PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Curso: Engenharia de Software
Algoritmos e Estruturas de Dados II – 2º semestre de 2019
Professor: Rodrigo Richard Gomes
Aluna: Jaqueline de Sousa Queroz

PUC Minas

Exercícios Escolhidos 1, 2, 3, 6, 8, 9, 11, 12, 16 e 25
Exercícios Obrigatórios **7, 10, 30, 31 e 32**

AEDII - Primeira Lista de Exercícios

1- Crie na CLista o método ***void InsereAntesDe(Object ElementoAInserir, Object Elemento)*** que insere o ***ElementoAInserir*** na posição anterior ao ***Elemento*** passado por parâmetro.

```
   public void InsereAntesDe(Object ElementoAInserir, Object Elemento) {
      if (!contem(Elemento)) {
   System.out.println("O elemento [" +Elemento + "] nao consta na lista");
   } else {
      boolean achou = false;
      CCelula aux  = primeira;
      CCelula aux2 = primeira.prox;
      achou = aux2.item.equals(Elemento); //confere se o primeiro elemento é
o que procuramos, caso seja, não entra no while.
      while (aux2 != null && !achou) {
         aux    = aux.prox;
         aux2   = aux2.prox;
         achou = aux2.item.equals(Elemento);
      }
      aux.prox = new CCelula(ElementoAInserir, aux2);
      qtde++;
   }
   }
```

Após ser chamado para inserir o [19] antes do [15], o método verifica se o [15] está na lista. Caso não seja encontrado, é exibida a mensagem de que não contém o elemento.

```
----jGRASP exec: java Main
A lista L esta vazia! Qtde. elementos = 0
[/]->[5]->[9]->[14]->[12]->[2]->[5]->[6]->[3]->[2]->[3]->null
A lista L possui elementos! Qtde. elementos = 10
O elemento [15] nao consta na lista
[/]->[5]->[9]->[14]->[12]->[2]->[5]->[6]->[3]->[2]->[3]->null
A lista L possui elementos! Qtde. = 10

----jGRASP: operation complete.
```

Caso o [15] seja encontrado, o [19] é inserido antes ele:

```
----jGRASP exec: java Main
A lista L esta vazia! Qtde. elementos = 0
[/]->[11]->[1]->[15]->[8]->[11]->[6]->[12]->[10]->[2]->[7]->null
A lista L possui elementos! Qtde. elementos = 10
[/]->[11]->[1]->[19]->[15]->[8]->[11]->[6]->[12]->[10]->[2]->[7]->null
A lista L possui elementos! Qtde. = 11

----jGRASP: operation complete.
```

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Curso: Engenharia de Software
Algoritmos e Estruturas de Dados II – 2º semestre de 2019
Professor: Rodrigo Richard Gomes
Aluna: Jaqueline de Sousa Queroz

PUC Minas

2- Crie na CLista o método ***void InsereDepoisDe(Object ElementoAInserir, Object Elemento)*** que insere o ***ElementoAInserir*** na posição anterior ao ***Elemento*** passado por parâmetro.

O enunciado indica para inserir na posição anterior ao elemento passado, mesmo o nome sendo InsereDepoisDe. Neste caso, o código permanece idêntico ao da questão anterior, mudando apenas o nome do método. Caso seja para inserir depois do elemento indicado, o código é o seguinte:

```
   public void InsereDepoisDe(Object ElementoAInserir, Object Elemento) {
   if (!contem(Elemento)) {
   System.out.println("O elemento [" +Elemento + "] nao consta na lista");
   } else {
      boolean achou = false;
      CCelula aux  = primeira;
      CCelula aux2 = primeira.prox;
      achou = aux2.item.equals(Elemento); //confere se o primeiro elemento é
o que procuramos, caso seja, não entra no while.
      while (aux != null && !achou) {
         aux   = aux.prox;
         aux2  = aux2.prox;
         achou = aux2.item.equals(Elemento);
      }
      aux = aux2.prox;
      aux2.prox = new CCelula(ElementoAInserir, aux);
      qtde++;
   }
   }
```

Caso o [15] não seja encontrado:

```
 ----jGRASP exec: java Main
A lista L esta vazia! Qtde. elementos = 0
[/]->[13]->[13]->[0]->[4]->[7]->[14]->[5]->[6]->[1]->[6]->null
A lista L possui elementos! Qtde. elementos = 10
O elemento [15] nao consta na lista
[/]->[13]->[13]->[0]->[4]->[7]->[14]->[5]->[6]->[1]->[6]->null
A lista L possui elementos! Qtde. = 10

 ----jGRASP: operation complete.
```

Caso o [15] seja encontrado, o [19] é inserido após ele:

```
 ----jGRASP exec: java Main
A lista L esta vazia! Qtde. elementos = 0
[/]->[2]->[12]->[11]->[11]->[15]->[7]->[14]->[6]->[0]->[3]->null
A lista L possui elementos! Qtde. elementos = 10
[/]->[2]->[12]->[11]->[11]->[15]->[19]->[7]->[14]->[6]->[0]->[3]->null
A lista L possui elementos! Qtde. = 11

 ----jGRASP: operation complete.
```

3- Crie na CLista o método ***void InsereOrdenado(int ElementoAInserir)*** que insere ***ElementoAInserir*** em ordem crescente (*perceba que para funcionar corretamente, todos os elementos precisarão, necessariamente, ser inseridos através desse método*).

```
public void InsereOrdenado(int ElementoAInserir) {
  if (vazia()) {
    insereComeco(ElementoAInserir);
  }
  else  {
      if (ElementoAInserir < (int)retornaPrimeiro()) {
      insereComeco(ElementoAInserir);
      }
      else {
         if (ElementoAInserir > (int) retornaUltimo()) {
            insereFim(ElementoAInserir);
         }
         else {
         CCelula aux = primeira.prox;
         CCelula aux2 = primeira.prox.prox;
             while (true) {
               if ((int) aux.item <= ElementoAInserir &&
   (int) aux2.item >= ElementoAInserir) {
                   aux.prox = new
                   CCelula(ElementoAInserir, aux2);
                   qtde++;
                   break;
               }
                aux = aux.prox;
                aux2 = aux2.prox;
            }
         }
      }
  }
}
```

Ao chamar o método para inserir ordenado, ele verifica se a lista está vazia. Se estiver, o elemento será o primeiro e não é necessário comparar com outros elementos, já que não existem. Na próxima execução a lista não estará vazia e os elementos serão comparados para saber onde dever ser inseridos para se manterem ordenados.

```
----jGRASP exec: java Main
A lista L esta vazia! Qtde. elementos = 0
[/]->[1]->[12]->[15]->[16]->[18]->[19]->[21]->[26]->[27]->[28]->null
A lista L possui elementos! Qtde. elementos = 10

----jGRASP: operation complete.
```

6- Crie a função ***CPilha ConcatenaPilha(CPilha P1, CPilha P2)*** que concatena as pilhas P1 e P2 passadas por parâmetro.

7- * A classe **RandomQueue** é uma Fila que retorna elementos aleatórios ao invés de sempre retornar o primeiro elemento. Crie a classe RandomQueue com os seguintes métodos:

```
class RandomQueue {
    RandomQueue() { } // Construtora – cria uma RandomQueue vazia
    bool IsEmpty() { } // Retorna true se a RandomQueue estiver vazia
    void Enqueue(Object item) { } // Adiciona um item
    Object Dequeue() { } // Remove e retorna um elemento aleatório da RandomQueue
    Object Sample() { } // Retorna um elemento aleatório sem remove-lo da RandomQueue
}
```

Exemplo de uso da classe RandomQueue:

```
RandomQueue RQ = new RandomQueue();
for(int i = 1; i <= 5; i++)
    RQ.Enqueue(i);
System.out.print("Remove e retorna um elemento qualquer = "+RQ.Dequeue());
System.out.print("\nRetorna um elemento sem remover = "+RQ.Sample());
```

```
class Main {
  public static void main(String[] args) {

  RandomQueue RQ = new RandomQueue();
   for(int i = 1; i <= 5; i++){
   RQ.Enqueue(i);
   }
   RQ.mostra();
   System.out.println("Remove e retorna um elemento qualquer = "+RQ.Dequeue());
   RQ.mostra();
   System.out.println("\nRetorna um elemento sem remover = "+RQ.Sample());
   RQ.mostra();
  }
}
```

Execução da classe Main:

```
----jGRASP exec: java Main
[ 1 2 3 4 5 ]
Remove e retorna um elemento qualquer = 2
[ 1 3 4 5 ]

Retorna um elemento sem remover = 4
[ 1 3 4 5 ]

----jGRASP: operation complete.
```

Classe RandomQueue na próxima página.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Curso: Engenharia de Software
Algoritmos e Estruturas de Dados II – 2º semestre de 2019
Professor: Rodrigo Richard Gomes
Aluna: Jaqueline de Sousa Queroz

PUC Minas

```
import java.util.Random;
class RandomQueue {
   private CCelula frente; // Celula cabeca.
   private CCelula tras; // Ultima celula.
   private int qtde;

   RandomQueue() {
      frente = new CCelula();
      tras = frente;
   } // Construtora - cria uma RandomQueue vazia

   boolean IsEmpty() {
      return frente == tras;
   } // Retorna true se a RandomQueue estiver vazia

   void Enqueue(Object item) {
      tras.prox = new CCelula(item);
      tras = tras.prox;
      qtde++;
   } // Adiciona um item

   Object Dequeue() {
      Object item = null;
      Random aleatorio = new Random();
      int valor = aleatorio.nextInt(5);
      if (valor==1) {
         return desenfileira();
      } else {
         CCelula c = frente.prox;
         for (int aux = 1; aux<(valor-1); aux++){
         c = c.prox;
         }
         item = c.prox.item;
         c.prox = c.prox.prox;
         qtde--;
         return item;
      }
    } // Remove e retorna um elemento aleatório da RandomQueue

   Object Sample() {
      Random aleatorio = new Random();
      int valor = aleatorio.nextInt(5);
      if (valor==1) {
         return peek();
      } else {
      CCelula c = frente.prox;
      for (int aux = 1; aux<valor; aux++){
      c = c.prox;
      }
      return c.item
      }
      } // Retorna um elemento aleatório sem removê-lo da RandomQueue
```

```
//METODOS ADICIONAIS
   public Object desenfileira()
   {
      Object item = null;
      if (frente != tras) {
         frente = frente.prox;
         item = frente.item;
         qtde--;
      }
      return item;
   }

   public void mostra() {
      System.out.print("[ ");
      for (CCelula c = frente.prox; c != null; c = c.prox)
         System.out.print(c.item + " ");
      System.out.println("] ");
   }

      public Object peek() {
      if (frente != tras)
         return frente.prox.item;
      else
         return null;
   }
}
```

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Curso: Engenharia de Software
Algoritmos e Estruturas de Dados II – 2º semestre de 2019
Professor: Rodrigo Richard Gomes
Aluna: Jaqueline de Sousa Queroz

PUC Minas

8- Crie na CListaDup o método ***int primeiraOcorrenciaDe(Object elemento)*** que busca e retorna o índice da primeira ocorrência do elemento passado por parâmetro. Caso o elemento não exista, sua função deve retornar um valor negativo. *Obs: considere que o primeiro elemento está na posição 1.*

```
public int primeiraOcorrenciaDe(Object elemento) {
   if (!contem(elemento)) {
      return -1;
   } else {
      int cont = 0;
      boolean achou = false;
      CCelulaDup aux = primeira;
      while (!achou) {
         aux = aux.prox;
         cont++;
         achou = aux.item.equals(elemento);
      }
      return cont;
    }
 }
```

Ao passar como parâmetro o número [6], o método retornará a posição em que a primeira ocorrência do elemento se encontra. Caso ele não estivesse na lista, o retorno seria -1.

```
----jGRASP exec: java Main

LISTA
==============
[/]->[6]->[2]->[6]->[3]->[6]->[4]->[6]->[5]->[6]->[6]->null
A primeira ocorrencia do numero 6 esta na 1a posicao
 ----jGRASP: operation complete.
```

9- Crie na CListaDup o método ***int ultimaOcorrenciaDe(Object elemento)*** que busca e retorna o índice da última ocorrência do elemento passado por parâmetro. Caso o elemento não exista, sua função deve retornar um valor negativo. *Obs: considere que o primeiro elemento está na posição 1.*

```
public int ultimaOcorrenciaDe(Object elemento) {
      if (!contem(elemento)) {
         return -1;
      } else {
         int cont = qtde;
         boolean achou = false;
         CCelulaDup aux = ultima;
         achou = aux.item.equals(elemento);//caso a última já seja
a célula desejada ele não precisa entrar no while
         while (!achou ) {
            aux = aux.ant;
            cont--;
            achou = aux.item.equals(elemento);
         }
         return cont;
       }
    }
```

Caso o elemento não esteja na lista, o método retorna -1. Caso ele esteja, o método verifica os elementos, começando da última posição.

```
 ----jGRASP exec: java Main

LISTA
==============
[/]->[6]->[2]->[6]->[3]->[6]->[4]->[6]->[5]->[6]->[6]->null
A ultima ocorrencia do numero 5 esta na 8a posicao
 ----jGRASP: operation complete.
```

PUC Minas

10- * **Deque** (Double-ended-queue) é um Tipo Abstrato de Dados (TAD) que funciona como uma Fila e como uma Pilha, permitindo que itens sejam adicionados em ambos os extremos. Implemente a classe Deque, usando duplo encadeamento, com os seguintes métodos:

```
class Deque {
    Deque() { } // Construtora – cria uma Deque vazia
    boolean isEmpty() { } // Retorna true se a Deque estiver vazia
    int size() { } // Retorna a quantidade de itens da Deque
    void pushLeft(Object item) { } // Adiciona um item no lado esquerdo da Deque
    void pushRight(Object item) { } // Adiciona um item no lado direito da Deque
    Object popLeft() { } // Remove e retorna um item do lado esquerdo da Deque
    Object popRight() { } // Remove e retorna um item do lado direito da Deque
    }
```

```
class Main {
  public static void main(String[] args) {

  Deque D = new Deque();

   System.out.println ("Inserindo a esquerda: ");
   for(int i = 1; i <= 5; i++){
   D.pushLeft(i);
   }
   D.imprime();

   System.out.println ("Inserindo a direita: ");
   for(int i = 1; i <= 5; i++){
   D.pushRight(i);
   }
   D.imprime();

   System.out.println ("Remover a esquerda: ");
   D.popLeft();
   D.imprime();

   System.out.println ("Remover a direita: ");
   D.popRight();
   D.imprime();
}
}
```

Execução da classe Main:

```
 ----jGRASP exec: java Main
Inserindo a esquerda:
[5 4 3 2 1 ]
Inserindo a direita:
[5 4 3 2 1 1 2 3 4 5 ]
Remover a esquerda:
[4 3 2 1 1 2 3 4 5 ]
Remover a direita:
[4 3 2 1 1 2 3 4 ]

 ----jGRASP: operation complete.
```

Classe Deque está na próxima página.

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Curso: Engenharia de Software
Algoritmos e Estruturas de Dados II – 2º semestre de 2019
Professor: Rodrigo Richard Gomes
Aluna: Jaqueline de Sousa Queroz

PUC Minas

```
public class Deque {
   private CCelulaDup primeira;
   private CCelulaDup ultima;
   private int qtde;

   public Deque() {
      primeira = new CCelulaDup();
      ultima = primeira;
   } // Construtora - cria uma Deque vazia

   public boolean isEmpty() {
      return primeira == ultima;
   } // Retorna true se a Deque estiver vazia

   public int size() {
      return qtde;
   } // Retorna a quantidade de itens da Deque

   public void pushLeft(Object item) {
      if (primeira == ultima) {
         ultima.prox = new CCelulaDup(item, ultima,null);
         ultima = ultima.prox;
      }
      else {
         primeira.prox = new CCelulaDup(item, primeira,primeira.prox);
         primeira.prox.prox.ant = primeira.prox;
      }
      qtde++;
      } // Adiciona um item no lado esquerdo da Deque

   public void pushRight(Object item) {
      ultima.prox = new CCelulaDup(item, ultima, null);
      ultima = ultima.prox;
      qtde++;
   } // Adiciona um item no lado direito da Deque

   public Object popLeft() {
      if (primeira != ultima) {
         CCelulaDup aux = primeira.prox;
         primeira = primeira.prox;
         primeira.ant = null;
         qtde--;
         return aux.item;
      }
      return null;
      } //Remove e retorna um item do lado esquerdo da Deque

   public Object popRight() {
      if (primeira != ultima) {
        CCelulaDup aux = ultima;
        ultima = ultima.ant;
        ultima.prox = null;
        qtde--;
        return aux.item;
     }
     return null;
     } // Remove e retorna um item do lado direito da Deque
```

```
//MÉTODO ADICIONAL PARA IMPRIMIR

   public void imprime() {
      CCelulaDup aux = primeira.prox;
      System.out.print("[");
      while (aux != null) {
        System.out.print(aux.item+" ");
        aux = aux.prox;
       }
       System.out.println("]");
      }
      }
```

11- Crie na CLista o método ***void RemovePos(int n)*** que remove o elemento na n-esima posição da lista.

```
     public void  RemovePos(int n){
     if ((n > 0) && (n <= qtde) && (primeira != ultima)) { // lista
não vazia e posição maior que 0
        CCelula aux = primeira.prox;
        if (n == 1) {//caso seja para remover o primeiro elemento,
muda o elemento apontado pela célula cabeça
         primeira.prox = primeira.prox.prox;
         } else {
           for (int i = 1; i < n - 1; i++, aux = aux.prox) ;
//encontra a n-ésima posição.
            aux.prox = aux.prox.prox; //faz o aux.prox apontar para o
prox.prox.
       }
       qtde--;
     }
   }
```

Caso peça para remover o primeiro elemento:

```
 ----jGRASP exec: java Main
[/]->[13]->[10]->[12]->[12]->[2]->[1]->[13]->[12]->[13]->[11]->null
A lista L possui elementos! Qtde. elementos = 10
[/]->[10]->[12]->[12]->[2]->[1]->[13]->[12]->[13]->[11]->null
A lista L possui elementos! Qtde. = 9

 ----jGRASP: operation complete.
```

Caso peça para remover o 10° elemento:

```
 ----jGRASP exec: java Main
[/]->[5]->[5]->[2]->[3]->[5]->[7]->[6]->[13]->[11]->[11]->null
A lista L possui elementos! Qtde. elementos = 10
[/]->[5]->[5]->[2]->[3]->[5]->[7]->[6]->[13]->[11]->null
A lista L possui elementos! Qtde. = 9

 ----jGRASP: operation complete.
```

12- Crie na CListaDup o método ***void RemovePos(int n)*** que remove o elemento na n-esima posição da lista.

```
       public void RemovePos(int n) {
       // Verifica se é uma posição válida e se a lista possui
elementos
       if ((n >= 1) && (n <= qtde) && (primeira != ultima)) {
          int i = 1;
          CCelulaDup aux = primeira.prox;
          while (i < n) {
             aux = aux.prox;
             i++;
          }
          aux.ant.prox = aux.prox;
          if (aux != ultima){
             aux.prox.ant = aux.ant;
             }  else {
             ultima = aux.ant;}
          qtde--;
       }
    }
```

Ao chamarmos o método para remover um elemento, ele realiza as verificações sobre o parâmetro passado. Ao passarmos o 3 como parâmetro, ele remove o terceiro elemento da lista:

```
----jGRASP exec: java Main

LISTA
==============
[/]->[6]->[2]->[6]->[3]->[6]->[4]->[6]->[5]->[6]->[6]->null

LISTA APOS REMOCAO DO ELEMENTO NA POSICAO 3
==============
[/]->[6]->[2]->[3]->[6]->[4]->[6]->[5]->[6]->[6]->null
```

16- Crie na CLista o método ***Object[] copiaParaVetor()*** que copia todos os elementos da Lista para um vetor.

```
  public Object[] copiaParaVetor() {
      Object vetor[];
      vetor = new Object[qtde]; //o vetor é criado com o tamanho
para a quantidade de itens da lista
      CCelula aux = primeira.prox;

      for(int i = 0; i < qtde; i++){
      vetor[i] = aux.item;
      aux  = aux.prox;
      }
      for(int j = 0; j < qtde; j++){//apenas para mostrar como ficou
o vetor
      System.out.print (vetor[j] + " - ");
      }
         return null;
   }
```

A variável aux já é inicializada com o primeiro item da lista, ignorando a célula cabeça e o resultado é o seguinte:

```
 ----jGRASP exec: java Main
[/]->[10]->[8]->[15]->[7]->[3]->[9]->[11]->[12]->[7]->[3]->null
A lista L possui elementos! Qtde. elementos = 10
10 - 8 - 15 - 7 - 3 - 9 - 11 - 12 - 7 - 3 -
 ----jGRASP: operation complete.
```

25- Crie na classe CFila o método ***void RemoverApos(Object item)***, o qual remove TODOS os elementos que seguem o item passado como parâmetro.

```
public void RemoverApos(Object item) {
      CCelula aux = frente.prox;
      boolean achou = false;
      while (aux != null && !achou) {
         achou = aux.item.equals(item);
         if (achou) {
            break;
         }
         aux = aux.prox;
      }
      aux.prox = null;
   }
```

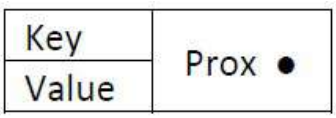
O método vai até o elemento passado e chegando nele, faz o elemento apontar para null. Ao chamar o método para remover o elemento [6], ele tem a seguinte saída:

```
----jGRASP exec: java Main
FILA ANTES DA REMOCAO
[ 0 2 4 6 8 10 12 14 16 18 ]
FILA APOS A REMOCAO
[ 0 2 4 6 ]
```

30- * Crie as classes **CCelulaDicionario** e **CDicionario** conforme a interface abaixo.

```
class CCelulaDicionario
{
      // Atributos
      public Object key, value;
      public CCelulaDicionario prox;

      // Construtora que anula os três atributos da célula
      public CCelulaDicionario()
      {
      }

      // Construtora que inicializa key e value com os argumentos passados
      // por parâmetro e anula a referência à próxima célula
      public CCelulaDicionario(Object chave, Object valor)
      {
      }

      // Construtora que inicializa todos os atribulos da célula com os argumentos
      // passados por parâmetro
      public CCelulaDicionario(Object chave, Object valor, CCelulaDicionario proxima)
      {
      }
}

class CDicionario
{
      private CCelulaDicionario primeira, ultima;

      public CDicionario()
      {
      }

      public boolean vazio()
      {
      }

      public void adiciona(Object chave, Object valor)
      {
      }

      public Object recebeValor(Object chave)
      {
      }
}
```

A classe CDicionario é muito semelhante a classe CLista. A principal diferença fica por conta da célula, que ao invés de ter apenas o valor do item e a referência para a próxima célula, tem também uma chave para valor adicionado.

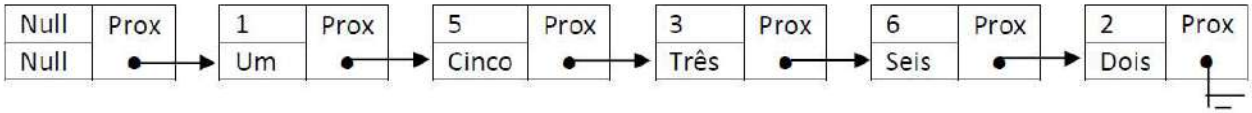


Algumas observações sobre sua classe:
• A construtora de sua classe CDicionario deve criar uma célula cabeça
• O método Adicionar deve adicionar o novo emento (chave/valor) na ultima posição do dicionário.

Atenção: sua classe não deve permitir a inserção de elementos com chaves duplicadas

- O método RecebeValor deve localizar e retornar o valor associado a chave passada por parâmetro. Caso a chave não exista, o método deve retornar null.

Exemplo de um **Dicionario** cuja chave é um número inteiro e o valor é o valor por extenso.



Agora usando sua classe **CDicionario,** crie um dicionário com URL's e IP's dos websites abaixo e mais 5 a sua escolha. O seu dicionário deve ser implementado usando a classe Hashtable e terá a URL como chave e o IP correspondente como valor (por exemplo, se digitarmos como chave a URL www.google.com, seu programa deve retornar o IP 74.125.234.81). O seu programa deve permitir que o usuário digite uma URL e deve imprimir o IP correspondente. Para descobrir o IP de um website, basta digitar **ping + URL do website** (exemplo: **ping www.google.com**).

www.google.com	www.yahoo.com	www.amazon.com	www.uol.com.br
www.pucminas.br	www.microsoft.com	research.microsoft.com	www.hotmail.com
www.gmail.com	www.twitter.com	www.facebook.com	www.cplusplus.com
www.youtube.com	www.brasil.gov.br	www.whitehouse.gov	www.nyt.com
www.capes.gov.br	www.wikipedia.com	www.answers.com	www.apple.com

```
import java.util.Scanner;
public class Hashtable {
   public static void main(String[] args) {
   CDicionario dicionario = new CDicionario();
   dicionario.adiciona("www.google.com", "172.217.5.100");
   dicionario.adiciona("www.yahoo.com", "98.138.219.231");
   dicionario.adiciona("www.amazon.com", "172.231.14.116");
   dicionario.adiciona("www.uol.com.br", "54.239.132.82");
   dicionario.adiciona("www.pucminas.br", "200.229.32.27");
   dicionario.adiciona("www.microsoft.com", "184.27.30.29");
   dicionario.adiciona("research.microsoft.com", "13.67.218.189");
   dicionario.adiciona("www.uol.com.br", "54.239.132.82");
   dicionario.adiciona("www.gmail.com", "172.217.0.37");
   dicionario.adiciona("www.twitter.com", "104.244.42.129");
   dicionario.adiciona("www.facebook.com", "157.240.22.35");
   dicionario.adiciona("www.cplusplus.com", "167.114.170.15");
   dicionario.adiciona("www.youtube.com", "172.217.164.110");
   dicionario.adiciona("www.brasil.gov.br",  "170.246.252.243");
   dicionario.adiciona("www.whitehouse.gov", "23.9.33.34");
   dicionario.adiciona("www.capes.gov.br", "200.130.18.222");
   dicionario.adiciona("www.whitehouse.gov", "23.9.33.34");
   dicionario.adiciona("www.nyt.com", "151.101.189.164");
   dicionario.adiciona("www.capes.gov.br", "200.130.18.222");
   dicionario.adiciona("www.wikipedia.com", "198.35.26.96");
   dicionario.adiciona("www.answers.com", "151.101.188.203");
   dicionario.adiciona("www.apple.com", "172.230.107.90");

   Scanner entrada = new Scanner (System.in);

   System.out.println ("Digite o site desejado: ");
   String site = entrada.next();
   System.out.println(dicionario.recebeValor(site));
   }
}
```

Saída caso o elemento não seja encontrado:

```
 ----jGRASP exec: java Hashtable
Digite o site desejado:
www.yahoo.com.br
null

 ----jGRASP: operation complete.
```

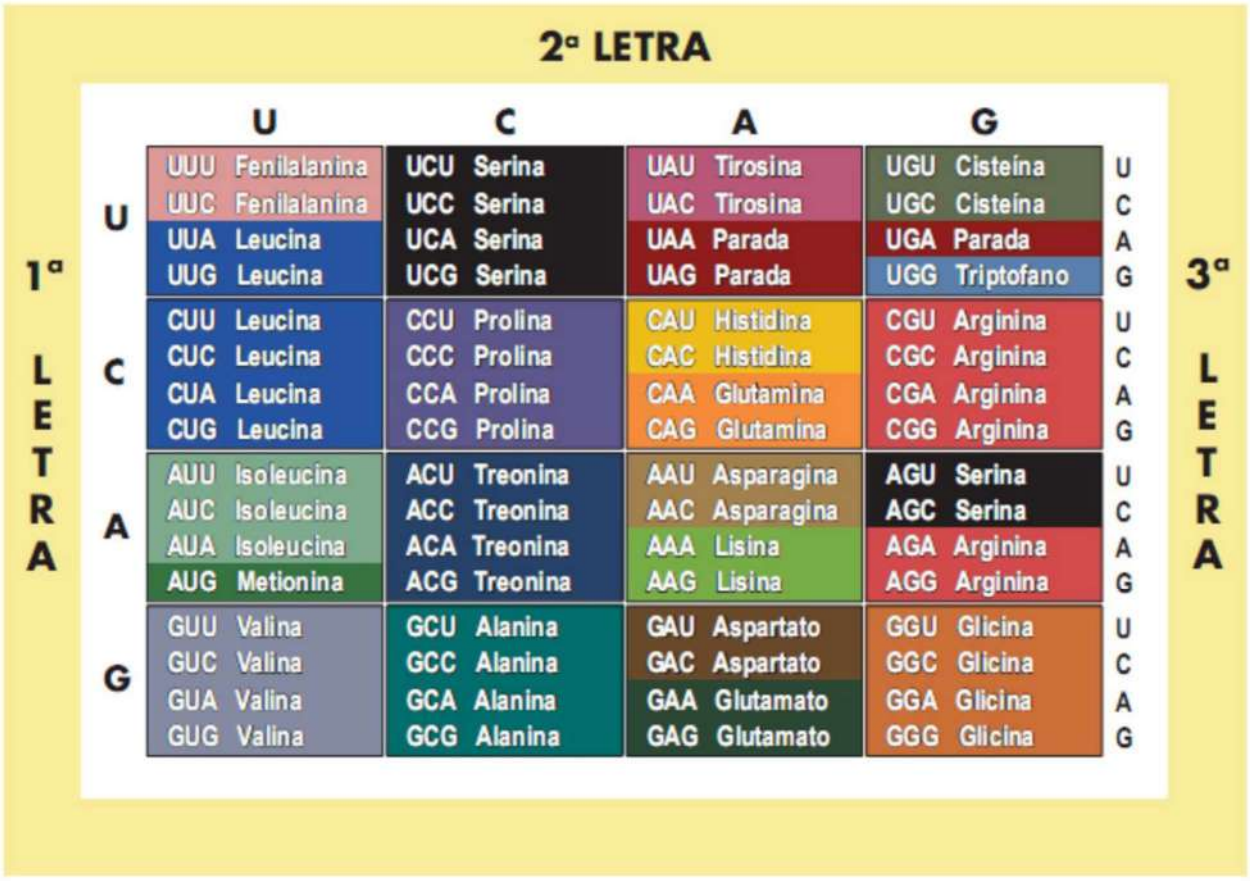
Saída caso o elemento seja encontrado:

```
 ----jGRASP exec: java Hashtable
Digite o site desejado:
www.amazon.com
172.231.14.116

 ----jGRASP: operation complete.
```

```
class CCelulaDicionario
   {
      // Atributos
      public Object key, value;
      public CCelulaDicionario prox;

      // Construtora que anula os três atributos da célula
      public CCelulaDicionario()
   {
      key = null;
      value = null;
      prox = null;

   }
      // Construtora que inicializa key e value com os argumentos
passados
      // por parâmetro e anula a referência à próxima célula
      public CCelulaDicionario(Object chave, Object valor)
   {
      this.key = key;
      this.value = value;
      this.prox = null;
   }
      // Construtora que inicializa todos os atribulos da célula
com os argumentos
      // passados por parâmetro
      public CCelulaDicionario(Object chave, Object valor,
CCelulaDicionario proxima)
   {
      this.key = chave;
      this.value = valor;
      this.prox = proxima;

   }
   }
```

```
class CDicionario
   {
      private CCelulaDicionario primeira, ultima;
      public CDicionario()
   {
      primeira = new CCelulaDicionario();
      ultima = primeira;
   }
      public boolean vazio()
   {
      return primeira == ultima;
   }
      public void adiciona(Object chave, Object valor)
   {
      ultima.prox = new CCelulaDicionario(chave, valor, null);
      ultima = ultima.prox;
   }
      public Object recebeValor(Object chave)
   {
      boolean achou = false;
      CCelulaDicionario aux1 = primeira.prox;
      Object aux2 =null;
      while (aux1 != null && !achou) {
         achou = aux1.key.equals(chave);
         aux2 = aux1.value;
         aux1 = aux1.prox;
      }
      if (achou) {
      return aux2;
      } else {
      return null;
      }
      }
   }
```

31- * Um biólogo precisa de um programa que traduza uma trinca de nucleotídeos em seu aminoácido correspondente. Por exemplo, a trinca de aminoácidos ACG é traduzida como o aminoácido Treonina, e GCA em Alanina. Crie um programa em Java que use a sua classe CDicionario para criar um dicionário do código genético. O usuário deve digitar uma trinca (chave) e seu programa deve mostrar o nome (valor) do aminoácido correspondente. Use a tabela a seguir para cadastrar todas as trincas/aminoácidos.



1ª LETRA	2ª LETRA: U	2ª LETRA: C	2ª LETRA: A	2ª LETRA: G	3ª LETRA
U	UUU Fenilalanina	UCU Serina	UAU Tirosina	UGU Cisteína	U
	UUC Fenilalanina	UCC Serina	UAC Tirosina	UGC Cisteína	C
	UUA Leucina	UCA Serina	UAA Parada	UGA Parada	A
	UUG Leucina	UCG Serina	UAG Parada	UGG Triptofano	G
C	CUU Leucina	CCU Prolina	CAU Histidina	CGU Arginina	U
	CUC Leucina	CCC Prolina	CAC Histidina	CGC Arginina	C
	CUA Leucina	CCA Prolina	CAA Glutamìna	CGA Arginina	A
	CUG Leucina	CCG Prolina	CAG Glutamina	CGG Arginina	G
A	AUU Isoleucina	ACU Treonina	AAU Asparagina	AGU Serina	U
	AUC Isoleucina	ACC Treonina	AAC Asparagina	AGC Serina	C
	AUA Isoleucina	ACA Treonina	AAA Lisina	AGA Arginina	A
	AUG Metionina	ACG Treonina	AAG Lisina	AGG Arginina	G
G	GUU Valina	GCU Alanina	GAU Aspartato	GGU Glicina	U
	GUC Valina	GCC Alanina	GAC Aspartato	GGC Glicina	C
	GUA Valina	GCA Alanina	GAA Glutamato	GGA Glicina	A
	GUG Valina	GCG Alanina	GAG Glutamato	GGG Glicina	G

```
import java.util.Scanner;
public class Hashtable {
   public static void main(String[] args) {
   CDicionario dicionario = new CDicionario();

      dicionario.adiciona("UUU", "Fenilalanina");
      dicionario.adiciona("UUC", "Fenilalanina");

      dicionario.adiciona("UUA", "Leucina");
      dicionario.adiciona("UUG", "Leucina");
      dicionario.adiciona("CUU", "Leucina");
      dicionario.adiciona("CUC", "Leucina");
      dicionario.adiciona("CUA", "Leucina");
      dicionario.adiciona("CUG", "Leucina");

      dicionario.adiciona("AUU", "Isoleucina");
      dicionario.adiciona("AUC", "Isoleucina");
      dicionario.adiciona("AUA", "Isoleucina");

      dicionario.adiciona("AUG", "Metionina");

      dicionario.adiciona("GUU", "Valina");
      dicionario.adiciona("GUC", "Valina");
      dicionario.adiciona("GUA", "Valina");
      dicionario.adiciona("GUG", "Valina");

      dicionario.adiciona("UCU", "Serina");
      dicionario.adiciona("UCC", "Serina");
      dicionario.adiciona("UCA", "Serina");
      dicionario.adiciona("UCG", "Serina");

      dicionario.adiciona("CCU", "Prolina");
      dicionario.adiciona("CCC", "Prolina");
      dicionario.adiciona("CCA", "Prolina");
      dicionario.adiciona("CCG", "Prolina");

      dicionario.adiciona("ACU", "Treonina");
      dicionario.adiciona("ACC", "Treonina");
      dicionario.adiciona("ACA", "Treonina");
      dicionario.adiciona("ACG", "Treonina");
     //INSERIR O RESTANTE
   Scanner entrada = new Scanner (System.in);

   System.out.println ("Digite a trinca correspondente: ");
   String trinca = entrada.next();
   System.out.println(dicionario.recebeValor(trinca));
   }
}
```

A alteração em relação a questão anterior é apenas no que diz respeito ao conteúdo do dicionário.

Caso a chave não seja encontrada:

```
----jGRASP exec: java Hashtable
Digite a trinca correspondente:
BBB
null

----jGRASP: operation complete.
```

Caso seja encontrada:

```
----jGRASP exec: java Hashtable
Digite a trinca correspondente:
UCU
Serina

----jGRASP: operation complete.
```

32- * Crie a classe **CListaSimples** que é uma lista simplesmente encadeada sem célula cabeça e que possui apenas os métodos definidos na interface abaixo. **Atenção: não podem ser acrescentados novos atributos ou métodos às classes CListaSimples e/ou CCelula abaixo.**

```
class CCelula
{
    public int item;
    public CCelula prox;
}

class CListaSimples
{

    private CCelula primeira, ultima;

    public CListaSimples()
    {
        // Código da função construtora
    }

    public bool vazia()
    {
        // Código para verificar se a Lista está vazia
    }

    public void insereComeco(Object valorItem)
    {
        // Código para inserir valorItem no início da Lista
    }

    public Object removeComeco()
    {
        // Código para remover e retornar o elemento do início da Lista
    }

    public void insereFim(Object valorItem)
    {
        // Código para inserir valorItem no fim da Lista
    }

    public Object removeFim()
    {
        // Código para remover e retornar o elemento do fim da Lista
    }

    public void imprime()
    {
        // Código para imprimir todos os elementos da Lista
    }

    public bool contem(Object elemento)
    {
        // Código para verifica se a Lista contem o elemento passado
        // como parâmetro
    }
}
```

PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS
Curso: Engenharia de Software
Algoritmos e Estruturas de Dados II – 2º semestre de 2019
Professor: Rodrigo Richard Gomes
Aluna: Jaqueline de Sousa Queroz

PUC Minas

```
class CListaSimples {
   private CCelula primeira, ultima;

   public CListaSimples() {// Código da função construtora
      primeira = ultima;
      ultima = primeira;
   }

   public boolean vazia() {// Código para verificar se a Lista está vazia
      return primeira == ultima;
    }

  public void insereComeco(Object valorItem) {// Código para inserir valorItem no início da Lista
      primeira = new CCelula (valorItem, primeira);
      if (primeira.prox == null){
         ultima = primeira.prox;
         }
      }

   public Object removeComeco() {// Código para remover e retornar o elemento do início da Lista
      if (primeira != ultima) {
         CCelula aux = primeira.prox;
         primeira.prox = aux.prox;
      if (primeira.prox == null)
         ultima = primeira;
         return aux.item;
         }
         return null;
      }

      public void insereFim(Object valorItem){ // Código para inserir valorItem no fim da Lista
      ultima.prox = new CCelula(valorItem, null);
      ultima = ultima.prox;
   }

      public Object removeFim() { // Código para remover e retornar o elemento do fim da Lista
      if (primeira != ultima) {
         CCelula aux = primeira;
         while (aux.prox != ultima)
            aux = aux.prox;
         CCelula aux2 = aux.prox;
         ultima = aux;
         ultima.prox = null;

         return aux2.item;
      }
      return null;
   }

      public void imprime() { // Código para imprimir todos os elementos da Lista
      CCelula aux = primeira.prox;
      while (aux != null) {
         System.out.println(aux.item);
         aux = aux.prox;
      }
   }

      public boolean contem(Object elemento) {// Código para verifica se a Lista contem o elemento
passado como parâmetro
      boolean achou = false;
      CCelula aux = primeira.prox;
      while (aux != null && !achou) {
         achou = aux.item.equals(elemento);
         aux = aux.prox;
      }
      return achou;
   }

}
```

```
class CCelula
{
   public Object item;
   public CCelula prox;

   public CCelula(Object valorItem, CCelula proxCelula)
    {
        item = valorItem;
        prox = proxCelula;
    }
}
```

Exemplo de execução chamando os métodos imprime e insereComeco:

```
----jGRASP exec: java Main
9 8 7 6 5 4 3 2 1
10 9 8 7 6 5 4 3 2 1
----jGRASP: operation complete.
```