ACH2023 ALGORITMOS E ESTRUTURAS DE DADOS I

Semestre 2019-2 - Exercício prático 1 –Decodificação de strings – t.02

Estagiário PAE: Samuel Caetano da Silva (samuel.caetano.silva@usp.br)

1. O objetivo do trabalho é implementar de forma correta e completa a função *decodificar* utilizando apenas listas ligadas de implementação dinâmica como estrutura de armazenamento, conforme modelo fornecido. A assinatura da função é a seguinte:

   NO* decodificar(char* frase)

2. A função recebe como entrada um vetor de caracteres formando um *string* não-vazio representando uma frase codificada, e retorna a mesma frase na forma de uma lista ligada de implementação dinâmica na qual cada nó contém um caractere. A codificação existente no string de entrada respeita duas regras:

   a. Letras e caracteres especiais são representadas normalmente, ou seja, um símbolo "a" resulta em um nó "a" da lista ligada, um símbolo "," resulta em um nó contendo uma vírgula, e assim por diante.

   b. Os símbolos do tipo dígito são especiais. Ao invés de reproduzir o dígito na lista de saída, um dígito de 0..9 deve ser interpretado como a quantidade de vezes que o próximo símbolo da sequência deve aparecer repetido. Assim, um 5 seguindo de um "n" no string resulta em uma sequência de cinco nós contendo, cada um deles, uma letra "n". O dígito 1 cria um único nó exibindo o símbolo seguinte (que pode ele próprio ser um dígito), e o dígito zero indica que o símbolo seguinte não deve aparecer na lista de saída, ou seja, sequências de 0 seguidas de um símbolo são ignoradas.

3. Exemplo (dígitos indicando repetição aparecem em negrito):

   Entrada (string): Li **1**2 vezes mas não entendi n**4**ada**5**! Vou tirar **0**0**1**0.
   Saída (lista ligada): Li 2 vezes mas não entendi naaaada!!!!! Vou tirar 0.

4. A quantidade de nós a serem criados é sempre representada por um dígito entre 0 e 9. Havendo um dígito na última posição do string de entrada, ele deve ser tratado como um caractere normal.

5. Restrições de implementação:

   - Não use nenhum vetor na sua implementação. Se necessitar de estruturas auxiliares, use sempre listas ligadas de implementação dinâmica.
   - Não defina variáveis globais.
   - Não exiba nenhuma mensagem na tela, nem solicite que o usuário pressione nenhuma tecla etc.

6. A função implementada deve estar inteiramente contida em um arquivo de nome **trabalho.cpp**. Note no entanto que para testar a sua implementação e garantir sua correção você provavelmente terá de criar várias outras funções auxiliares (e.g., entrada de dados, exibição etc.) que não serão avaliadas.

7. O EP pode ser desenvolvido individualmente ou em duplas, desde que estas sejam cadastradas até **8 de setembro** no link a seguir. Alunos que queiram trabalhar individualmente também devem se cadastrar, e duplas formadas tardiamente não serão consideradas. Por favor acesse o link abaixo usando seu email USP:

   https://docs.google.com/spreadsheets/d/1wZHlZHKaMf1hthc4DfbidXGO9FxkdhfHzn9rmvLZFeI/edit?usp=sharing

   Importante: anote o número do seu grupo para informá-lo no código a ser entregue.

O que/como entregar:

- 0 programa deve ser compilável no Codeblocks 13.12 sob Windows 7 ou superior. Será aplicado um **desconto de 50%** na nota do EP caso ele não seja **prontamente** compilável nesta configuração.
- Entregue um arquivo trabalho.cpp (exatamente com este nome, sem compactação) contendo a função do EP e todas as rotinas que ela invoca.
- A entrega será via *upload* no sistema Tidia por **apenas um** integrante de cada dupla.
- Preencha no código as declarações *nroUSP1*, *nroUSP2* e *grupo* para que você seja identificado. Se o EP for individual, mantenha o valor do segundo nro. como zeros.

Prazos:
O EP deve ser depositado no prazo definido na atividade cadastrada no sistema Tidia. Não serão aceitos EPs entregues depois do prazo, independentemente do motivo. Entregas no último dia são assim por conta e risco do aluno, e nenhum tipo de imprevisto de última hora (e.g., problemas de saúde, indisponibilidade de rede etc.) pode ser usado como justificativa para o atraso. O EP é uma atividade para ser desenvolvida ao longo de várias semanas, não no último dia da entrega.

O sistema Tidia envia um email de confirmação da submissão. É responsabilidade do aluno que fez o *upload* do arquivo verificar se o mesmo foi corretamente recebido pelo sistema (ou seja, sugere-se fazer *download* novamente para ter certeza). Atrasos/falhas na submissão invalidam o trabalho realizado. Certifique-se também de que a versão entregue é a correta *antes* do prazo final. Não serão aceitas substituições.

Avaliação:
O objetivo do exercício é o de realizar codificações **corretas**, que não necessariamente precisam ser eficientes ou "elegantes". Ou seja, basta que o programa forneça a resposta correta para cada string de teste fornecido. Não existe assim solução "meio" correta: ou a função faz o que foi proposto, ou não faz. Erros de execução e de alocação de memória (muito comuns!) também invalidam o teste, assim como a ausência de eventuais funções auxiliares necessárias para a execução do programa.

O EP será testado com uma série de 10 chamadas fornecendo diversos *strings* não vazios como entrada. Cada teste sem erro de execução que retornar a resposta esperada vale um ponto. Para qualquer resultado diferente do esperado, passa-se ao próximo teste sem pontuar.

Este EP deve ser desenvolvido obrigatoriamente por *todos* os alunos de AED1. Sua nota é parte integrante da média final da disciplina e *não é* passível de substituição. Problemas com EPs – principalmente erros de execução e plágio – são a principal causa de reprovação em ACH2023. Não tente emprestar sua implementação para outros colegas, nem copiar deles, pois isso invalida o trabalho de todos os envolvidos.

Dicas:

Sugere-se tratar todos os *strings* utilizando apenas funções padrão para o tipo char* em C:

```
int x = strlen(frase);          // retorna o tamanho da frase

strcpy(destino, origem);        // copia o conteúdo do string origem em destino
strcpy(texto, "hello");

strcat(cabeca, cauda);          // concatena cauda à cabeça; cauda não é alterada

x = strcmp(s1, s2);             // retorna 0 se s1 e s2 são iguais, ou outro nro. se diferentes .
```

Não está funcionando? Use comandos *printf* para saber por onde seu programa está passando e os valores atuais das variáveis, ou os recursos de *debug* do *CodeBlocs*. O propósito do exercício é justamente o de encontrar erros por conta própria – não espere ajuda para isso. Bom trabalho!