Integer aux = atual.getDado();
                    atual.setDado(proximo.getDado());
                    proximo.setDado(aux);
                } //if
                atual = atual.getProximo();
                proximo = atual.getProximo();
            } //for
        } //for
    } //if
} //sort

//Questão 2
//Insere Inicio - Algoritmo
//caso seja possível, cria-se um novo nó e o popula com o dado passado
//corrige-se as referencias a este novo nó (proximo dele, e anterior de ponteiro de inicio)
//incrementa aquantidade de elementos
public void insertBegin(Object data) {
    if (!estaCheia()) {
        Nodo novoNodo = new Nodo();
        novoNodo.setDado(data);
        novoDado.setProximo(ponteiroInicio);
        if (!estaVazia()) {
            ponteiroInicio.setAnterior(novoDado);
        } else {    //para 0 elementos
            ponteiroFim = novoNodo;
        }
        ponteiroInicio = novoDado;
        quantidade++;
    } else {    //para (tamanho-1) elementos
        System.err.println("List is full!");
    }
}

//Questão 3
//É Palindromo - Algoritmo
//roda o laço até metade do tamanho do Texto (caso o tamanho seja ímpar, até o maior par menor que a metade do tamanho) empilhando os caracteres do Texto
//testa-se os caracteres empilhados são iguais aos caracteres da metade pro final do Texto
//se algum caracter for diferente retorna falso, caso contrário retorna verdadeiro
public boolean isPalindrome(String data) {
    boolean retorno = true;
    Empilhavel p = new PilhaDinamica(100);
    for (int i = 0; i < (data.length()/2); i++) {
        p.empilhar(data.charAt(i));
    }
    for (int i = (data.length()/2)+1; i < data.length(); i++) {
        char parte1 = (Char) p.desempilhar();
        char parte2 = data.charAt(i));
        if (parte1 != parte2) {
            retorno = false;
            break;
        }
    }
    return retorno;
}
```

```java

    ////////TIPO 2
    //Questão 1
    //Apaga Final - Algoritmo
    //caso seja possível, e haja mais que 1 elemento
    //retrocede o fim
    //corrige-se as referencias a este novo fim (elimina a referencia ao nodo posterior a ele)
    //Caso só há um elemento e ele será removido, a ED volta ao seu estado inicial
    //decrementa a quantidade de elementos
    public Object removeEnd() {
        Object temp = null;
        if (!estaVazia()) {
            temp = ponteiroFim.getDado();
            if (quantidade > 1) {
                ponteiroFim = ponteiroFim.getAnterior();
                ponteiroFim.setProximo(null);
            } else { //para 1 único elemento
                ponteiroFim = null;
                ponteiroInicio = null;
            }
            quantidade--;
        } else {    //para 0 elementos
            System.err.println("Queue is empty!");
        }
        return temp;
    }

    //Questão 2
    //Ordena - Algoritmo
    //Para 1 único elemento considera-se que já está ordenado
    //É necessário um laço mais externo que a cada iteração, faça com que o menor elemento em sua faixa de elementos, vá para sua posição correta (último, penúltimo, antepenúltimo, etc.)
    //Laço mais interno que compara nodos adjacentes, e troca os seus conteúdos caso estejam crescente.
    public void sort() {
        if (quantidade > 1) {
            for (int i = 0, ; i < (quantidade-1); i++) {
                Nodo atual = ponteiroInicio;
                Nodo proximo = atual.getProximo();
                for (j = 0; j < (quantidade-1)+i; j++) {
                    if (atual.getDado() < proximo.getDado()) {
                        Integer aux = atual.getDado();
                        atual.setDado(proximo.getDado());
                        proximo.setDado(aux);
                    } //if
                    atual = atual.getProximo();
                    proximo = atual.getProximo();
                } //for
            } //for
        } //if
    } //sort

    //Questão 3
    //Decimal para Binario - Algoritmo
    //enquanto o dividendo for maior que 2, dá pra continuar dividindo
    //quociente vira novo dividendo
    //empilha-se os restos da divisão inteira ( 0s ou 1s) do dividendo por 2.
    //empilha-se o último dividendo
    //retorna o conteúdo da pilha
    public String decToBin(String data) {
        Empilhavel p = new PilhaDinamica(100);
        int dividendo = Integer.parseInt(data);
        while (dividendo > 1) {
            int resto = dividendo % 2;
            int quociente = dividendo / 2;
            dividendo = quociente;
            p.empilha(resto);
        }
        p.empilha(dividendo);
        return p.imprimir();
    }
```

Instituto Federal da Bahia - IFBA

Curso: Bacharelado em Sistemas de Informação

Disciplina: Estruturas de Dados

Cláudio Rodolfo Sousa de Oliveira

Aluno: _____

Matrícula: _____ Questão 1: ___ Questão 2: ___

Prova 2 Extraoficial da Unidade 2 (Valor: 6,0 Pontos)

Instruções

Número Aluno (NA): _____ Número Professor (NP): _____
deslocamento = NA + NP
soma = Somatório dos números individuais de matrícula + deslocamento

Questão 1: soma % 23
Questão 2: (soma + deslocamento) % 23

Questões

0. Inserir um dado no início de uma Lista Dinâmica.
1. Inserir um dado no fim de uma Lista Dinâmica.
2. Inserir um dado numa posição lógica específica de uma Lista Dinâmica.
3. Buscar um dado do início de uma Lista Dinâmica.
4. Buscar um dado do fim de uma Lista Dinâmica.
5. Buscar um dado de uma posição lógica específica em uma Lista Dinâmica.
6. Buscar todos os dados de uma Lista Dinâmica.
7. Atualizar o dado do início de uma Lista Dinâmica.
8. Atualizar o dado do fim de uma Lista Dinâmica.
9. Atualizar uma posição lógica específica de uma Lista Dinâmica.
10. Atualizar todos os dados de uma Lista Dinâmica.
11. Apagar um dado do início de uma Lista Dinâmica.
12. Apagar um dado do fim de uma Lista Dinâmica.
13. Apagar um dado uma posição lógica específica de uma Lista Dinâmica.
14. Apagar todos os dados de uma Lista Dinâmica.
15. Ordenar os dados de forma crescente de uma Lista Dinâmica.
16. Ordenar os dados de forma decrescente de uma Lista Dinâmica.
17. Inverter os dados de uma Lista Dinâmica.
18. Imprimir os dados do início ao fim de uma Lista Dinâmica.
19. Imprimir os dados do fim ao início de uma Lista Dinâmica.
20. Verificar se um determinado dado existe em uma Lista Dinâmica.
21. Verificar a posição lógica da primeira ocorrência de um determinado dado em uma Lista Dinâmica.
22. Verificar a posição lógica da última ocorrência de um determinado dado em uma Lista Dinâmica.

Boa Prova!

```java
public class Gabarito2 {

//Questão 00
    public void inserirInicio(Object dado){
        if(!estaCheia()){
            NoDuplo noTemporario = new NoDuplo();
            noTemporario.setDado(dado);
            if(!estaVazia())
                ponteiroInicio.setAnterior(noTemporario);
            else
                ponteiroFim = noTemporario;

            noTemporario.setProximo(ponteiroInicio);
            ponteiroInicio = noTemporario;
            quantidade++;
        } else {
            System.err.println("Lista Cheia!");
        }
    }

//Questão 01
    public void inserirFim(Object dado) {
        if(!estaCheia()) {
            NoDuplo noTemporario = new NoDuplo();
            noTemporario.setDado(dado);
            if (!estaVazia()) {
                ponteiroFim.setProximo(noTemporario);
            } else  {
                ponteiroInicio = noTemporario;
            }
            noTemporario.setAnterior(ponteiroFim);
            ponteiroFim = noTemporario;
            quantidade++;
        } else {
            System.err.println("Lista Cheia!");
        }
    }

//Questão 02
    public void inserir(int posicao, Object dado) {
        if(!estaCheia()) {
            if(posicao >= 0 && posicao <= quantidade) {
                NoDuplo noTemporario = new NoDuplo();
                noTemporario.setDado(dado);

                NoDuplo ponteiroAnterior = null;
                NoDuplo ponteiroProximo = ponteiroInicio;

                for (int i = 0; i < posicao; i++) {
                    ponteiroAnterior = ponteiroProximo;
                    ponteiroProximo = ponteiroProximo.getProximo();
                }

                if (ponteiroAnterior != null)
                    ponteiroAnterior.setProximo(noTemporario);
                else
                    ponteiroInicio = noTemporario;

                if (ponteiroProximo != null)
                    ponteiroProximo.setAnterior(noTemporario);
                else
                    ponteiroFim = noTemporario;

                noTemporario.setAnterior(ponteiroAnterior);
                noTemporario.setProximo(ponteiroProximo);

                quantidade++;
            } else {
                System.err.println("Indice Invalido!");
            }
        } else {
            System.err.println("Lista Cheia!");
        }
```

```
74        }
75
76    //Questão 03
77        public Object selecionarInicio() {
78            Object dadoTemporario = null;
79            if (!estaVazia()) {
80                dadoTemporario = ponteiroInicio.getDado();
81            } else {
82                System.err.println("Lista Vazia!");
83            }
84            return dadoTemporario;
85        }
86
87    //Questão 04
88        public Object selecionarFim() {
89            Object dadoTemporario = null;
90            if (!estaVazia()) {
91                dadoTemporario = ponteiroFim.getDado();
92            } else {
93                System.err.println("Lista Vazia!");
94            }
95            return dadoTemporario;
96        }
97
98    //Questão 05
99        public Object selecionar(int posicao) {
100           Object dadoTemporario = null;
101           if (!estaVazia()) {
102               if (posicao >= 0 && posicao < quantidade) {
103                   NoDuplo ponteiroAuxiliar = ponteiroInicio;
104                   for (int i = 0; i < posicao; i++)
105                       ponteiroAuxiliar = ponteiroAuxiliar.getProximo();
106
107                   dadoTemporario = ponteiroAuxiliar.getDado();
108               } else {
109                   System.err.println("Indice Invalido!");
110               }
111           } else {
112               System.err.println("Lista Vazia!");
113           }
114           return dadoTemporario;
115       }
116
117   //Questão 06
118       public Object selecionarTodos() {
119           Object[] dadosTemp = new Object[quantidade];
120           if (!estaVazia()) {
121               NoDuplo ponteiroAuxiliar = ponteiroInicio;
122               for (int i = 0; i < quantidade; i++) {
123                   dadosTemp[i] = ponteiroAuxiliar.getDado();
124                   ponteiroAuxiliar = ponteiroAuxiliar.getProximo();
125               }
126           } else {
127               System.err.println("Lista Vazia!");
128           }
129           return dadosTemp;
130       }
131
132   //Questão 07
133       public void atualizarInicio(Object dado) {
134           if (!estaVazia()) {
135               ponteiroInicio.setDado(dado);
136           } else {
137               System.err.println("Lista Vazia!");
138           }
139       }
140
141   //Questão 08
142       public void atualizarFim(Object dado) {
143           if (!estaVazia()) {
144               ponteiroFim.setDado(dado);
145           } else {
146               System.err.println("Lista Vazia!");
```

```
147             }
148         }
149 
150     //Questão 09
151         public void atualizar(int posicao, Object novoDado) {
152             if (!estaVazia()) {
153                 if ((posicao >= 0) && (posicao < quantidade)) {
154                     NoDuplo ponteiroAuxiliar = ponteiroInicio;
155                     for (int i = 0; i < posicao; i++)
156                         ponteiroAuxiliar = ponteiroAuxiliar.getProximo();
157 
158                     ponteiroAuxiliar.setDado(novoDado);
159                 } else {
160                    System.err.println("Indice Invalido!");
161                 }
162             } else {
163                 System.err.println("Lista Vazia!");
164             }
165         }
166 
167     //Questão 10
168         public void atualizarTodos(Object dado) {
169             if (!estaVazia()) {
170                 NoDuplo ponteiroAuxiliar = ponteiroInicio;
171                 for (int i = 0; i < quantidade; i++) {
172                     ponteiroAuxiliar.setDado(dado);
173                     ponteiroAuxiliar = ponteiroAuxiliar.getProximo();
174                 }
175             } else {
176                 System.err.println("Lista Vazia!");
177             }
178         }
179 
180     //Questão 11
181         public Object apagarInicio() {
182             Object dadoTemporario = null;
183             if (!estaVazia()) {
184                 dadoTemporario = ponteiroInicio.getDado();
185                 ponteiroInicio = ponteiroInicio.getProximo();
186                 if (ponteiroInicio != null);
187                     ponteiroInicio.setAnterior(null);
188 
189                 quantidade--;
190             } else {
191                 System.err.println("Lista Vazia!");
192             }
193             return dadoTemporario;
194         }
195 
196     //Questão 12
197         public Object apagarFim(){
198             Object dadoTemporario = null;
199             if (!estaVazia()) {
200                 dadoTemporario = ponteiroFim.getDado();
201                 ponteiroFim = ponteiroFim.getAnterior();
202                 if (ponteiroFim != null);
203                     ponteiroFim.setProximo(null);
204 
205                 quantidade--;
206             } else {
207                 System.err.println("Lista Vazia!");
208             }
209             return dadoTemporario;
210         }
211 
212     //Questão 13
213         public Object apagar(int posicao) {
214             Object dadoTemporario = null;
215             if (!estaVazia()) {
216                 if (posicao >=0 && posicao < quantidade) {
217                     NoDuplo ponteiroAuxiliar = ponteiroInicio;
218                     for (int i = 0; i < posicao; i++)
219                         ponteiroAuxiliar = ponteiroAuxiliar.getProximo();
```

```
220
221                 dadoTemporario = ponteiroAuxiliar.getDado();
222
223                 NoDuplo ponteiroAnterior = ponteiroAuxiliar.getAnterior();
224                 NoDuplo ponteiroProximo = ponteiroAuxiliar.getProximo();
225
226                 if (ponteiroAnterior != null)
227                     ponteiroAnterior.setProximo(ponteiroProximo);
228                 else
229                     ponteiroInicio = ponteiroInicio.getProximo();
230
231                 if (ponteiroProximo != null)
232                     ponteiroProximo.setAnterior(ponteiroAnterior);
233                 else
234                     ponteiroFim = ponteiroFim.getAnterior();
235
236                 quantidade--;
237             } else {
238                 System.err.println("Indice Invalido!!");
239             }
240         } else {
241             System.err.println("Lista Vazia!");
242         }
243         return dadoTemporario;
244     }
245
246 //Questão 14
247     public Object[] apagarTodos(){
248         Object[] dadosTemp = new Object[quantidade];
249         if (!estaVazia()) {
250             NoDuplo ponteiroAuxiliar = ponteiroInicio;
251             for (int i = 0; i < quantidade; i++) {
252                 dadosTemp[i] = ponteiroAuxiliar.getDado();
253                 ponteiroAuxiliar = ponteiroAuxiliar.getProximo();
254             }
255
256             ponteiroInicio = null;
257             ponteiroFim = null;
258             quantidade = 0;
259
260         } else {
261             System.err.println("Lista Vazia!");
262         }
263         return dadosTemp;
264     }
265
266 //Questão 15
267     public void ordenarCrescente() {
268         if (!estaVazia()) {
269             for(int i = 0; i < quantidade-1; i++){
270                 NoDuplo atual = ponteiroInicio;
271                 NoDuplo proximo = atual.getProximo();
272                 for(int j = 0; j < quantidade-1; j++){
273                     if((Integer) atual.getDado() > (Integer) proximo.getDado()){
274                         Object aux = atual.getDado();
275                         atual.setDado(proximo.getDado());
276                         proximo.setDado(aux);
277                     }
278                     atual= atual.getProximo();
279                     proximo = atual.getProximo();
280                 }
281             }
282         } else {
283             System.err.println("Lista Vazia!");
284         }
285     }
286
287 //Questão 16
288     public void ordenarDecrescente() {
289         if (!estaVazia()) {
290             for(int i = 0; i < quantidade-1; i++){
291                 NoDuplo atual = ponteiroInicio;
292                 NoDuplo proximo = atual.getProximo();
```

```
                for(int j = 0; j < quantidade-1; j++){
                    if((Integer) atual.getDado() < (Integer) proximo.getDado()){
                        Object aux = atual.getDado();
                        atual.setDado(proximo.getDado());
                        proximo.setDado(aux);
                    }
                    atual= atual.getProximo();
                    proximo = atual.getProximo();
                }
            }
        } else {
            System.err.println("Lista Vazia!");
        }
     }

// Questão 17
   public void inverter(){
        if (!estaVazia()) {
            NoDuplo ponteiroAuxInicio = ponteiroInicio;
            NoDuplo ponteiroAuxFim = ponteiroFim;
            for (int i = 0; i < quantidade/2; i++) {
                Object dadoAux = ponteiroAuxInicio.getDado();
                ponteiroAuxInicio.setDado(ponteiroAuxFim.getDado());
                ponteiroAuxFim.setDado(dadoAux);

                ponteiroAuxInicio = ponteiroAuxInicio.getProximo();
                ponteiroAuxFim = ponteiroAuxFim.getAnterior();
            }
        } else {
            System.err.println("Lista Vazia!");
        }
    }

// Questão 18
     public String imprimirDeInicioParaFim() {
        String resultado = "[";
        NoDuplo ponteiroAuxiliar = ponteiroInicio;
        for (int i = 0; i < quantidade; i++) {
        if (i == quantidade-1) {
                resultado += ponteiroAuxiliar.getDado();
            } else {
                resultado += ponteiroAuxiliar.getDado() + ",";
            }
            ponteiroAuxiliar = ponteiroAuxiliar.getProximo();
        }
        return resultado + "]";
     }

// Questão 19
     public String imprimirDeFimParaInicio() {
        String resultado = "[";
        NoDuplo ponteiroAuxiliar = ponteiroFim;
        for (int i = 0; i < quantidade; i++) {
            if (i == quantidade-1) {
                resultado += ponteiroAuxiliar.getDado();
            } else {
                resultado += ponteiroAuxiliar.getDado() + ",";
            }
            ponteiroAuxiliar = ponteiroAuxiliar.getAnterior();
        }
        return resultado + "]";
     }

// Questão 20
     public boolean existe(Object dado) {
        boolean existeDado = false;
        if (!estaVazia()) {
            NoDuplo temp = ponteiroInicio;
            while (temp != null) {
                if (temp.getDado().equals(dado)) {
                    existeDado = true;
                    break;
                }
```

```
                temp = temp.getProximo();
            }
        } else {
            System.err.println("Lista Vazia!");
        }
        return existeDado;
    }

//Questão 21
  public int primeiraOcorrencia(Object dado) {
        int ocorrencia = -1;
        if (!estaVazia()) {
            NoDuplo temp = ponteiroFim;
            int i = 0;
            while (temp != null) {
                if (temp.getDado().equals(dado)) {
                    ocorrencia = i;
                    break;
                }
                i++;
                temp = temp.getProximo();
            }
        } else {
            System.err.println("Lista Vazia!");
        }
        return ocorrencia;
    }

//Questão 22
  public int ultimaOcorrencia(Object dado) {
        int ocorrencia = quantidade;
        if (!estaVazia()) {
            NoDuplo temp = ponteiroFim;
            int i = quantidade-1;
            while (temp != null) {
                if (temp.getDado().equals(dado)) {
                    ocorrencia = i;
                    break;
                }
                i--;
                temp = temp.getAnterior();
            }
        } else {
            System.err.println("Lista Vazia!");
        }
        return ocorrencia;
    }

//-----------------------------------------------
    private int quantidade;
    private int tamanho;
    private NoDuplo ponteiroInicio;
    private NoDuplo ponteiroFim;

    public Gabarito2() { this(10); }

    public Gabarito2(int tamanho) {
        quantidade = 0;
        this.tamanho = tamanho;
        ponteiroInicio = null;
        ponteiroFim = null;
    }

    public boolean estaCheia() { return (quantidade == tamanho); }
    public boolean estaVazia() { return (quantidade == 0); }
}
```