MAC122 – Princípios de Desenvolvimento de Algoritmos

DCC - IME - USP
2º Semestre de 2018
Prof.: Dr. Paulo Miranda
pmiranda@vision.ime.usp.br

Lista 03: Funções Recursivas

Parte A: Funções recursivas simples

Implemente as funções especificadas abaixo de forma recursiva:

1) `void ImprimePiramide(int n);`
Imprime uma pirâmide conforme o exemplo abaixo de forma recursiva onde o maior valor **n** é fornecido como argumento.
Ex: Para **n**=5 temos:

```
1
2 1
3 2 1
4 3 2 1
5 4 3 2 1
```

2) `void DigitosInvertidos(int n);`
Imprime os dígitos de um número inteiro **n** em ordem inversa.
Ex: Para **n**=123 deve imprimir 321.

3) `int ContaDigitos(int n);`
Retorna o número de dígitos do valor **n** fornecido.
Ex: Para **n**=450 a função deve retornar 3.

4) `float Potencia(float x, int n);`
Calcula $\mathbf{x}^{\mathbf{n}}$ para qualquer **x** real e **n** inteiro, inclusive para valores de **n** negativos.

Parte B: Funções recursivas com vetores

Problema 1: Implemente a função abaixo que calcula de forma recursiva a média dos elementos de um vetor de reais de tamanho **n**.

```
float CalculaMedia(float V[], int n);
```

Problema 2:

Escreva uma função recursiva que analisa os elementos de um vetor e retorna um dentre os seguintes códigos:

Código	Condição do vetor
0	Elementos desordenados
1	Elementos ordenados em ordem crescente
2	Elementos constantes
3	Elementos ordenados em ordem decrescente

```
int AnalisaVetor(int V[], int n);
```

Use o protótipo acima onde "n" indica o número de elementos presentes no vetor.

Exemplos:

Para `V={1,2,2,5,6,6,7,8,8,9}` retorna código `1.`
Para `V={20,15,11,10,8,8,5,2}` retorna código `3.`
Para `V={1,20,2,5,6,6,7,80,9}` retorna código `0.`
Para `V={8,8,8,8,8,8,8,8,8,8}` retorna código `2.`

Problema 3:

Escreva uma função recursiva que calcula o número de ocorrências de um dado valor em um vetor, ou seja, quantas vezes esse valor aparece no vetor.

```
int Frequencia(int a, int V[], int n);
```

Use o protótipo acima onde "n" indica o número de elementos presentes no vetor e "a" indica o valor que terá a sua frequência contada.

Exemplo: `Para V={3,3,5,3,2,2,3,3} e a=3 retorna 5.`

Problema 4:

A moda de um conjunto de dados é o valor que detém o maior número de observações, ou seja, o valor mais frequente (ou seja, o que "está na moda"...). No caso de empate qualquer um dos valores de frequência máxima pode ser considerado como sendo uma moda. Implemente uma função recursiva que calcula uma moda de um vetor de inteiros.

```
int Moda(int V[], int n);
```

Use o protótipo acima onde "n" indica o número de elementos presentes no vetor.

Exemplo: `Para V={3,3,5,3,2,2,3,3} devolve moda 3.`

Obs: Você pode chamar a função da questão anterior, como função auxiliar.
No caso de empate escolha uma das modas arbitrariamente.

Problema 5:

Critique a seguinte função recursiva para calcular uma moda de um vetor de inteiros, onde "inic" indica o índice do primeiro elemento e "fim" o índice do último elemento.

```
int Moda(int V[], int inic, int fim){
  int meio = (inic+fim)/2;
  int m1,m2,f1,f2,i;

  if(inic==fim) return V[inic];

  m1 = Moda(V, inic, meio);
  m2 = Moda(V, meio+1, fim);

  f1 = 0;
  f2 = 0;
  for(i=inic; i <= fim; i++){
    if( V[i] == m1 ) f1++;
    if( V[i] == m2 ) f2++;
  }

  if(f1 > f2) return m1;
  else        return m2;
}
```

Caso a função esteja errada, apresente um contra-exemplo, isto é, um exemplo que mostre que o algoritmo não é correto.

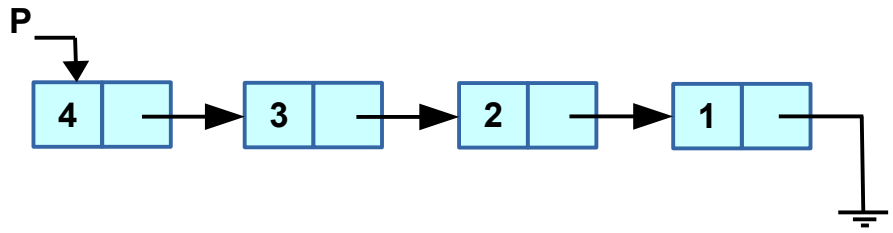
Parte C: Funções recursivas com listas

Problema 1:

Dada a definição abaixo:

```
struct Reg{
  int num;
  struct Reg *prox;
};
```

Faça uma função recursiva que recebe um número inteiro "n", e gera uma lista ligada simples (sem nó-cabeça) contendo os números de 1 a n em ordem decrescente.
Ex: Para n=4, temos:

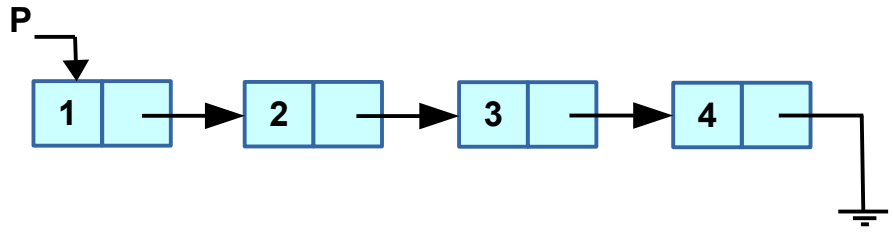



```
struct Reg *GeraListaDecrescente(int n);
```

Use o protótipo acima onde "n" indica o número de elementos.
A função deve devolver o endereço do primeiro nó da lista alocada.

Problema 2:

Faça uma função recursiva que recebe o endereço do primeiro nó de uma lista ligada simples (sem nó-cabeça), e que devolve a lista invertida. Ou seja, a função deve devolver o endereço do último nó, e todos os apontadores da lista devem ser invertidos.
Ex: Usando a lista do exemplo da questão anterior como entrada, teremos que ela será modificada de modo a ficar com a seguinte disposição final:



struct Reg *InverteLista(struct Reg *p);

Use o protótipo acima onde "p" aponta para o primeiro nó da lista de entrada fornecida.
OBS: A função não deve alocar memória adicional, as alterações devem ser feitas na própria lista fornecida.