MC202 — Estruturas de Dados
Lista de Exercícios 1

Primeiro semestre de 2017 - Turmas B e C
Professor: Emilio Francesquini
francesquini@ic.unicamp.br

1. Indique a diferença entre as seguintes atribuições:

```
char a;
a = '6';
a = 6;
```

2. Faça uma função recursiva em C que calcule o máximo divisor comum de dois números m, n. Você deve utilizar a seguinte regra do cálculo do mdc com $m \geq n$:

$$mdc(m, n) = m \text{ se } n = 0$$
$$mdc(m, n) = mdc(n, m\%n) \text{ se } n > 0$$

3. Implemente uma versão iterativa e outra recursiva de uma função que recebe como parâmetro um inteiro i e devolve como resultado o valor do i-ésimo elemento da sequência de Fibonacci.

4. Qual é a saída do programa abaixo?

```
void f1 (int a) {
    a = 2;
}

void f2 (int *a) {
    *a = 2;
}

int main (int argc, char ** argv) {
    int a = 0;
    f1(a);
    printf("%d\n", a);
    f2(&a);
    printf("%d\n", a);
}
```

5. Escreva uma função que dada uma lista ligada sem cabeça devolva o número de elementos da lista.

6. Escreva uma função que dado um vetor de inteiros devolva uma lista ligada de inteiros cujos elementos são apresentados na mesma ordem do vetor. Faça duas versões: uma iterativa e uma recursiva.

7. Escreva uma função que dada uma lista ligada de inteiros devolva um ponteiro para um vetor de inteiros cujos elementos são apresentados na mesma ordem da lista ligada dada. Faça duas versões: uma iterativa e uma recursiva.

8. Escreva uma função que dado uma lista ligada e um valor v remova da lista **todas** as células que contêm o valor v.

9. Escreva uma função que remova a k-ésima célula de uma lista encadeada sem cabeça. Faça duas versões: uma iterativa e uma recursiva.

10. Escreva uma versão de `No *selectionsort(No *ini)` que recebe uma lista ligada sem cabeça `ini` e devolve um ponteiro para uma lista ligada ordenada sem cabeça com os nós da lista original. Como você adaptaria este código caso a lista tivesse cabeça?

11. Escreva uma função `void mergesort(No *cabeca)` que recebe uma lista ligada com cabeça e devolve a lista ordenada para isto utilizando o algoritmo de ordenação MergeSort.

12. Escreva uma função `remove_llcc(No *ini, int k)` que recebe um ponteiro para uma lista ligada circular com cabeça `ini` e remove o primeiro nó com chave `k`. Use o método do sentinela.

13. Escreva uma função `remove_ldlcc(NoD *p)` que recebe um ponteiro para um nó `p` de uma lista duplamente ligada circular com nó cabeça e o remove da lista. Naturalmente, suponha que `p` não é o nó cabeça da lista.

14. Escreva uma função `insere_lcco(No *ini, int k)` que recebe uma lista ligada circular com cabeça ordenada `ini` e insere um novo nó com chave `k` na posição correta.

15. Problema de Josephus:

    - Um grupo de N pessoas precisa eleger um líder.
    - Decidiu-se usar a seguinte ideia para eleger um líder: forma-se um círculo com as N pessoas e escolhe-se um inteiro k. Começamos com uma pessoa qualquer e percorremos o círculo em sentido horário, eliminando cada k-ésima pessoa. A última pessoa que restar será o líder. Veja o verbete sobre Josephus na Wikipedia.

    Coloque os números 1, 2, ..., N em um círculo nesta ordem e começando em 1 aplique o algoritmo acima com um valor k. Determine o último número, denotado `J(N, k)`. Escreva uma função `Josephus(int N, int k)` que calcula `J(N, k)`.

16. Implemente uma pilha de tamanho limitado usando um vetor e em seguida implemente uma pilha de tamanho arbitrário usando listas ligadas. Sua implementação deve conter ao menos as funções `pop`, `push`, `peek` e `empty`.

17. Implemente uma fila de tamanho limitado usando um vetor e em seguida implemente uma fila de tamanho arbitrário usando listas ligadas. Sua implementação deve conter ao menos as funções `enqueue`, `dequeue`, `peek` e `empty`.

18. Considere o problema de verificar se uma sequência de parênteses e colchetes está balanceada. Por exemplo: "()", "[([])]", "((([([])])))" e "[]" são exemplos de sequências balanceadas enquanto ")(", "(]", "([)]" e "[[()])" não são. Escreva uma função que recebe uma sequência de parênteses e colchetes dada por uma string (`char *`) e que devolve 1 caso a sequência esteja balanceada ou 0 caso contrário.

19. [Feofiloff] Escreva uma função iterativa que simule o comportamento da seguinte função recursiva. Use uma pilha.

```
int TTT (int x[], int n) {
   if (n == 0) return 0;
   if (x[n] > 0) return x[n] + TTT (x, n-1);
   else return TTT (x, n-1);
}
```

20. Implemente uma função que inverta a ordem dos elementos de uma fila usando apenas a sua API (veja questão 17), ou seja, você não pode usar informações sobre a implementação da fila (se é feita usando listas ligadas ou vetores).

21. Implemente uma fila dinâmica (de tamanho arbitrário) que utiliza um vetor como base. (Dica: defina uma regra clara e simples a ser usada quando o vetor deverá crescer ou diminuir)

22. Escreva uma função que inverte a ordem dos elementos de uma lista encadeada com cabeça. Por exemplo, a lista: CABEÇA $\rightarrow 1 \rightarrow 2 \rightarrow 3 \rightarrow$ NULL deve ser transformada em CABEÇA $\rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow$ NULL. O seu algoritmo não pode alocar memória para armazenar novas células.

Para as questões relativas sobre árvores binárias considere a seguinte definição:

```
struct _NoArv {
    int valor;
    struct _NoArv *dir, *esq;
}
typedef struct _NoArv NoArv;
```

23. Escreva as funções `void preordem(NoArv *p)`, `void inordem(NoArv *p)` e `void posordem(NoArv *p)` que percorrem a árvore binária `p` recebida como parâmetro e imprimem o valor contido em cada célula.

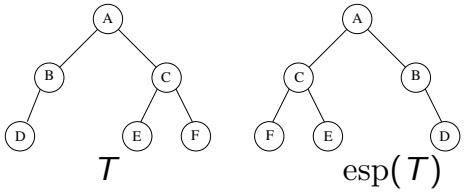
24. Escreva uma função `int tamanho(NoArv *p)` que recebe uma árvore binária `p` e devolve o número de nós de `p`.

25. Escreva uma função `int altura(NoArv *p)` que recebe uma árvore binária `p` e devolve a sua altura.

26. Suponha que você tem uma árvore binária com um campo adicional `pai`, mas apenas os campos `esq` e `dir` estão corretos. Escreva uma função `void papai(NoArv *p)` que recebe uma árvore binária deste tipo e atualiza o campo `pai` de cada um dos seus nós.

27. Escreva uma função void `minmax(NoArv *p, int *min, int *max)` que recebe uma árvore binária `p` e devolve em `min` e `max` o menor e o maior valores presentes na árvore.

28. O espelho esp(A) da árvore binária A é a árvore binária definida recursivamente da seguinte forma. Se A for vazia então esp(A) é a árvore vazia. Senão, se A tem raiz r, subárvore esquerda A_e e e subávore direita A_d, então esp(A) é a árvore binária com raiz r, subávore esquerda esp(A_d) e subárvore direita esp(A_e).



Escreva uma função `NoArv *espelho(NoArv *p)` que recebe uma árvore binária `p` e devolve seu espelho. A árvore original não deve ser modificada.

29. Podemos representar uma árvore binária quase completa por um vetor A[0 . . . n − 1]. Como seriam os índices dos filhos esquerdo e direito de um nó i? Quem seria o pai de i?

30. Um min-heap é uma árvore binária quase completa representada por um vetor A[1 . . . n] tal que A[i/2] A[i] (ou seja, pai filho) para todo nó i, i = 2, 3, . . . , n. Escreva o análogo da função Heapify para esta versão.

31. Escreva uma versão do Heapsort que ordena um vetor em ordem decrescente.

32. Insira as chaves 50, 30, 70, 20, 40, 60, 80, 15, 25, 35, 45, 36 nesta ordem em uma árvore de busca inicialmente vazia. Desenhe a árvore resultante. Remova o nó que tem chave 30 e desenhe a árvore de busca resultante.

33. Considere uma árvore binária de busca. Escreva uma versão recursiva e outra iterativa de uma função `NoArv *insere(NoArv *r, NoArv *novo)` que insere o novo nó na árvore mantendo as suas propriedades de busca.

Para as questões a seguir considere a remoção de uma raiz de árvore binária de busca descrita nas páginas 16 a 26 da Aula 12.

34. Na função remove raiz, em vez de colocar o sucessor de r como a nova raiz, pode-se escolher o predecessor de r. Escreva uma versão de remove raiz que faz isto.

35. Escreva uma versão recursiva da função remove raiz.

36. Considere árvores binárias em geral. Seja $n \geq 0$ um número inteiro. Prove que uma árvore binária com n nós tem altura mínima de $\lfloor lg\ n \rfloor$, onde:

    - $lg\ x$ indica $log_2 x$
    - $\lfloor x \rfloor$ é o piso de x