1) Implemente uma função que receba uma lista duplamente encadeada L e um dado X como parâmetros de entrada. O objetivo é descobrir se o dado X pertence à lista L, e em "qual posição" da lista L. Caso X não seja encontrado, retornar a posição 0 (zero) como resposta (2,5).

   Obs.: *Considere o (primeiro) nó apontado por L como a posição 1, o próximo como sendo posição 2, e assim por diante*

Questão 2 (3,3)

Considere uma Lista Dinâmica Duplamente Encadeada. **Implemente um método** que receba um dado `valor` como parâmetro de entrada e **retorne a posição** do mesmo na lista (o objetivo é descobrir se `valor` pertence à lista, e em "qual posição" da lista). Caso `valor` não seja encontrado, retornar a posição 0 (zero) como resposta.

Obs.: Considere o primeiro nó apontado como a posição 1, o próximo como sendo posição 2, e assim por diante

Questão 4 (2,5)

Implemente um método que receba um dado `valor` como parâmetro de entrada. O objetivo é descobrir se `valor` pertence à lista, e em "qual posição" da lista. Caso `valor` não seja encontrado, retornar a posição 0 (zero) como resposta.

Obs.: Considere o (primeiro) nó apontado como a posição 1, o próximo como sendo posição 2, e assim por diante

Questão 3 – o presente de aniversário da Camila (0,5 ponto)

Implemente o método imprimirInverso() de uma Lista Dinâmica Simplesmente Encadeada (LDSE). Através deste método, os valores deverão ser impressos na ordem inversa em que encontram na lista, ou seja, do último para o primeiro elemento.

Questão 4 – o bônus prometido (0,5 ponto)

Através de exemplos (representados também graficamente, ou seja, com desenhos), apresente a diferença entre a lista LDDE e LDSE para a implementação do método
`booleanRemoverDet(char valor)`

//se não houver retorna falso

Questão 2 (4,0)

Como ilustração do uso de listas duplamente encadeadas, consideremos a extensão da implementação de inteiros longos em lista de modo a incluir inteiros negativos e positivos. O nó descritor da lista representando um inteiro longo contém uma indicação de o inteiro ser positivo ou negativo. Para somar um inteiro positivo com um negativo, o menor valor absoluto precisa ser subtraído do maior valor absoluto e o resultado deve receber o sinal do inteiro com o maior valor absoluto. Por conseguinte, é necessário um método para verificar qual dos dois inteiros representados como listas circulares tem o maior valor absoluto.

O primeiro critério que pode ser usado para identificar o inteiro com maior valor absoluto é o tamanho dos inteiros (presumindo-se que eles não contenham zeros iniciais). A lista com mais nós representará o inteiro com o maior valor absoluto. Entretanto, contar o número de nós exige um percurso adicional na lista. Em vez de contar o número de nós, a contagem poderia ser mantida como parte do nó descritor e referenciada conforme a necessidade. Contudo, se ambas as listas tiverem o mesmo número de nós, o inteiro cujo dígito mais significativo for maior terá o maior valor absoluto. Se os dígitos iniciais de ambos os inteiros forem iguais, será necessário percorrer as listas a partir do dígito mais significativo até o menos significativo para determinar o maior número. Observe que esse percurso ocorre na direção oposta à do percurso usado na verdadeira adição e subtração de dois inteiros. Como precisamos percorrer as listas nos dois sentidos, são usadas listas duplamente ligadas para representar tais inteiros.

Considere o formato do nó descritor. Além de um ponteiro da direita e da esquerda, o descritor precisa conter o tamanho da lista e uma indicação de o número ser positivo ou negativo. Esses dois fragmentos de informação podem ser combinados num único inteiro cujo valor absoluto seja o tamanho da lista e cujo sinal seja o sinal do número sendo representado.

Implemente o TAD "Números Inteiros Longos" de acordo com as especificações acima. Para este TAD deve ser implementado o conjunto de funções como segue:

`construtor( )`
// cria um inteiro longo sem valor algum, inicialmente

`inserir(d)`
// insere, no inteiro longo, o dígito **d**. A inserção dos dígitos 1,2,3, nesta ordem, formariam o inteiro 123.

`soma(i1,i2)`
// soma os inteiros longos **i1** e **i2**, com resultado na lista (i = i1+i2)

`imprimir()`
// apresenta o inteiro

Outras funções, como descreve o texto acima, podem ser necessárias para a correta implementação deste TAD, da forma como é solicitado. Você deverá implementá-las também.

Questão 3 (2,0)

Com seu conhecimento sobre implementação de um programa para verificar se uma palavra é palíndrome ou não, será fácil responder ao que se pede:

Sem usar vetores, sem conhecimento do funcionamento interno da Lista (você não sabe se ela é estática ou dinâmica, quais são seus atributos, etc... ou seja, tudo "privado") e utilizando somente os métodos `inserirInicio(char valor)`, `removerInicio()`, `estahVazia()`, `prim()` de uma classe `Lista`, implemente um método `verifica(Stringstr)` que retorna o mesmo objeto str com o seu conteúdo invertido, de traz para frente.

Obs1: Você pode criar quantas listas se fizerem necessárias dentro deste método.
Obs2: Você pode acessar cada posição da String com o método `charAt(int i)`
Obs3: NÃO é para implementar os métodos da Lista. Eles já estão implementados. **Se você os implementar, você não obterá nenhum ponto desta questão.**

Questão 4 (1,0)

Explique os motivos da diferença de tamanho entre os códigos dos métodos `removerFim()` da LDSE e da LDDE.

Questão 1 (2,5 pontos)

Dois amigos encontram-se para um torneio de cartas mágicas. Cada um tem um baralho de cartascom monstros, contendo cada carta um número inteiro que indica a força de combate do personagem.
Cada jogador empilha "estrategicamente" n cartas (número igual para os dois jogadores). Odesenvolvimento de um jogo ocorre numa sequência de combates da seguinte forma. O jogadorA e o jogador B tiram a carta de topo do seu baralho.
• Se a carta do jogador A for mais poderosa do que a do jogador B, então o jogador A ganhaesse combate, e a carta do jogador B fica para o jogador A. Coloca na base do seu baralhoprimeiro a sua carta e depois a de B.
• Se a carta do jogador B tiver mais poder de ataque que a carta do jogador A, então B ganhaesse combate. Neste caso, é o jogador B que fica com a carta do jogador A. Coloca na basedo seu baralho primeiro a sua carta e depois a de A.
• Se ambas as cartas tiverem o mesmo poder de ataque então ocorre um empate. Neste caso,cada jogador coloca a sua carta na base do seu baralho.
O jogo vai prosseguindo de modo análogo. A particularidade e interesse do jogo advém do fato dascartas serem mágicas. De tal forma que, quando duas cartas se encontram, independentemente doresultado do combate, o poder da carta é decrementado de um (porque o monstro fica cansado docombate e perde energia). Se o valor da carta chegar a 0 ela é retirada de jogo. Esta regra introduzinteresse no jogo, pois caso contrário o jogador com a carta mais alta nunca poderia perder o jogo.O jogador que durante uma sequência de combates perder todas as

cartas perde o jogo. Se ambos os jogadores ficarem sem cartas simultaneamente o jogotambém fica empatado.

Tarefa
Utilizando os métodos de listas (escolha a que lhe convier, mas lembre-se: a lista já está implementada; você somente utilizará seus métodos), implemente um programa que, dados os baralhos dos jogadores, simule o jogo para determinar qual dos dois jogadores ganha, conforme indicação abaixo.

Input
Na primeira linha digitada tem-se o número n de cartas com que jogarão (1 <= n <= 20). Nas linhas seguintestem-se a descrição das n cartas do baralho do jogador A (isto é, n inteiros que representam o poder dascartas escolhidas, do topo até a base). Nas n restantes, tem-se a descrição das n cartas do jogador B.

Output
O programa deve escrever A, B ou E, para indicar o jogador que ganha o jogo, seguido de mudançade linha. Escreverá E se tiver havido empate.

Exemplo 1
Input
3
3
5
2
2
1
9
Output
B

Exemplo 2
Input
5
4
2
1
6
8
3
3
4
2
1
Output
A

Questão 1 (valor 1,0)

Considerando uma Lista Dinâmica Duplamente Encadeada que possui os atributos quant, prim e ult, conforme descritos em aula, elabore o método `esvaziarLista( )`

Questão 2 (valor 1,0)

É possível implementar uma Lista Estática Duplamente Encadeada? Ilustre sua resposta através de figuras que apresentem a lista e o vetor que a armazena, de tal forma que justifique sua resposta (se **sim**, o vetor e a lista devem indicar esta possibilidade; se **não**, o vetor e a lista devem deixar claro porque é impossível tal implementação)

Questão 3 (valor 2,0)

Descreva o que faz o seguinte método de uma Lista Dinâmica Duplamente Encadeada:

```
public void fazAlgumaCoisa ( ){
        while(!ok){
intcont=0;
            No x = prim;
            do{
                if(x.getInfo() >x.getProx().getInfo()){
cont++;
                    char aux = x.getInfo();
x.setInfo(x.getProx().getInfo());
x.getProx().setInfo(aux);
                }
                if (cont ==0) ok = true;
                x = x.getProx();
            } while(x != ult);
}
}
```

Questão 4 (valor 2,0)

Em seu novo emprego, o seu superior imediato achou que, devido aos seus conhecimentos anteriores, sobre Listas, você teria como implementar uma Pilha rapidamente. Para ajudá-lo, ele passou as seguintes informações:

Pilhas são listas onde a **inserção** de um novo item ou a **remoção** de um item já existente se dá em **uma única extremidade**, no topo. Este critério de inserção/remoção é chamado de LIFO (Last-in, First-out / Último a entrar, Primeiro a sair).

Para implementarmos uma pilha, precisamos um atributo que faça referência ao topo, ***topo,*** indicando a extremidade dessa lista linear onde ocorrerão as operações de inserção e retirada de nós.

Para auxiliar mais um pouco, ele encontrou na Internet e lhe repassou estas figuras, que pretendem ser um exemplo didático de inserção e retirada:

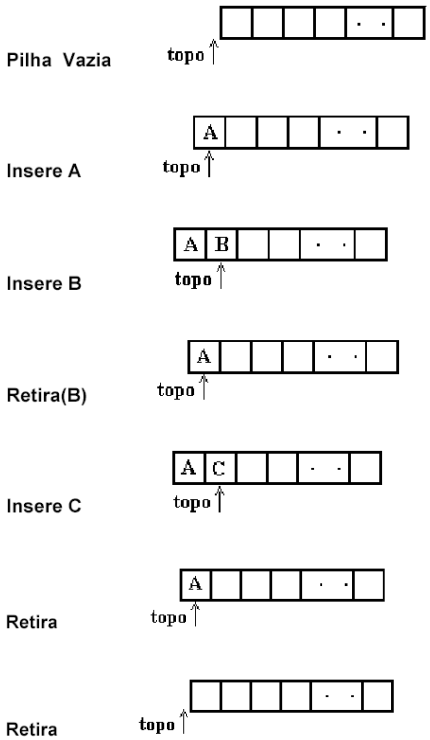


Figura 1 - Exemplo didático de inserção e retirada.

Operações Associadas:

1. `publicPilha( )` – construtor de uma Pilha vazia 0,1
2. `publicvoidpush(char x)` –**insere** x no topo da Pilha (ou seja, empilha) 0,4
3. `publicbooleanestahVazia( )` – testa se a Pilha está vazia, retornando TRUE ou FALSE 0,1
4. `public char topo( )` – **retorna** o elemento do topo da Pilha (sem eliminar) 0,4
5. `publicvoidpop( )` –**retira** o elemento do topo da Pilha (ou seja, desempilha) 0,4

Observação: Seu superior pediu que você **implemente as operações acima considerando que ele não sabe ao certo quanto** 0,1 **entos poderão ser armazenados**.

Assim, é sua tarefa apresentar o TAD Pilha, desde seus **atributos** até os 5 métodos descritos acima, que serão pontuados conforme a marcação indica. A qualidade da sua implementação no que tange ao cuidado com a mesma e a apresentação de um código "inteligente"e eficiente valem 0,5 pontos.

4. Uma aplicação popular para as filas é o desenvolvimento de aplicações baseadas em eventos. Aplicações baseadas em eventos operam amplamente sob a direção de ocorrências em tempo real, chamadas eventos. Em uma interface de uso gráfico desenvolvida em Java, TCL/TK ou C#, por exemplo, o comportamento de uma aplicação depende bastante dos toques de teclado, movimento do mouse e outros eventos determinados pelo usuário. Outros exemplos de aplicações baseadas em eventos ocorrem com freqüência em sistemas de controle, tais como aqueles encontrados em aeronaves e equipamentos de fábricas.

Em quase todas as aplicações baseadas em eventos, eles podem ocorrer a qualquer momento, por isso as filas têm papel importante na armazenagem de eventos até que uma aplicação esteja pronta para lidar com a questão. Uma fila funciona bem nesses casos porque as aplicações lidam com os eventos mais ou menos na mesma ordem em que eles ocorrem.

Sua tarefa é implementar duas funções para o tratamento de eventos: *receive_event* e *process_event*. Ambas as funções operam em uma fila contendo eventos do tipo *Event*. *Event*é definido em *event.h* que não é apresentado. Uma aplicação chama *receive_event* para enfileirar um evento sobre o qual tenha sido avisada. A forma exata através da qual uma aplicação é informada de um evento varia, porém os avisos freqüentemente começam com uma interrupção do hardware. É informado, como parâmetro do *receive_event*, o número do evento que foi ativado. Quando a aplicação decide que é hora de processar um evento chama *process_event*. Dentro de *process_event*um evento é retirado da fila de eventos e é passado para uma função que se encarrega de processá-lo em uma aplicação específica. Entenda essa chamada de função de processamento de evento como sendo a impressão do número do evento que foi retirado da fila.

Dica: Você pode entender as chamadas destas funções como sendo realizadas em um programa principal. A fila já está implementada em *fila.h* e você somente utilizará suas funções. Atenção: não é possível processar um evento se não houver algum já enfileirado (3,0).

1. Como ilustração do uso de listas duplamente ligadas, consideremos a extensão da implementação de inteiros longos em lista de modo a incluir inteiros negativos e positivos. O nó descritor da lista representando um inteiro longo contém uma indicação de o inteiro ser positivo ou negativo. Para somar um inteiro positivo com um negativo, o menor valor absoluto precisa ser subtraído do maior valor absoluto e o resultado deve receber o sinal do inteiro com o maior valor absoluto. Por conseguinte, é necessário um método para verificar qual dos dois inteiros representados como listas circulares tem o maior valor absoluto.

   O primeiro critério que pode ser usado para identificar o inteiro com maior valor absoluto é o tamanho dos inteiros (presumindo-se que eles não contenham zeros iniciais). A lista com mais nós representará o inteiro com o maior valor absoluto. Entretanto, contar o número de nós exige um percurso adicional na lista. Em vez de contar onúmero de nós, a contagem poderia ser mantida como parte do nó descritor e referenciada conforme a necessidade. Contudo, se ambas as listas tiverem o

mesmo número de nós, o inteiro cujo dígito mais significativo for maior terá o maior valor absoluto. Se os dígitos iniciais de ambos os inteiros forem iguais, será necessário percorrer as listas a partir do dígito mais significativo até o menos significativo para determinar o maior número. Observe que esse percurso ocorre na direção oposta à do percurso usado na verdadeira adição e subtração de dois inteiros. Como precisamos percorrer as listas nos dois sentidos, são usadas listas duplamente ligadas para representar tais inteiros.

Considere o formato do nó descritor. Além de um ponteiro da direita e da esquerda, o descritor precisa conter o tamanho da lista e uma indicação de o número ser positivo ou negativo. Esses dois fragmentos de informação podem ser combinados num único inteiro cujo valor absoluto seja o tamanho da lista e cujo sinal seja o sinal do número sendo representado.

Implemente o TAD "Números Inteiros Longos" de acordo com as especificações acima. Para este TAD deve ser implementado o conjunto de funções como segue:

`criar(i)`
// cria um inteiro longo sem valor algum, inicialmente

`inserir(i,d)`
// insere, no inteiro longo **i** o dígito **d**. A inserção dos dígitos 1,2,3, nesta ordem, formariam o inteiro 123.

`soma(i1,i2,r)`
// soma os inteiros longos **i1** e **i2**, com resultado em **r**

`imprimir(i)`
// apresenta o inteiro **i**

Outras funções, como descreve o texto acima, podem ser necessárias para a correta

implementação deste TAD, da forma como é solicitado. Você deverá implementá-

las também. Valor da questão (2,5)

2. Implementar a função `removerFim(lista *l)` baseada no tipo Lista Estática Encadeada. Apresente a função e a definição do tipo de dado.

   Valor da questão (0,5)

3. Implemente o TAD Fila, seguindo o esquema de uma fila dinâmica simplesmente encadeada circular. Nesta Fila o último elemento aponta para o primeiro elemento. Apresente a definição do tipo de dado e a implementação das funções `inserir`, `remover`, `criar`, `estahVazia`, `prim`.

   Valor da questão (1,5)

   1. Implemente uma função que receba uma lista duplamente encadeada L e um dado X como parâmetros de entrada. O objetivo é descobrir se o dado X

pertence à lista L, e em "qual posição" da lista L. Caso X não seja encontrado, retornar a posição 0 (zero) como resposta.

Obs.: Considere o (primeiro) nó apontado por L como a posição 1, o próximo como sendo posição 2, e assim por diante (2,5).

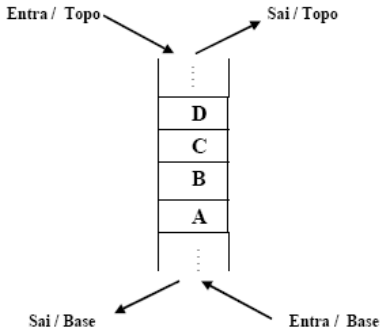
Questão 3 (valor 3,0)

As estruturas de dados podem ser organizadas de formas diferentes e especiais, de acordo com **a maneira com que os dados são inseridos e retirados**. Um tipo particular de estruturas de dados, é o **DEQUE**, onde a inserção pode ocorrer no início ou no final do deque (início/final do vetor) e a retirada dos dados também pode ocorrer no início ou no final do deque (início/final do vetor). Note que uma característica importante dos deques é que as operações de inserção e remoção de elementos ocorrem sempre nas extremidades, onde a princípio não devem ser realizadas inserções ou remoções de dados que se encontram no interior do mesmo.

Manipulação de Deques:

=> Entra = Insere no início ou final do deque (em uma das extremidades)

<= Sai = Retira do início ou final do deque (em uma das extremidades)



Implemente a classe Deque usando listas encadeadas. Esta classe deve conter os seguintes métodos:

voidaddTopo(valor) // adiciona valor no topo
voidaddBase(valor) // adiciona valor na base
removeTopo( ) // remove valor do topo
removeBack( ) // remove valor da base
chargetTopo( ) // retorna valor do topo
chargetBase( ) // retorna valor da base

1)

Implemente o método imprimirInverso( ) para imprimir os dados de uma lista dinâmica simplesmente encadeada do último para o primeiro elemento. (2,0)

2) Implemente o método intremoverElemento(char valor) que remove todas as ocorrências de valor da lista dinâmica duplamente encadeada retornando a quantidade de remoções efetuadas. (2,0)

3) Implemente o método intremoverElemento(char valor) que remove todas as ocorrências de valor da lista estática sequencial retornando a quantidade de remoções efetuadas. (2,0)

4) Quando é vantagem usar uma lista estática seqüencial ao invés de uma lista dinâmica? (1,0)

5) Quando é vantagem usar uma lista dinâmica duplamente encadeada no lugar de uma lista dinâmica simplesmente encadeada? (1,0)

Questão 1

Existe uma Estrutura de Dados chamada Fila, que possui o critério de inserção/remoção chamado FIFO (First In – First Out / Primeiro a Entrar – Primeiro a Sair). Esta Estrutura contém somente os métodos inserir (que insere ao fim da Fila), remover (que remove do início da Fila), primeiro (que retorna quem é o primeiro elemento da Fila), estahVazia (que retorna True se a Fila estiver vazia) e estahCheia (que retorna True se a Fila estiver cheia, e que é usada somente no caso de Fila estática).

Sabendo disso, e utilizando as Listas que você já conhece, responda as perguntas a seguir:

a) Para Fila Estática, qual seria o melhor método das Listas conhecidas (indique o método e a Lista a que pertence este método) que implementaria o método inserir facilmente e de forma otimizada? Justifique (0,5)
b) Para a Fila Estática, qual seria o melhor método das Listas conhecidas (indique o método e a Lista a que pertence este método) que implementaria o método remover facilmente e de forma otimizada? Justifique (0,5)
c) Para Fila Dinâmica, qual seria o melhor método das Listas conhecidas (indique o método e a Lista a que pertence este método) que implementaria o método inserir facilmente e de forma otimizada? Justifique (0,5)
d) Para a Fila Dinâmica, qual seria o melhor método das Listas conhecidas (indique o método e a Lista a que pertence este método) que implementaria o método remover facilmente e de forma otimizada? Justifique (0,5)