2023 0001178 Denio

Universidade Federal da Fronteira Sul - UFFS
Campus Chapecó
Ciência da Computação
Estrutura de Dados

Instruções

1. Coloque apenas a sua MATRÍCULA na folha resposta.
2. Aparelhos eletrônicos desligados (ou no modo silencioso).
3. Responda TODAS questões na folha resposta, enumerando as mesmas na ordem que melhor lhe convier.
4. As questões podem ser resolvidas a lápis porém o professor se reserva a não aceitar reclamações oriundas da correção das questões.
5. Consulta permitida apenas à cola oficial.
6. Coloque, na folha resposta, o nome do professor da sua turma.

Avaliação 02

1. (4 pontos) Imagine duas listas encadeadas simples, estas duas listas tem um ponteiro apontando para o primeiro elemento das mesmas, com a seguinte estrutura:

```
struct tlist {
   int valor;
   struct tlist *next;
};
typedef struct tlist list;
```

Implemente a função `list *mergeSub(list *h1, list *h2)` que recebe dois ponteiros de duas listas do tipo *list* e retorna o endereço de uma NOVA lista formada pela subtração dos elementos nas mesmas posições. Caso a posição seja NULL, mantenha o valor original. Implemente apenas a função, sem a necessidade da `main`. A função `main` teria o seguinte trecho:

```
:
list *sub;
sub=mergeSub(p1, p2);
```

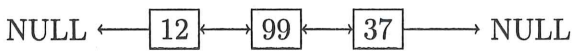
p1 e *p2* são os ponteiros das listas utilizadas para realizar a subtração e *sub* apontará para o primeiro elemento da nova lista.

Exemplo: Caso a lista apontada por *p1* contenha os elementos 10, 12, 3, 14, 5 e a lista apontada por *p2* os elementos 2, 4, 6, 8, o retorno da função será um ponteiro para uma lista contendo os elementos 8 (10-2), 8 (12-4), -3 (3-6), 6 (14-8) e 5 (5 e NULL). Cuidado ao percorrer as listas pois o número de elementos é variado, ou seja, em uma configuração a lista apontada por *p1* pode ter mais elementos que a apontando por *p2* ou ao contrário. Assim, o uso do **while** permite uma lógica menos complexa.

2. (3 pontos) Dada estrutura abaixo:

```
struct tdl {
   int n;
   struct tdl *prev, *next;
};
typedef struct tdl dl;
```

Implemente a função `void ShowNeighbors(dl *list, int key)` que imprima o elemento **anterior** e o elemento **posterior** a posição de memória onde o valor de `key` esteja armazenado. A lista apontada por `list` não possui valores repetidos. Cuidado: atente para os casos em que `key` esteja na primeira ou na última posição (ou não exista). Exemplo, dada a lista abaixo:



Suponha que `f` aponte para o primeiro elemento dessa lista.
A chamada `ShowNeighbors(f,99)` imprimirá 12 e 37.
A chamada `ShowNeighbors(f,12)` imprimirá 99.
A chamada `ShowNeighbors(f,1)` imprimirá `Valor 1 não encontrado.`

3. (3 pontos) A função abaixo implementa a exclusão de um valor passado como parâmetro, caso este valor seja o último da lista. Se não for, mantém a lista sem alteração. Preencha as lacunas para completar este código (0.375 cada acerto).

```
#include <stdio.h>
#include <stdlib.h>
struct tdlist {
  int value;
  struct tdlist *prev;
  struct tdlist *next;
};
typedef (a)_____ dlist;

struct tsent {
    dlist *head, *tail;
};
typedef struct tsent sentinela;

void *deleteIfLast(sentinela *s, dlist *f, int v)
{
   dlist *aux;
   if (s->(b)_____ == NULL)
   {
       printf((c)"_____");
       return;
   }
   if ((d)_____!=v) {
      print("%d não é o último valor\n",v)
      return;
   }
   aux=(e)_____;
   s->tail=(f)_____;
   s->tail->(g)_____=NULL;
   free((h)_____);
   return;
}
```

BOA PROVA & BOA SORTE.