**Universidade Federal de Mato Grosso**
**Instituto de Computação**

Disciplina: Algoritmos II
Professora: Vanessa de Oliveira Campos

## LISTA DE EXERCÍCIOS

## VARIÁVEIS E LISTAS DINÂMICAS

1. Faça o algoritmo de uma função lógica que inclua um elemento no final de uma lista.
2. Faça o algoritmo de um procedimento que exclua de uma lista de inteiros, todos os elementos que tenham o mesmo valor passado como parâmetro.
3. Faça o algoritmo de um procedimento que esvazie uma lista.
4. Faça o algoritmo de uma função que retorne a quantidade de elementos da lista.
5. Faça o algoritmo de uma função lógica que inclua um elemento na posição *i* de uma lista.
6. Faça o algoritmo de uma função que, a partir de duas listas simplesmente encadeadas, cria uma terceira lista que seja a união sem repetição dos elementos das duas primeiras.
7. Faça o algoritmo de uma função inteira que retorne a posição de um determinado elemento *x* real aparece na lista encadeada simples. Retorne -1 em caso de erro.
8. Faça o algoritmo de uma função real [iterativa/recursiva] para retornar a média aritmética simples dos valores inteiros de uma lista encadeada simples. Retorne 0 no caso da lista estar vazia.
9. Faça o algoritmo de uma função lógica para dizer se um determinado elemento *x* aparece junto de um elemento *y*, nesta ordem, na lista encadeada simples de reais.
10. Faça o algoritmo de uma função lógica [iterativa/recursiva] que diz se todos os elementos de uma lista simplesmente encadeada ordenada de reais encontram-se contidos em outra lista simplesmente encadeada também ordenada.
11. Faça o algoritmo de uma função lógica [iterativa/recursiva] para remover um elemento da posição p de uma lista [simplesmente/duplamente] encadeada de inteiros, retornando VERDADEIRO se o elemento foi removido com sucesso.
12. Faça o algoritmo de uma função lógica para inserir um elemento no início de uma lista duplamente encadeada.

13. Faça o algoritmo de uma função lógica para inserir um elemento inteiro na posição ordenada de uma lista duplamente encadeada que está ordenada.

14. Faça o algoritmo de uma função lógica [iterativa/recursiva] para remover todos os elementos que aparecem em uma lista [simplesmente/duplamente] encadeada *a* de uma outra lista b, o que equivale à operação $b-a$ em conjuntos, retornando VERDADEIRO se pelo menos um elemento foi removido com sucesso.

15. Faça o algoritmo de uma função lógica [iterativa/recursiva] para remover todos os elementos que aparecem após um elemento *a* e antes de um elemento *b* em uma lista simplesmente encadeada, retornando VERDADEIRO se pelo menos um elemento foi removido com sucesso.

16. Faça o algoritmo de uma função inteira [iterativa/recursiva] para remover todos os elementos duplicados que aparecem em uma lista simplesmente encadeada de reais ordenada de forma ascendente, retornando a quantidade de elementos (nós) efetivamente retirados.

17. Faça o algoritmo de um procedimento que faça com que o último elemento de uma lista simplesmente encadeada de reais, recebida por parâmetro, vá para a primeira posição da mesma.

18. Faça o algoritmo de um procedimento que faça com que o primeiro elemento de uma lista [simplesmente/duplamente] encadeada de reais vá para a última posição da mesma, modificando-a.

19. Faça o algoritmo de um procedimento [iterativo|recursivo] para inverter os elementos de uma lista simplesmente encadeada de caracteres passada como parâmetro. Por exemplo, se a lista entregue é {a,b,c,d,e} então a lista devolvida é {e,d,c,b,a}. Faça as devidas definições de tipo previamente à declaração da função.

20. Faça o algoritmo de uma função lógica para juntar dois elementos consecutivos de uma lista simplesmente encadeada de reais, das posições $p$ e $p+1$, somando seus valores, em um único nó. Caso alguma das posição não exista, nada deve ser feito. A função deve retornar VERDADEIRO apenas no caso da junção ter sido feita.

21. Faça o algoritmo de uma função ponteiro de duplicação de uma lista [simplesmente/duplamente] encadeada de reais, passada por parâmetro, retornando o ponteiro para a lista-cópia, ou NULO em caso de erro.

22. Faça um algoritmo para dividir uma lista [simplesmente/duplamente] encadeada em duas metades, formando duas listas encadeadas.

23. Dada uma lista simplesmente encadeada de inteiros, faça um algoritmo para criar outras duas que contenham, respectivamente, somente os números pares e somente os números ímpares, mantendo a ordem relativa original dos elementos nas novas listas criadas.

24. Faça o algoritmo de uma função inteira para remover todos os elementos duplicados que aparecem em uma lista simplesmente encadeada de reais ordenada de forma ascendente, retornando a quantidade de elementos (nós) efetivamente retirados.

25. Faça o algoritmo [iterativo|recursivo] de uma função lógica que receba duas listas encadeadas representando dois “conjuntos” de caracteres e devolva (em um terceiro parâmetro da função) uma terceira lista encadeada que represente a “união de conjuntos”. Por exemplo, se a primeira lista A tem os valores {a,b,c,d,e,f,g,h,i,j} e a segunda lista B tem os valores {a,d,e,h,p,q}, então a lista resultante criada A ∪ B terá os valores {a,b,c,d,e,f,g,h,i,j,p,q}. Considere que os valores encontram-se ordenados nas listas. A função deve retornar VERDADEIRO se, e apenas se, a lista foi criada com sucesso. Faça as devidas definições de tipo previamente à declaração da função.

26. Faça o algoritmo [iterativo|recursivo] de uma função lógica que receba duas listas encadeadas representando dois “conjuntos” de caracteres e devolva (em um terceiro parâmetro da função) uma terceira lista encadeada que represente a “intersecção de conjuntos”. Por exemplo, se a primeira lista A tem os valores {a,b,c,d,e,f,g,h,i,j} e a segunda lista B tem os valores {a,d,e,h,p,q}, então a lista resultante criada A∩B terá os valores {a,d,e,h}. Considere que os valores encontram-se ordenados nas listas. A função deve retornar VERDADEIRO se, e apenas se, a lista foi criada com sucesso. Faça as devidas definições de tipo previamente à declaração da função.

27. Faça o algoritmo [iterativo|recursivo] de uma função lógica que receba duas listas encadeadas representando dois “conjuntos” de caracteres e devolva (em um terceiro parâmetro da função) uma terceira lista encadeada que represente a “diferença de conjuntos”. Por exemplo, se a primeira lista A tem os valores {a,b,c,d,e,f,g,h,i,j} e a segunda lista B tem os valores{a,d,e,h}, então a lista resultante criada A−B terá os valores{b,c,f,g,i,j}. Considere que os valores encontram-se ordenados nas listas. A função deve retornar VERDADEIRO se, e apenas se, a lista foi criada com sucesso. Faça as devidas definições de tipo previamente à declaração da função.

28. Implemente uma lista circular ordenada duplamente encadeada que armazena em cada nó uma chave inteira e um nome. Faça um algoritmo para cada uma das seguintes operações:

    a) Buscar um nome, dado o valor da chave;
    b) Inserir um novo elemento na lista mantendo a ordem;
    c) Remover um elemento da lista;
    d) Imprimir os valores da lista;
    e) Copiar uma lista l1 para uma lista l2;
    f) Concatenar uma lista l1 com uma lista l2;
    g) Intercalar l1 e l2.

29. Escreva um programa que simule o controle de uma pista de decolagem de aviões em um aeroporto. Neste programa, o usuário deve ser capaz de realizar as seguintes tarefas:

    a) Listar o número de aviões aguardando na fila de decolagem;
    b) Autorizar a decolagem do primeiro avião da fila;
    c) Adicionar um avião à fila de espera;
    d) Listar todos os aviões na fila de espera;
    e) Listar as características do primeiro avião da fila.

Considere que os aviões possuem um nome e um número inteiro como identificador. Adicione outras características conforme achar necessário.

30. O Coeficiente de Rendimento Escolar (CRE) é calculado semestralmente e corresponde a média ponderada das notas finais obtidas em cada disciplina cursada, com aprovação ou não. O peso corresponde à carga horária total das disciplinas. O Coeficiente de Rendimento Escolar é determinado pela seguinte expressão:

$$CRE = \frac{(N_1 * H_1) + (N_2 * H_2) + (N_3 * H_3) + \ldots + (N_k * H_k)}{(H_1 + H_2 + H_3 + \ldots + H_k)}$$

onde $H_k$ é a carga horária da disciplina $k$ e $N_k$ é a nota da disciplina $k$.
Um professor do curso de computação gostaria de saber o CRE dos candidatos para oferecer uma bolsa ao melhor. Os alunos que se candidataram são de turmas diferentes e podem ter cursado uma quantidade diferente de disciplinas. Além disso, podem ter cursado uma quantidade diferente de semestres. Cada aluno entrega ao professor uma ficha semelhante a mostrada na tabela a seguir, com as informações de todos semestres que já cursou.

| **Nome** | | | |
|---|---|---|---|
| Fulano de Tal | | | |
| **RGA** | 201010000 | | |
| **Semestre** | **Disciplina** | **Nota** | **Carga Horária** |
| 1 | Cálculo I | 8,3 | 90 |
| 1 | Algoritmos I | 4,6 | 60 |
| 1 | Geometria Analítica | 8,0 | 60 |
| 2 | Calculo II | 7,1 | 90 |
| 2 | Algoritmos I | 8,6 | 60 |
| 2 | Física I | 6,6 | 60 |
| ... | ... | ... | ... |

Como cada aluno pode ter uma quantidade diferente de disciplinas, a quantidade de linhas desta ficha pode ser diferente. A partir dessas informações, elabore um algoritmo que leia, calcule e exiba ao final de todas as leituras as seguintes informações na ordem mostrada abaixo:
a) O CRE de cada aluno e o semestre que está cursando;
b) A quantidade de reprovações de cada aluno (alunos com nota menor do que 5 são considerados reprovados);
c) As disciplinas que ele reprovou no seu último semestre;
Estas informações devem ser exibidas na ordem em que aparecem. Utilize alocação dinâmica para resolver este problema.
Depois de exibir cada uma das informações, mostre o nome do melhor aluno que será o que tiver melhor CRE e menor quantidade de reprovações. Considere que estas duas situações nunca serão iguais para 2 alunos.

31. Um estudante deseja realizar um churrasco para comemorar sua colação de grau, ele não irá cobrar dos convidados, mas dos agregados destes convidados será cobrada uma taxa de acordo com a idade. Ele quer usar um algoritmo para calcular quanto cada convidado irá pagar pelos seus agregados e quanto irá precisar comprar de carne, cerveja e refrigerante. Para lhe ajudar ele pensou no seguinte registro para organizar sua lista de convidados e de seus agregados.

```
1 TIPO TCONVIDADO
2     NOME LITERAL
3     IDADE NUMÉRICO
4     BEBE_CERVEJA LÓGICO
5     QUANT_AGREGADOS NUMÉRICO
6     *LISTA_AGREGADOS TCONVIDADO
7     VALOR_AGREGADOS NUMÉRICO
8 FIMTIPO
```

Ele utiliza/armazena as mesmas informações que os convidados para gerenciar os agregados, mas um agregado não tem direito de convidar mais pessoas. Cada convidado tem direito de convidar quantos agregados quiser. Utilizando as tabelas mostradas abaixo, faça um algoritmo que

a) Pergunte ao estudantes, quantos convidados ele chamará para a festa

b) Pergunte as informações de cada convidado e seus agregados e calcule o custo dos agregados para cada convidado

c) Calcule a partir das informações dos convidados e agregados, quanto será necessário comprar de em litros de cerveja, litros de refrigerante e quilos de carne para o churrasco.

Obrigatoriamente, menores de 18 anos não bebem. Caso seja maior de 18 anos, pergunte se a pessoa beberá cerveja. Assuma que todos beberão refrigerante.

| Faixa etária | Quant. de carne |
|---|---|
| 0 a 5 | 100g |
| 6 a 12 | 300g |
| 13 a 23 | 450g |
| acima de 23 | 600g |

| Faixa etária | refrig | cerveja |
|---|---|---|
| 0 a 5 | 100ml | 0 ml |
| 6 a 12 | 500ml | 1200ml |
| 13 a 23 | 1000ml | 1500ml |
| acima de 23 | 800ml | 600ml |

| Faixa etária | Custo |
|---|---|
| 0 a 17 | R$0,00 |
| 18 a 23 | R$10,00 |
| 23 a 50 | R$40,00 |