MAC122 – Princípios de Desenvolvimento de Algoritmos

DCC - IME - USP

2º Semestre de 2014

Prof.: Dr. Paulo Miranda & Marco Aurélio Gerosa

pmiranda@vision.ime.usp.br, gerosa@ime.usp.br

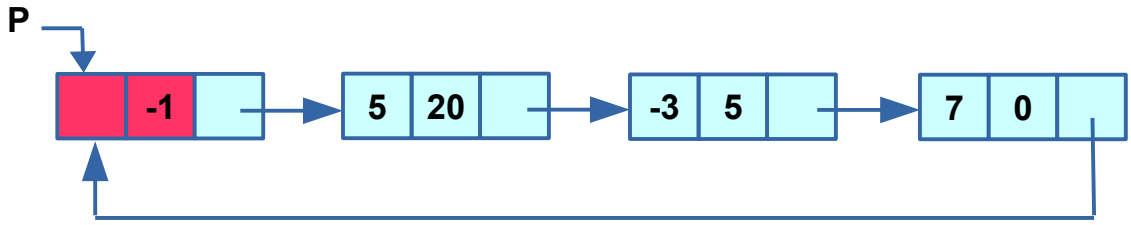
Lista 02:
Estruturas ligadas

Questão 1:

Dado o problema de manipulação de polinômios de grau $n \geq 0$, como por exemplo:
$P(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x^1 + a_0 x^0$
Considere a solução usando listas circulares com nó-cabeça tal que:

a) Cada nó representa um termo com coeficiente não nulo .
b) Os termos estão em ordem decrescente dos expoentes.
c) O nó-cabeça tem expoente -1.

Por exemplo, o polinômio $P(x) = 5 x^{20} - 3 x^5 + 7$ é representado da seguinte forma:



Escreva uma possível definição em C para a estrutura de um termo do polinômio e escreva funções que implementem as operações especificadas abaixo:

A) Sem usar nenhuma função auxiliar, faça uma função que calcula e devolve o valor numérico de um polinômio para um dado valor de x:

```
float Valor(Polinomio p, float x);
```

Exemplo: Se $p(x) = x^3 - 5x + 2$ para x = -1 temos p(-1) = 6.
OBS: Assuma que `Polinomio` é um tipo apontador para um termo da lista, e que `p` aponta para o nó-cabeça da lista.

B) Faça uma função em C para liberar toda a memória de um polinômio.
Use o protótipo abaixo:

```
void LiberaPolinomio(Polinomio p);
```

C) Faça uma função que calcula e devolve a derivada de um polinômio fornecido:
OBS: A função deve criar um segundo polinômio que será devolvido pela função.

```
Polinomio Derivada(Polinomio  p);
```

Exemplo:
Se $p(x) = 3.00\ x^{10} + 2.50\ x^7 + 3.00\ x^1 + 5.00$, temos a derivada:
$p'(x) = 30.00\ x^9 + 17.50\ x^6 + 3.00$

Lembrete: $(a)' = 0$ e $(ax^n)' = na\ x^{n-1}$

D) Faça uma função que calcula e devolve a derivada segunda (derivada da derivada) de um polinômio fornecido:
OBS: A função deve criar um segundo polinômio que será devolvido pela função.

```
Polinomio DerivadaSegunda(Polinomio  p);
```
Exemplos:

Para $p(x) = 5.00\ x^4 + 7.50\ x^2 + 3.00\ x^1$, temos a derivada segunda:
$p''(x) = 60.00\ x^2 + 15.00$

Para $p(x) = 3.00\ x^{10} + 2.50\ x^7 + 3.00\ x^1 + 5.00$, temos a derivada segunda:
$p''(x) = 270.00\ x^8 + 105.00\ x^5$

Para $p(x) = x^3 - x^2$, temos a derivada segunda:
$p''(x) = 6.00\ x^1 - 2.00$

OBS: Faça duas soluções: A primeira que usa as funções dos itens anteriores como funções auxiliares, e uma segunda que calcula a derivada segunda diretamente.

Questão 2:

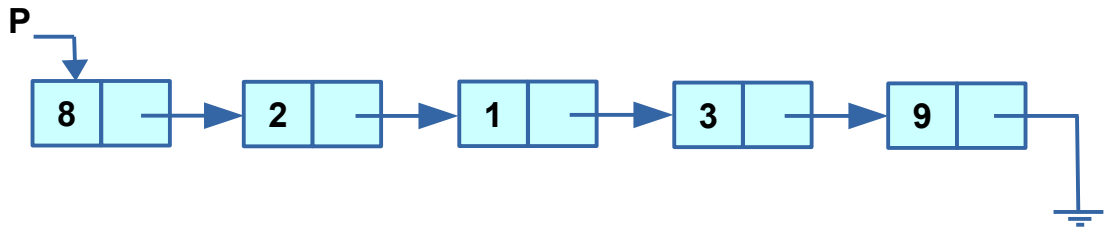
Dada a definição abaixo:

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

A) Faça uma função que recebe o endereço do primeiro nó de uma lista ligada simples (sem nó-cabeça), e que devolve o número de elementos da lista.

```
int NumElementos(struct Reg *p);
```

Ex: Assumindo a lista abaixo como entrada, a função deve devolver tamanho 5.



B) Faça uma função que recebe o endereço do primeiro nó de uma lista ligada simples (sem nó-cabeça), e que verifica se um valor fornecido "x" pertence a lista.

```
int Pertence(struct Reg *p, int x);
```

Use o protótipo acima onde "p" aponta para o primeiro nó da lista de entrada fornecida, e "x" é o valor a ser procurado. A função deve devolver verdadeiro ou falso.

Ex: Assumindo a lista do exemplo anterior como entrada, no caso de x=3 a função deve devolver verdadeiro. Já se x=4 a função deve devolver falso.

C) Faça uma função que recebe duas listas ligadas simples (sem nó-cabeça), e que devolve uma terceira lista (ligada simples) contendo a intersecção das duas primeiras.

```
struct Reg *Interseccao(struct Reg *p, struct Reg *q);
```

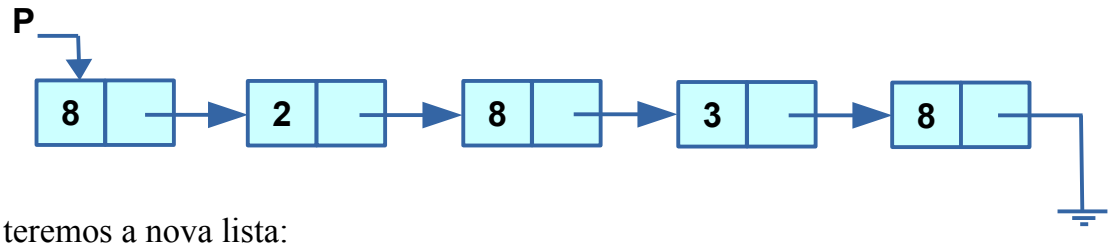
Use o protótipo acima onde "p" aponta para o primeiro nó da primeira lista fornecida, e "q" aponta para o primeiro nó da segunda lista. A função deve devolver o endereço da lista contendo a intersecção. Essa lista deve ser alocada dinamicamente pela função.

Obs: Você pode usar as funções dos itens anteriores como funções auxiliares. Assuma também que não existem elementos repetidos nas listas fornecidas.

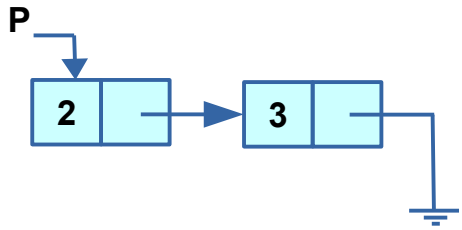
Exemplo: Assumindo "p" como no exemplo do item A, e "q" apontando para uma lista com os números `{7,3,5,1}`, teremos a lista de intersecção contendo apenas os números 3 e 1.

Questão 3:

A) Faça uma função que recebe o endereço do primeiro nó de uma lista ligada simples (sem nó-cabeça), e que elimina todas as ocorrências de um dado valor.
A função deve devolver o endereço do novo início da lista.
Ex: Assumindo a lista abaixo como entrada e valor de remoção 8,



teremos a nova lista:




```
struct Reg *RemoveNum(struct Reg *p, int a);
```

Use o protótipo acima onde "p" aponta para o primeiro nó da lista de entrada fornecida, e "a" é o valor a ser eliminado.
OBS: Você deve liberar a memória dos nós que foram eliminados da lista, e reaproveitar os demais sem a necessidade de alocar memória adicional.

B) Faça uma função que recebe o endereço do primeiro nó de uma lista ligada simples (sem nó-cabeça), e que remove todos os elementos repetidos da lista.
Você pode usar a função do item anterior como função auxiliar.

```
struct Reg *RemoveRepetidos(struct Reg *p);
```

Use o protótipo acima onde "p" aponta para o primeiro nó da lista de entrada fornecida. A função deve devolver o endereço do novo início da lista.

Exemplo: Para a lista de entrada do item anterior, teremos: