MAC122 Princípios de Desenvolvimento de Algoritmos
Lista de exercícios 01

Prof.: Paulo Miranda
Instituto de Matemática e Estatística (IME)
Universidade de São Paulo (USP)

Alocação Dinâmica & Matrizes

Questão 1:

A) Faça uma função em C que recebe uma matriz de inteiros **A**, com **m** linhas e **n** colunas, contendo valores no intevalo de 0 a 255, e que devolve o seu histograma, com as frequências (número de ocorrências) para cada um dos valores do intervalo. O histograma deve ser representado como um vetor de inteiros de 256 posições alocado dinamicamente. Use o protótipo abaixo:

```
int *Histograma(int **A, int m, int n);
```

Exemplo: Para a matriz $\begin{bmatrix} 1 & 2 & 3 & 5 \\ 3 & 5 & 2 & 255 \\ 255 & 2 & 5 & 2 \end{bmatrix}$

O histograma deve ser: `[0, 1, 4, 2, 0, 3, 0, …, 0, 2]`

B) Faça a função abaixo em C que recebe uma matriz de inteiros **A**, com **m** linhas e **n** colunas, alocada dinamicamente (conforme a representação mostrada em aula, onde cada linha é armazenada em um vetor separado), e que altera a própria matriz **A** de modo a inverter suas linhas, virando a matriz na vertical (Flip Vertical).

```
void FlipVertical(int **A, int m, int n);
```

Exemplo:

antes $\begin{bmatrix} 5 & 3 & 7 \\ 3 & 8 & 2 \\ 12 & 6 & 0 \\ 4 & 10 & 9 \end{bmatrix}$ depois $\begin{bmatrix} 4 & 10 & 9 \\ 12 & 6 & 0 \\ 3 & 8 & 2 \\ 5 & 3 & 7 \end{bmatrix}$

C) Usando a mesma representação de matriz do item anterior, agora faça a inversão dos elementos na horizontal (espelhar horizontalmente). Use o protótipo abaixo:

```
void FlipHorizontal(int **A, int m, int n);
```

D) Uma matriz é triangular inferior se ela é quadrada e todos os seus elementos acima da diagonal principal são nulos. Exemplo:

$$\begin{bmatrix} 5 & 0 & 0 & 0 & 0 \\ 3 & 8 & 0 & 0 & 0 \\ 0 & 6 & 4 & 0 & 0 \\ 2 & 10 & 9 & 1 & 0 \\ 8 & 7 & 1 & 3 & 8 \end{bmatrix}$$

Na representação usual existe um desperdício de memória de `(n-1)n/2` elementos nulos, onde `n` é a ordem da matriz. Uma representação linearizada alternativa em um vetor com `n(n+1)/2+1` elementos é mostrada abaixo, onde os valores nulos acima da diagonal são representados uma única vez na primeira posição do vetor:

`[0, 5, 3, 8, 0, 6, 4, 2, 10, 9, 1, 8, 7, 1, 3, 8]`

Escreva uma função que converte os índices `(i,j)` de um elemento da matriz usual para o seu índice correspondente na representação linearizada.

Estruturas ligadas

Questão 2:

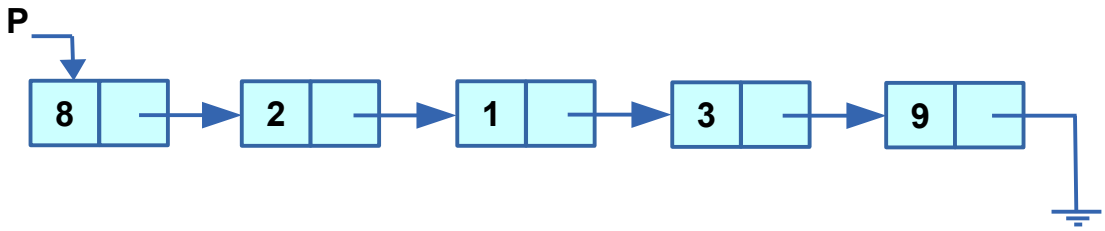
Dada a definição abaixo:

```
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);
```

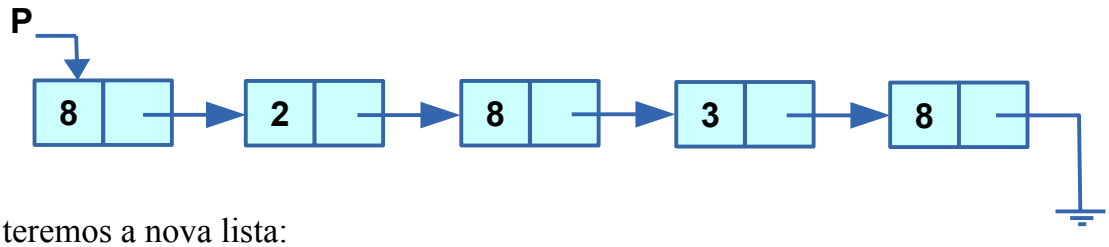
Use o protótipo acima onde "p" aponta para o primeiro nó da primeira lista fornecida e "q" aponta para o primeiro nó da segunda lista. A função deve devolver o endereço da lista contendo a intersecção. Essa lista deve ser alocada dinamicamente pela função.

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

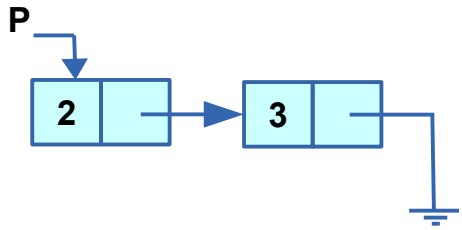
Exemplo: Assumindo "p" como no exemplo do item A, e "q" apontando para uma lista com os números `{7,3,5,1}`, teremos a lista de intersecção contendo apenas os números 3 e 1.

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);
```

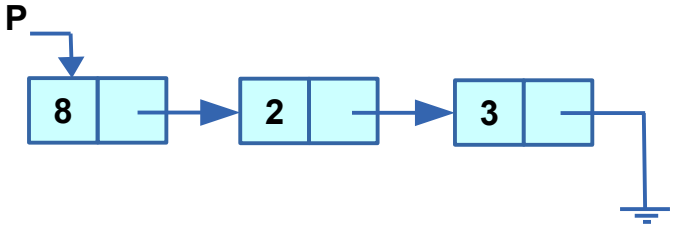
Use o protótipo acima onde "p" aponta para o primeiro nó da lista de entrada fornecida e "a" é o valor a ser eliminado.
OBS: Você deve liberar a memória dos nós que foram eliminados da lista e reaproveitar os demais sem a necessidade de alocar memória adicional.

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:



Questão 4:

Dadas as definições abaixo:

```
struct node {
  int item;
  struct node *prox;
};

typedef struct node* lista;
```

Explique o funcionamento da função XXXX abaixo, dado que ela recebe o endereço do primeiro nó de uma lista encadeada simples cujo último nó aponta para NULL:
OBS: Faça desenhos ilustrando exemplos do progresso do algoritmo a cada iteração do laço.

```
lista XXXX(lista x) {
    lista t, y, r;
    y = x;
    r = NULL;
    while (y != NULL) {
       t = y->prox;
       y->prox = r;
       r = y;
       y = t;
    }
    return r;
}
```

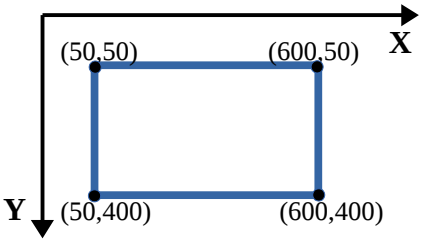
Exemplo de chamada dessa função:

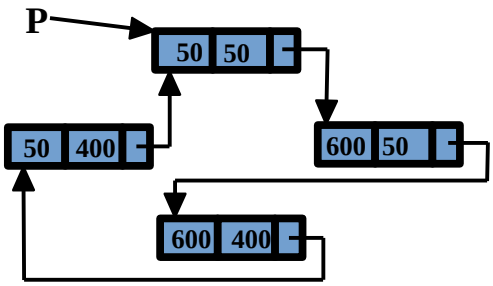
```
int main(){
   lista p = NULL;
   lista t;
   int i;

   /* adiciona alguns elementos na lista p */
   for(i = 3; i >= 0; i--){
      t = (lista)malloc(sizeof(struct node));
      t->item = i;
      t->prox = p;
      p = t;
   }

   p = XXXX(p);

   ImprimeLista(p);

   return 0;
}
```

Estruturas ligadas, Matrizes e Arquivos

Questão 5:

Um polígono pode ser representado pela sequência de seus vértices, armazenados em uma lista circular em sentido horário. Exemplo:





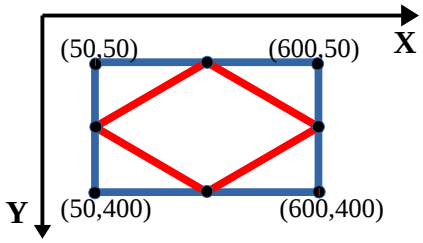
Considere a seguinte estrutura:

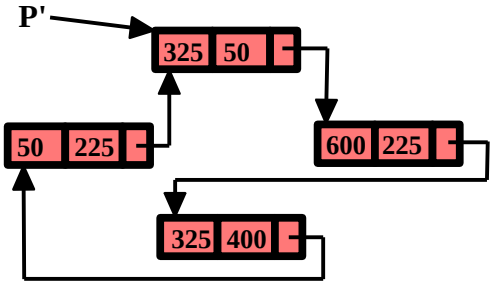
```
struct Vertice{
        float x;
        float y;
        struct Vertice *prox;
};
typedef struct Vertice* Poligono;
```

Você deve implementar as seguintes funções:

int NumeroDeVertices(Poligono P);
Função que conta o número de vértices de um polígono fornecido. Por exemplo, no caso da lista circular do exemplo acima, a função deve devolver 4.

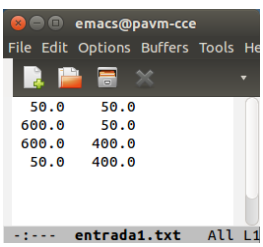
Poligono PoligonoPorPontoMedio(Poligono P);
Função que devolve um segundo polígono derivado de P, de tal modo que seus vértices são obtidos pelos pontos médios dos lados do primeiro polígono.
Exemplo:

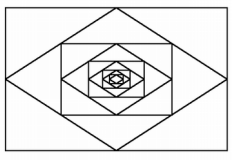


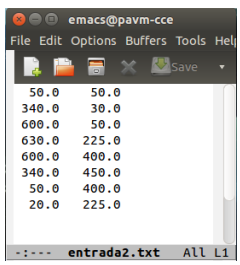


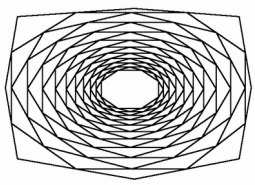
OBS: O polígono gerado deve ser armazenado em uma segunda lista circular que deve ser alocada dinamicamente. A função pode devolver o endereço de qualquer um dos vértices do polígono gerado (não existe uma ordem preferencial). A função deve devolver NULL se o polígono P for inválido, isto é, com menos que três vértices.

Faça um programa que lê um polígono de um arquivo texto fornecido, contendo as coordenadas de seus vértices (um vértice por linha), e que gera uma imagem de 640x480 contendo o desenho de uma sequência de polígonos por ponto médio, até um número máximo de repetições R. O nome do arquivo de saída deve ser "poligono.pgm".









(a) (b) (c) (d)

Figura 1: (a,c) Arquivos de entrada. (b,d) Imagens dos polígonos por ponto médio de a e c (com R=10 e 20 respectivamente).

Dica: utilize a função auxiliar abaixo que desenha um segmento de reta $\overline{AB}$ na imagem com o valor **`val`** fornecido. A imagem é dada pela matriz M com m linhas e n colunas, A = (x1,y1) e B = (x2,y2).

```
void DesenhaReta(int **M, int m, int n, int x1, int y1, int x2, int y2, int val){
  float x,y,dx,dy;
  int xi,yi;

  dx = (x2-x1);
  dy = (y2-y1);
  if(fabsf(dx) > fabsf(dy)){
    dy = dy/(fabsf(dx));
    dx = dx/(fabsf(dx));
  }
  else{
    dx = dx/(fabsf(dy));
    dy = dy/(fabsf(dy));
  }
  x = (float)x1;
  y = (float)y1;
  xi = x1;
  yi = y1;
  while(xi != x2 || yi != y2){
    if(yi >= 0 && yi < m && xi >= 0 && xi < n)
      M[yi][xi] = val;
    x += dx;
    y += dy;
    xi = (int)(x + 0.5);
    yi = (int)(y + 0.5);
  }
  if(y2 >= 0 && y2 < m && x2 >= 0 && x2 < n)
    M[y2][x2] = val;
}
```