Ministério da Educação
Universidade Tecnológica Federal do Paraná
Câmpus Pato Branco

2ªAvaliação – AE22CP – 2017/1

Instruções para a avaliação:

- Leia a prova com atenção e coloque o seu nome completo na mesma.
- A avaliação é individual.
- A interpretação da prova faz parte da avaliação.
- É PROIBIDA a consulta à Internet.
- O peso de cada exercício está indicado ao final do enunciado.
- Cada exercício deve ser resolvido em um arquivo .c separado.
- As soluções não devem conter absolutamente NENHUM "scanf", "getch", ou esperar qualquer tipo de entrada do usuário. A "main" deve ser capaz de já testar as funções, provando sua eficácia.
- Cada solução será avaliada em termos de funcionalidade (leva em consideração todos os casos possíveis?), legibilidade (o código está compreensível?), uso adequado de tipos, estruturas, etc. Dica: variáveis globais e "breaks" não são legais!
- Ao final da avaliação, os arquivo devem ser enviados pelo MOODLE.

Nome : _____

1. Zeynar sonha que um dia será o melhor jogador de futebol do mundo e, considerando esta utopia, resolveu começar a armanezar informações de todos os seus gols. Você o ajudará criando uma lista estática com as seguintes informações sobre cada gol: tipo de gol (cabeça, chute ou pênalti) e tempo (1º ou 2º tempo). **(Peso 3,0)**

a) Defina tipos e funções necessárias para armazenar no mínimo 500 gols.

b) Crie as seguintes funções "estatísticas":

1) "GolsPorTipo": imprime quantos foram os gols de cabeça, quantos por chute e quantos de pênalti;

2) "GolsPorTempo": quantos gols foram marcados no 1º tempo e quantos no 2º;

3) "Total": imprime o total de gols marcados.

2. Uma frase pode ser representada por uma lista simplesmente encadeada **L1**, sendo que o único campo de cada nodo desta lista é um caracter. Implemente uma função "CortaLista" que recebe como parâmetro dois índices de caracteres, **i1** e **i2**, e retorne o ponteiro para uma nova lista **L2**, representando a frase que inicia com o caractere i**1** e termina com o caractere **i2**. **(Peso: 3,0)**

Obs. 1: a lista original não deve ser alterada. Após a chamada da função "CortaLista", imprima o conteúdo de **L1** e **L2** para garantir que ambas estão de acordo.

Obs. 2: lembre-se de verificar se existem os índices **i1** e **i2** em **L1.**

Obs. 3: a lista **L2** não deve conter cópias dos nodos de **L1**, e sim reaproveitá-los!

3. Em geral, linguagens de programação possuem um tipo de dados inteiro (**`int`**) que suporta operações básicas como: adição, subtração, multiplicação e divisão. Na prática, no entanto, essas operações são limitadas a valores inteiros menores que um certo limite (`INT_MAX`). Deste modo, para aplicações específicas, em que valores maiores que `INT_MAX` precisem ser manipulados, o tipo **`int`** se torna inadequado.

Uma forma alternativa de se representar inteiros maiores que `INT_MAX`, seria utilizar uma estrutura de dados lista, na qual cada item da lista armazene um único dígito (`0-9`) do inteiro em questão.

a) Utilize listas duplamente encadeadas dinâmicas para implementar a adição de inteiros grandes positivos! **(Peso: 3,0)**

Especificações:

```
Lista* add(Lista* A, Lista* B); // Implementar
```

Entrada:

Considere que os inteiros grandes (positivos) estão incialmente em uma string cada. Exemplo:

```
const char* x ="92233720368547758079223372036854775807";

const char* y ="2147483";
```

Cada string deve ser, então, transformada em lista por uma função a ser implementada.

```
Lista* tolist(const char* string); // Implementar
```

***Obs**.:* Para converter um caractere numérico c='1' para um inteiro a=1, utilize o seguinte código.

```
int a = (int) (c - '0');
```

Objetivo:

Dadas duas listas dinâmicas duplamente encadeadas `A` e `B`, representando números inteiros grandes, adicioná-las e imprimir a lista resultante.

```
Lista* A = tolist(x);

Lista* B = tolist(y);

printList( add(A, B) );
```

b) Indique uma motivação para se utilizar listas dinâmicas, ao invés de estáticas, na resolução do problema acima? **(Peso: 1,0)**