Universidade Federal de Goiás
Goiânia 29 de março de 2018
Edson Júnior Frota Silva – 201515412
Engenharia de Computação

Atividade de Estrutura de Dados II

(1) Escreva uma versão da função busca para listas sem cabeça.

```
#include<stdio.h>

    busca(int x, celula *inicio)
    {
        celula*p;
        p = inicio;
        while(p != NULL && p->conteudo !x)
        {
            p = p->prox;
            return(p);
        }
    }
```

(2)Por que a seguinte versão de insere não funciona?

```
void insere (int x, celula *p)
{
    celula nova;
    nova.conteudo = x;
    nova.prox = p->prox;
    p->prox = &nova;
}
```

Por causa que está escrito como nova.prox, sendo que a variável prox é um ponteiro, então deveria ser expresso como nova->prox. p->prox recebe o endereço da nova variavel criada, não de seu conteúdo, então deveria estar escrito como p->prox = nova.

(3) Escreva uma função que insira um novo elemento em uma lista encadeada sem cabeça. (Será preciso tomar algumas decisões de Projeto antes de começar a programar.

```
#include<stdio.h>

    void insere (int x, celula *p)
```

```
    {
        celula *nova, *q = ini;
        nova = malloc (sizeof (celula));
        nova->conteudo = x;
        if (p == ini)
        {
            ini = nova;
            ini->prox = p;
       }
        else
        {
            while (q->prox != p)
                q = q->prox;
            nova->prox = q->prox;
            q->prox = nova;
        }
    }
```

(4) Invente um jeito de remover uma célula de uma lista encadeada sem cabeça. (Será preciso tomar algumas decisões de projeto antes de começar a programar.)

```
#include<stdio.h>

void remover(pessoa *p){
   pessoa *remove;
   remove = p->prox;
   p->prox = remove->prox;
   remover(lixo);
}
```

(6) Escreva uma função que faça uma cópia de uma lista dada como parâmetro e retorne a nova lista.

```
#include<stdio.h>

pessoa copiar(pessoa *lista1, pessoa *lista2)
{
    pessoa *p, *q;
    q = lista2;
    for(p = lista1; p->prox != NULL; p = p->prox)
    {
        q->num = p->num;
        q = q->prox;
    }
```

```
}
```

(7) Escreva uma função que concatena duas listas encadeadas (isto é, "amarra" a segunda no fim da primeira).

```
#include<stdio.h>

   void imprima (celula *ini)
{
   celula *p;
   for(p = ini->prox; p != NULL; p = p->prox)
      printf("%d\n", p->conteudo);
}
```

(8) Escreva uma função que conta o número de células de uma lista encadeada.

```
#include<stdio.h>

int contar(pessoa *p)
{
    pessoa *run;
    int n;
    while(run->prox != NULL)
    {
        run = run->prox;
        n++;
    }
    return n;
}
```

(9) Escreva uma função que verifica se duas listas dadas são iguais, ou melhor, se têm o mesmo conteúdo.

```
#include<stdio.h>

int compara(pessoa *p1, pessoa *p2)
{
    int n = 0;
    pessoa *p,*q;
    q = p2;
    for(p = p1; p->prox != NULL; p = p->prox)
    {
        if(q->num != p->num)
            n++;
        q = q->prox;
```

```
    }
    return n;
}
```

(10) Escreva uma função que inverte a ordem das células de uma lista encadeada (a primeira célula passa a ser a última, a segunda passa a ser a penúltima etc.). Faça isso sem usar espaço auxiliar; apenas altere os ponteiros.

```
#include<stdio.h>

void inverte(pessoa *p)
{
    pessoa *aux1, *aux2;
    aux1 = p;
    aux2 = p->prox;
    while(aux2->prox != NULL)
    {
        aux1->prox = aux1;
        aux1 = aux2;
        aux2 = aux2->prox;
    }
    p = aux2;
}
```