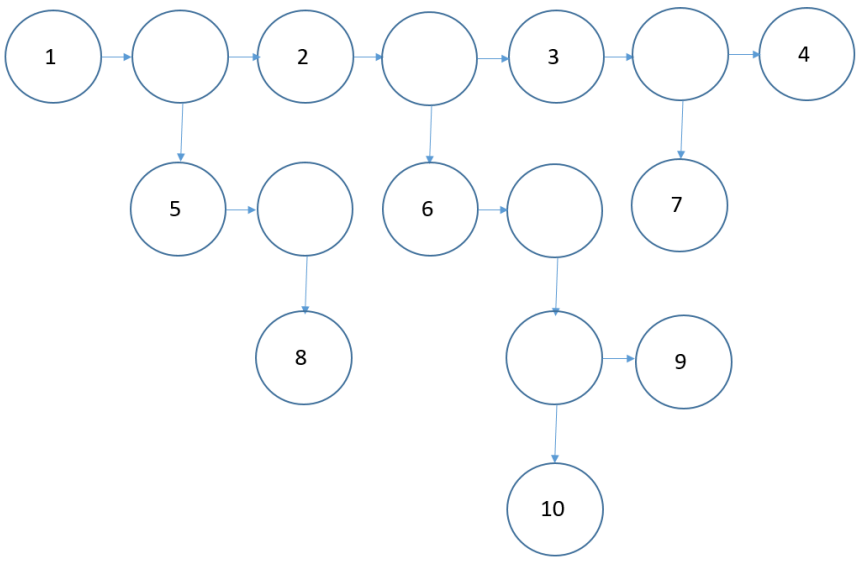
ACH2023 ALGORITMOS E ESTRUTURAS DE DADOS I

Semestre 2018-2 - Exercício prático 2 – Lista Generalizada – t.02

1. O projeto exemplo fornecido contém um arquivo **main.cpp** que pode ser usada para execução e teste do seu programa (já contendo os tipos de dados manipulados) e o arquivo **trabalho.cpp** onde deve ser implementada a tarefa propriamente dita.
2. O objetivo do trabalho é implementar de forma correta e completa a função *listarChaves* utilizando apenas listas ligadas generalizadas como estrutura de armazenamento. A assinatura da função é a seguinte:

   NO* listarChaves(NO* entrada)

3. A função recebe como entrada uma lista ligada generalizada de inteiros não-vazia, e retorna uma nova lista contendo todas as chaves de entrada **em ordem de profundidade** e do primeiro para o último elemento de cada lista individual. Considere para este fim que o primeiro nó possui profundidade 1, seus subordinados possuem profundidade 2 e assim por diante.
4. Por exemplo, para a estrutura de entrada ilustrada a seguir, com início em 1, a função deve retornar uma nova lista simples contendo as chaves 1,2,3,4,5,6,7,8,9,10 exatamente nesta ordem.



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 no módulo **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 pelas mesmas duplas do EP1**. Novas duplas não são aceitas.

O que/como entregar:

- Não remova a declaração include “ep.h” que estão no começo do código, e não duplique as declarações typedef que já existem no módulo ep.h.
- Remova a função main() do seu código e demais funções de teste etc. que não fazem parte da entrega.
- declarações *nroUSP1*, *nroUSP2* e *grupo* para que você seja identificado. Se o EP for individual, mantenha o valor do segundo nro. como zeros.
- 0 programa deve ser compilável no Code::blocks 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.
- Renomeie seu módulo trabalho.cpp criando um arquivo cujo nome é o número do seu grupo com dois dígitos e substituindo a extensão .cpp por .txt (por exemplo, trabalho.cpp se torna 02.txt para a dupla 2 da turma) sem compactação. Este arquivo deve conter 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.

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 obter listas **corretas**, e o código não necessariamente precisa ser eficiente ou elegante. Ou seja, basta que o programa funcione de tal forma que a lista resultante apresente as chaves na ordem estipulada. 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 5 chamadas fornecendo diferentes listas generalizadas não vazias como entrada. Cada teste sem erro de execução que retornar a resposta esperada vale dois pontos. 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.

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 *Code::Blocks*. O propósito do exercício é justamente o de encontrar erros por conta própria – não espere ajuda para isso. Bom trabalho!