Universidade Estadual de Maringá (UEM)
Departamento de Informática (DIN)

Curso:			
Disciplina:	Fundamentos de Algoritmos		
Professor:	Lucas de Oliveira Teixeira	**Data:**	_____
Aluno:	_____	**R.A.:**	_____

Terceiro trabalho - Lista encadeada

- **Objetivo:** Estudar ponteiros, alocação dinâmica e tipos abstratos de dados.
- **Descrição:** Uma lista encadeada (ou lista ligada) é uma estrutura de dados dinâmica composta por uma sequência de nós (ou células) em que cada nó da lista contém duas informações: um dado e uma referência (ponteiro) para o próximo item da lista. Assim, cada nó pode ser definido como:

```
1  struct no {
2    int dado;
3    struct no *proximo;
4  };
5
6  typedef struct no *No;
```

O comando **typedef struct no *No;** define o tipo **No** como sendo um ponteiro para **struct no**, isso reduz a quantidade de vezes que precisamos usar o operador de desrefereciamento * na declaração, acesso e nos parâmetros das funções.
Além disso, como cada nó possui um ponteiro para o próximo nó, só precisamos da referência para o primeiro nó da lista para conseguir acessar toda a lista:

```
1  int main() {
2    No raiz = NULL;
3  }
```

Por exemplo, a seguinte função recebe o ponteiro para a raiz da lista e imprimi todos os dados da lista, até o ponteiro para o próximo ser nulo.

```
1  void imprimi_lista_encadeada(No raiz) {
2    No p;
3    for (p = raiz; p != NULL; p = p->proximo) {
4      printf ("%d\n", p->dado);
5    }
6  }
```

O trabalho consiste de implementar uma série de funções para a manipulação da lista encadeada.
A primeira função necessária é a função de criação de um nó, essa função deve receber o dado como parâmetro e retornar uma nova instância para a estrutura nó dinamicamente alocada:

```
1  No cria_no(int dado) {
2    ...
3  }
```

A segunda função é a função de inserção de um novo nó na lista, essa função precisa usar a função anterior para instanciar um novo nó e em seguida deve inserir o novo nó no começo da lista, tratando as ligações entre os ponteiros de forma adequada para ligar o novo nó na lista:

```
1  struct no* insere_no(struct no *raiz, int dado) {
2    No novo_no = cria_no(dado);
3    ...
4  }
```

Finalmente, a última função é a de remoção de nós da lista, essa função deve remover o primeiro da lista e de fato liberar esse espaço da memória, tratando as ligações entre os ponteiuros de forma adequada para desligar o nó da lista:

```
1  void remove_inicio(No *raiz) {
2      ...
3  }
```

Por exemplo, o trecho de código abaixo insere três nós na lista e a imprimi; em seguida, remove os dois primeiros nós e imprime a lista novamente:

```
1  int  main() {
2    No raiz = NULL;
3
4    imprimi_lista_encadeada(raiz);
5
6     insere_inicio(&raiz, 10);
7     insere_inicio(&raiz, 20);
8     insere_inicio(&raiz, 30);
9
10   imprimi_lista_encadeada(raiz);
11
12   remove_inicio(&raiz);
13   remove_inicio(&raiz);
14
15    printf ("\n\n");
16   imprimi_lista_encadeada(raiz);
17 }
```

- O código fonte deve ser entregue via Moodle.