Universidade Tecnológica Federal do Paraná (UTFPR)
Departamento Acadêmico de Informática (DAINF)
Estrutura de Dados I
Professor: Rodrigo Minetto
Exercícios (lista encadeada)

Exercícios (seleção): necessário entregar TODOS (moodle)!

Exercício 1) Projete uma estrutura de dados **fila** através de uma lista encadeada. Precisamente, codifique as seguintes funções em **queue.c** (dentro de **arquivos.zip**):

```
/*Structure*/
typedef struct node {
  int data;
  struct node *next;
} Queue;

/*Interface:*/
Queue* create ();
void destroy (Queue *q);
Queue* enqueue (Queue *q, int elem);
Queue* dequeue (Queue *q);
int get_front (Queue *q);
int empty (Queue *q);
void print (Queue *q);
```

Para testar sua implementação de **fila** utilize o programa **test-queue.c** que tem uma implementação já feita (não precisa codificar nada) para o exercício jackpot (visto na aula de estrutura de dados **fila**). Se facilitar, você pode utilizar o arquivo **test-queue-codeblocks.c** (que tem o .h e .c de **fila** em um único arquivo).

Exercício 2) Projete uma estrutura de dados **pilha** através de uma lista encadeada. Precisamente, codifique as seguintes funções em **stack.c** (dentro de **arquivos.zip**):

```
/*Structure*/
typedef struct node {
  int data;
  struct node *next;
} Stack;

/*Interface:*/
Stack* create ();
void destroy (Stack *s);
Stack* push (Stack *s, int elem);
Stack* pop (Stack *s);
int get_peek (Stack *s);
```

```
int empty (Stack *s);
void print (Stack *s);
```

Para testar sua implementação de **pilha** utilize o programa **test-stack.c** que tem uma implementação já feita (não precisa codificar nada) para o exercício bem-formada (visto na aula de estrutura de dados **pilha**). Se facilitar, você pode utilizar o arquivo **test-stack-codeblocks.c** (que tem o .h e .c de **pilha** em um único arquivo).

Exercício 3) Escreva uma função que determina o elemento do meio de uma lista encadeada, tal que, você não pode percorrer a lista encadeada mais de uma vez para determinar este elemento. Não altere a função main, ou seja, não inclua qualquer contador durante a inserção dos elementos, apenas percorra a lista na função **int meio (List *l)** e retorne o elemento que se situa na metade da lista. Para tanto, utilize o programa em **meio-lista.c** (dentro de **arquivos.zip**), ou alternativamente **meio-lista-codeblocks.c**

A saída esperada para o programa é:

```
List: 2 1 0
Elemento no meio da lista: 1
List: 9 8 7 6 5 4 3 2 1 0
Elemento no meio da lista: 4
List: 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
Elemento no meio da lista: 9
```

Se o número de elementos na lista é par retorne qualquer um dos elementos centrais.