Segunda Avaliação de Estrutura de Dados I

Considere os seguintes tipos para a definição de listas encadeadas em todas as questões abaixo. Responda a todas as questões em linguagem C.

```
typedef struct tipoNo {
int dado;
tipoNo *prox;
} tipoNo;
```

```
typedef struct tipoLista {
tipoNo *prim;
} tipoLista;
```

1 – Faça uma função que receba como parâmetro uma lista encadeada, definida com o tipoLista dado acima, e retorne a quantidade de elementos presentes na lista encadeada passada como parâmetro. A função deve retornar zero para listas vazias. (2,5)

2 – Faça uma função que retorne 1 (verdadeiro) caso todos os valores presentes em uma lista encadeada passada como parâmetro sejam maiores que todos os valores presentes em uma segunda lista encadeada também passada como parâmetro (cada valor na primeira lista maior que todos os valores na segunda lista). Em caso contrário, a função deve retornar um valor correspondente a falso (0). A função deve retornar verdadeiro caso qualquer uma das listas seja vazia. (2,5)

3- Faca uma função que retorne verdadeiro caso duas listas encadeadas passadas como parâmetro sejam idênticas, contendo os mesmos valores ocorrendo na mesma ordem. A função deve retornar falso em caso contrário. Se as duas listas forem vazias, a função deve retornar verdadeiro. (2,5)

4 – Corrija a função de inserção ordenada em uma lista encadeada descrita no código abaixo. A função deve inserir em ordem crescente um elemento passado como parâmetro em uma lista encadeada também passada como parâmetro (buscar a posição correta para inserir o elemento de maneira que a lista, que veio ordenada, continue ordenada). Apresente o código corrigido e explique cada uma das alterações realizadas de maneira a ajudar o programador que cometeu os erros. Você deve realizar o mínimo possível de alterações e apenas as mudanças com devidas explicações serão aceitas. A resposta será considerada incorreta se a função corrigida apresentar qualquer erro. (2,5)

```
void inserirOrdenado(tipoLista L, int *d) {
  tipoNo *aux, *aux2 = NULL, *aux3 = L.prim;

  while((aux3) && (chave < aux3->dado)) { /* ideia aqui é aux2 apontar para o anterior à */
    aux3 = aux3->prox;                    /* posição de inserção  e aux3 para o seguinte */
    aux2 = aux3;
  }  /* fim do while */
  aux->dado = *d;
  if(aux2 != NULL) {
    aux->prox = aux3;
    aux2->prox = L.prim;
  } else {
    L.prim->prox = aux;
    L.prim = aux;
  } /* fim do if */
} /* fim da função */
```