Algoritmos e Estruturas de Dados - 2016 – 2
Turma IA
Prof: Valério Rosset

Lista de Exercícios Complementar

(OBS. Para resolver todos os exercícios lembrem: 1. de não alterar a estrutura das TADs vistas em aula. 2. Implementar as funções separadas do programa principal. 3. Usar comentários para explicar partes principais do seu código.)

Atividades com Lista Estática Sequencial – LES

1) O governo de Urubuzolandia (um pequeno país ao norte do sul magnético do leste da África) iniciou a pouco tempo o novo programa de recenseamento da sua população. Mas como o governo não pretende mais utilizar os antigos formulários do censo resolveu investir numa aplicação que ajude aos recenseadores a realizar seu trabalho em campo. Para isso eles necessitam de uma aplicação muito simples e enxuta que possa ser utilizada para armazenar e calcular os seguintes dados: o número de homens, mulheres, crianças e idosos por cada casa, e endereço dessas pessoas e o trabalho que realizam e sua renda anual. Ao final desse processo o governo pretende que a aplicação os mostre o número de habitantes do país, a quantidade total de homens e mulheres, e também o número de famílias que possuem renda mensal abaixo do salario mínimo de 510,00 UR$. (Info: Alguns especialistas julgam que Urubuzolandia não tem mais de que 30 habitantes). Use a estrutura de uma lista estática sequencial para resolver esse problema, lembre que a estrutura da lista não deve ser alterada e feita a parte em um arquivo “.h”.

2) O DER tem por função controlar o fluxo de veículos pesados nas estradas brasileiras. Para isso em algumas estradas o DER obrigatoriamente faz a pesagem dos caminhões que estão em circulação. Para isso eles necessitam de uma aplicação que receba para cada caminhão pesado dados como, a placa, modelo, marca, capacidade máxima legal de carga em Kg e peso da carga medido. O DER multa os proprietários de cada veículo que ultrapassar 5 % da carga máxima permitida. Usando uma Lista ES Implemente um programa que calcule e apresente o valor da multa para cada caminhão que ultrapasse os limites legais de peso, também a quantidade de caminhões que ultrapassam o limite permitido e o total de multas arrecadado (diário) para uma rodovia onde o trafego de caminhões não ultrapasse 100 por dia. Use a estrutura de uma lista estática sequencial para resolver esse problema, lembre que a estrutura da lista não deve ser alterada e feita a parte em um arquivo “.h”.

3) A CBF precisa de sua ajuda para organizar a tabela do campeonato brasileiro, porém é preciso que a ordem dos confrontos não seja feita de maneira a favorecer nenhuma das equipes (ou seu time do coração). Assumindo que o campeonato conta com 20 equipes, escreva um programa que monte a tabela do campeonato com turno e returno onde a distribuição dos confrontos entre as equipes em cada rodada seja aleatório porém nunca

se repita em cada turno (o fator jogo fora ou em casa não precisa ser levado em conta). Depois de montada a tabela a CBF gostaria de visualizar, a seu critério, todos os confrontos de cada rodada um de cada vez. Use a estrutura de uma lista estática sequencial para resolver esse problema, lembre que a estrutura da lista não deve ser alterada e feita a parte em um arquivo ".h".

Atividades com Lista Estática Encadeada – LEE

4) Levando em consideração a seguinte estrutura de lista estática encadeada LEE (e as funções apresentadas em sala) implemente uma rotina de busca para localizar qualquer elemento dessa lista baseado no campo valor. O programa deve incluir as rotinas de inserção ao final da lista, inicialização e impressão (utilize uma biblioteca listaEE.h para implementar essas rotinas).

```
struct info{
    int valor;
    int prox;
};
typedef struct info tipoInfo;
```

```
struct listaEE {
   int tamanhoLista;
   int primeiro;
   int ultimo;
   int posLivre[N];
   tipoInfo elemento[N];
};
typedef struct listaEE tipoListaEE;
```

5) Implemente um programa para simular uma lista de compras de um site de vendas na internet utilizando uma lista estática encadeada (LEE). Cada produto será identificado por um código (*codigoProduto*) seguido por sua descrição e preço unitário. Os produtos disponíveis para compra deverão estar armazenados em um *vetor de estruturas a parte*. O programa deve permitir a inclusão de produtos na lista de compras bem como a remoção de qualquer um deles. Para cada lista de compras existe uma taxa fixa de postagem de U$ 15, mais o adicional por produto de U$ 5 caso o valor total dos produtos na lista for menor que U$ 50. Ao final o programa deve fechar a compra apresentando o total a ser pago incluindo as taxas bem como a lista de produtos comprados. (utilize uma biblioteca listaEE.h para implementar essas rotinas)

Atividades com Lista Dinâmicas Encadeadas – LDE e LDDE

6)Levando em consideração a seguinte estrutura de lista dinâmica encadeada LDE implemente uma ***rotina de busca*** para localizar qualquer elemento dessa lista. O programa deve incluir as funções de inserção ao final da lista e inicialização de lista por motivos de verificação. Utilize uma biblioteca listaLDE.h para implementar as rotinas de manipulação da lista.

```
struct elemento {
   int info;
   struct elemento *prox;
};

typedef struct elemento tipoElemento;
```

```
struct listaDE{
      tipoElemento *primeiro;
      tipoElemento *ultimo;
      int tamanhoLista;
};
typedef struct listaDE tipoLDE;
```

7)Utilizando a mesma estrutura do Exercício 6 escreva uma rotina que insira N números inteiros em uma lista LDE de maneira ordenada. Depois escreva uma nova rotina que inverta a ordem dos elementos da lista LDE sem alocar ou desalocar memória nem utilizar lista outra auxiliar (Dica: Use apenas os ponteiros e não copie/troque o conteúdo interno de cada elemento).

8) Escreva um programa que implemente uma estrutura de lista dinâmica encadeada, para manipular um registro de alunos. Cada aluno tem os seguintes atributos: Matricula, nome, curso e cidade. O programa devera permitir o ***cadastro desses alunos de maneira ordenada por matrícula***, sua eliminação, listagem geral e busca por nome ou matrícula. Utilize uma biblioteca listaLDE.h para implementar as rotinas de manipulação da lista.

9) A entrada e saída de uma sala de segurança do banco é controlada com o uso de um mecanismo eletrônico sem fios RFID (Radio-Frequency IDentification) que se comunica com uma interface de recepção de sinal junto a porta da sala. Cada pessoa autorizada a entrar nessa sala possuí um transmissor RFID, acoplado a seu crachá de identificação pessoal, que transmite a identificação do usuário. Toda vez que o usuário precisa entrar na sala, o sistema de recepção verifica se o ID enviado pelo dispositivo faz parte de uma lista de usuários autorizados e emite um sinal de POSITIVO ou NEGATIVO, destrancando ou não a porta. Quando ocorre um sinal POSITIVO são gravados o ID e o Nome do funcionário em uma lista de ocupantes da sala. Quando um dos ocupantes saí da sala protegida o caso contrário ocorre e o funcionário é excluído da lista de ocupantes. Escreva um programa que vai ser implantado no dispositivo de controle da porta da sala protegida para liberar a porta e gerenciar a lista de ocupantes da sala. Utilize uma biblioteca listaLDE.h para implementar as rotinas de manipulação da lista.

10) Levando em consideração a seguinte estrutura de LDDE (e as rotinas apresentadas em sala) implemente uma ***função de busca RECURSIVA*** para localizar qualquer elemento dessa lista. O programa deve incluir também as as rotinas de inserção ao final da lista e inicialização e impressão. Utilize uma biblioteca listaLDDE.h para implementar as rotinas de manipulação da lista.

```
struct elemento {
   int info;
   struct elemento *prox;
   struct elemento *ant;
};
typedef struct elemento tipoElemento;
```

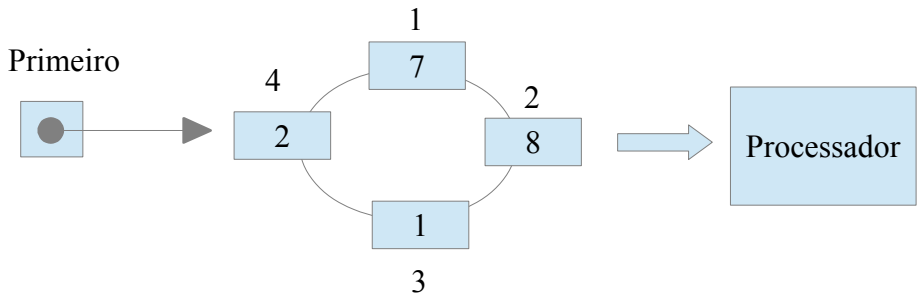
```
struct estruturaLDDE{
    tipoElemento *primeiro;
    tipoElemento *ultimo;
    int tamanhoLista;
};
typedef struct estruturaLDDE tipoLDDE;
```

11) Uma empresa de software precisa desenvolver um Navegador Web para um de seus clientes. Porém o seu programador chefe, que faltou as aulas de EDI, não sabe como implementar a recurso de histórico de páginas visitadas do novo navegador. Então você deverá implementar uma biblioteca LDDE que irá armazenar e manipular o histórico de endereços páginas visitadas para ser utilizada na implementação do novo navegador em questão. As rotinas necessárias para a sua implementação são: Inicialização, Inserção, Atualização (atualiza valor de um elemento). Utilize uma biblioteca listaLDDE.h para implementar as rotinas de manipulação da lista.

Atividades com Listas Circulares

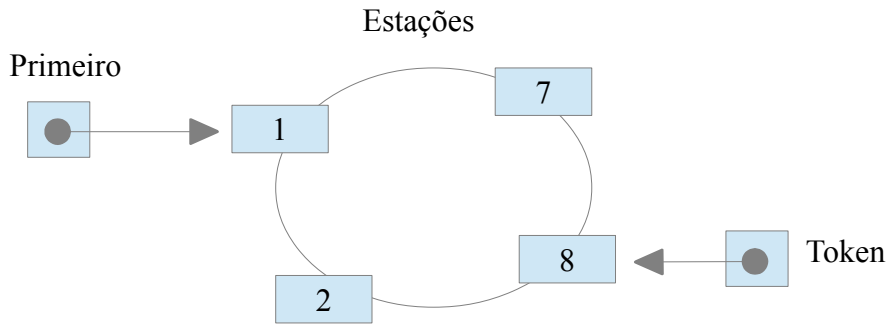
12) Utilize a mesma estrutura do exercício 10 para implementar uma busca RECURSIVA em uma Lista Circular Dinâmica Duplamente Encadeada – LCD-DE. A rotina de inserção (após o último) deverá ser adequada para a LCD-DE. Utilize uma biblioteca listaLCD-DE.h para implementar as rotinas de manipulação da lista.

13) Imagine um sistema operacional (SO) hipotético que escalona os processos a serem executados utilizando uma única LCD-DE (vide figura abaixo). Cada processo novo gerado pelo SO é incluído na posição imediatamente posterior ao último elemento da lista. Cada processo a ser executado carrega consigo um ID e um TE (tempo de execução em ciclos) aleatório. Como o escalonamento determina um tempo igual de execução no processador para cada processo, ao final de um ciclo de execução, todos os processos são atendidos pelo processador pelo menos uma vez, sendo subtraído de 1 seu TE. Quando um processo tem seu TE igual a zero, ele é removido da lista de escalonamento. Além disso o SO cria um novo processo exatamente a cada 3 ciclos de execução, incluindo esse novo processo na lista. Implemente um programa para simular o escalonamento de processos desse SO (assumindo que no primeiro ciclo já existam 3 processos). Esse programa deve ter uma rotina para controlar o ciclo onde cada elemento da lista será visitado simulando assim o processamento e executando as operações descritas anteriormente. O programa deve incluir as funções de inicialização, inserção e remoção de elementos da lista. O programa deve executar 50 ciclos e apresentar ao final o número de elementos na lista bem como o TE restante de cada elemento integrante da lista. Utilize uma biblioteca listaLCD-DE.h para implementar as rotinas de manipulação da lista.



14) Em uma rede sem fios hipotética o protocolo utilizado para controlar o acesso ao meio das estações de comunicação segue o modelo Token Ring. Nesse modelo cada elemento da rede tem o direito de comunicação fixo de 15 milissegundos. Esse direito de acesso ao meio é garantido por um Token (ou chave) que lhe é atribuído dentro de um ciclo de comunicação (partindo do primeiro elemento o Token passa para seu sucessor imediato e assim por diante, este processo leva 2 milissegundos). O elemento (estação) que possuí o Token é aquele que está efetivamente enviando informações pela rede. Cada elemento da rede (estação) é identificado por um ID (por exemplo Estação1, Estação2, etc). Como se trata de uma rede sem fios novos elementos podem entrar na rede aleatoriamente a cada ciclo, aumentando o tamanho do ciclo de comunicação. A cada ciclo de

comunicação o número de elementos que entram na rede varia de 1 a 5 elementos que são inseridos ao final da lista de passagem de Token. Escreva um programa para simular 50 ciclos de comunicação dessa rede e apresentar todas as estações presentes nessa rede e o tempo médio de espera de transmissão na rede após os 50 ciclos de comunicação. Utilize uma biblioteca listaLCD-DE.h para implementar as rotinas de manipulação da lista.



Atividades com **Filas, Filas Circulares e Filas Dinâmicas**

15) Levando em consideração a seguinte estrutura da FC e as rotinas apresentadas em sala, implemente uma ***função de remoção*** para a retirada de elementos dessa fila. O programa deve incluir as funções de inserção, inicialização e impressão por motivos de verificação. Utilize uma biblioteca filaFC.h para implementar as rotinas de manipulação da fila.

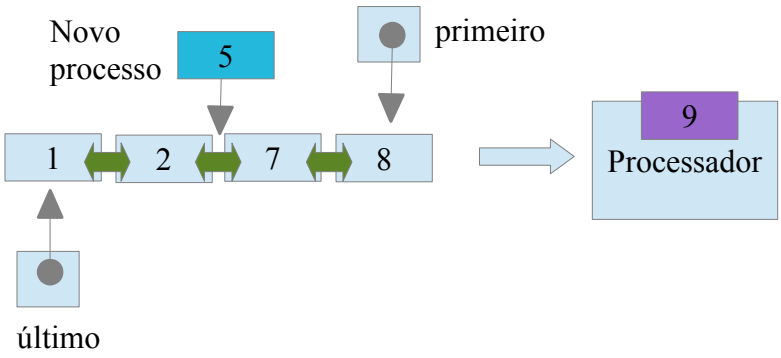
```
struct filaCircular {
   int primeiro;
   int ultimo;
   int elementos[tamanhoMAX];
};
typedef struct fila tipoFC;
```

16) Utilize a mesma estrutura do exercício 1 para implementar uma função de ***remoção com prioridade (baseado no maior valor da fila)***. O programa deve incluir as funções de inserção, inicialização e impressão por motivos de verificação.

17) Utilize a mesma estrutura do exercício 1 para implementar uma função de ***inserção com prioridade (baseado no maior valor)***. O programa deve incluir as funções de inicialização e impressão por motivos de verificação.

18) Implemente um programa usando uma fila com implementação dinâmica (FD) para simular uma fila de impressão de documentos levando em conta que a cada 3 minutos um número aleatório de documentos (entre 0 a 10) chegam ao buffer de entrada da impressora. Assuma que esse buffer é de tamanho limitado (máximo de 50 objetos) e cada objeto de impressão utiliza um tamanho máximo fixo e definido de memória. Assuma também que cada impressão leva entre 1 a 3 minutos. Simule o funcionamento dessa impressora por 60 minutos e gere um relatório de quantos trabalhos de impressão foram rejeitados durante esse período e quantos ainda estão na fila depois dos 60 min. Utilize uma biblioteca filaFD.h para implementar as rotinas de manipulação da fila.

19) Imagine um sistema operacional (SO) hipotético que escalona processos a serem executados em um processador utilizando uma única fila dinâmica. Cada processo novo gerado pelo SO é incluído na posição imediatamente posterior ao último elemento da fila. Cada processo a ser executado carrega consigo um ID, um TE (tempo de execução em ciclos) aleatório, e uma indicador de prioridade P. O escalonamento do SO determina que uma vez tomado o processador o mesmo só é liberado após todo o tempo de execução do processo ter terminado. Ao final de um ciclo de execução o processo atendido pelo processador tem subtraído de 1 seu TE. Quando um processo em execução tem seu TE igual a zero, o processador é liberado para o próximo processo que está na fila de espera. Porém, se existir algum outro processo recém criado com prioridade maior ao que está sendo executado, o processo tem seu processamento interrompido, sendo o mesmo realocado na fila dando a vez ao processo novo com maior prioridade. O SO cria um novo processo exatamente a cada 3 ciclos de execução, incluindo esse novo processo na fila de acordo com sua prioridade P. Implemente um programa para simular o escalonamento de processos desse SO. Esse programa deve ter uma função de ciclo onde cada elemento da lista será visitado simulando um processamento e executando as operações descritas anteriormente. O programa deve incluir as funções de inicialização, inserção e remoção de elementos da lista. O programa deve executar 50 ciclos e apresentar ao final o número de elementos na lista bem como o TE restante de cada elemento integrante da lista. Utilize uma biblioteca filaFD.h para implementar as rotinas de manipulação da fila.



20) Roteadores são elementos responsáveis por encaminhar pacotes em redes de comutação de pacotes. O roteador possui várias interfaces de entrada e saída de pacotes. O roteador implementa um buffer para cada porta de entrada, onde os pacotes que chegam vão se acumulando em ordem de chegada numa fila. É comum que roteadores formem filas grandes em seus buffers quando a taxa de chegada de pacotes supera a taxa de saída. Nesse caso, alguns pacotes que chegam não podem ser incluídos na fila porque as mesmas estão cheias (devido a pouca capacidade de armazenamento da memória do roteador), causando o chamado descarte de pacotes. Sua tarefa é simular o funcionamento de um roteador com 4 portas de entrada e 1 de saída em uma rede hipotética. Assuma que o roteador em questão tem capacidade de processar apenas 1 pacote por segundo, e a quantidade de pacotes que chegam a cada uma das portas do roteador varia de 1 a 10 por segundo. Cada pacote possui um ID e um campo de tamanho em bytes (variando de 500 a 1500 Bytes). Assuma que a memória disponível para cada buffer do roteador seja de 20 MBytes. Simule o funcionamento do roteador, e ao final de 1 hora determine total de pacotes descartados. Utilize uma biblioteca filaFD.h para implementar as rotinas de manipulação da fila.

21) Imagine um aeroporto que possui uma única pista para pousos e decolagens. O controle de voo precisa organizar os pousos e decolagens de maneira a não causar nenhum acidente. Está previsto que a cada 10 min uma aeronave deve decolar. A frequência de aeronaves que chegam para o pouso também é de 1 a cada 10 min, porém

as mesmas geralmente chegam com atrasos que variam de 1 a 40 min. O controle de uso da pista e é feito observando a regra de prioridade entre a fila de decolagem e a fila de pouso, onde a prioridade é dada ao maior tempo de espera. Caso o tempo de espera seja igual então é dada a prioridade para o pouso. Cada procedimento de pouso e decolagem sempre demora 3 a 9 min. Cada aeronave terá um ID e um tempo de espera que será incrementado a cada minuto a partir de sua entrada na fila de pouso ou decolagem. Além disso, é somado o atraso ao tempo de espera das aeronaves que entram na fila de pouso. Escreva um programa que controle os pousos e decolagens desse aeroporto durante 24 horas, ao fim apresente o atraso médio de espera em ambas as filas. Utilize uma biblioteca filaFD.h para implementar as rotinas de manipulação da fila.