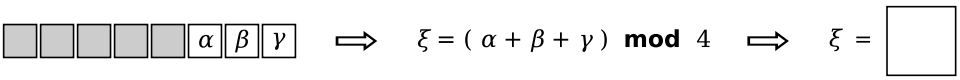
Prova 1 – Estruturas de Dados (INE5408) – 30mar2021

Ciências da Computação – Universidade Federal de Santa Catarina

Atenção: Esta prova deve ser respondida individualmente até amanhã, dia 31, às 12h (meio-dia). A submissão consiste em um único documento em "pdf" ou "odt" ou "docx" com todas as respostas digitadas ou digitalizadas (em caso de utilização de fotografias, verifique se há iluminação suficiente e resolução apropriada).

Defina ξ em função dos últimos três dígitos de sua matrícula:



1. *(2,0pt)* Você dispõe de toda a implementação de uma lista em vetor (`ArrayList`), mas precisa escrever um novo método, conforme descrição a seguir. Documente o código com o tratamento de todas as condições de contorno adotadas:

   - **Para alunos com $\xi = 0$**, pede-se a concatenação ou multiplicação da lista m vezes.
     Exemplo:
     - **Lista de entrada**: [A, B, C]
     - Para $m = 3$:
       - **Lista resultante**: [A, B, C, A, B, C, A, B, C]

     Sugestão de protótipo: `void multi(int m);`

   - **Para alunos com $\xi = 1$**, pede-se a duplicação de d elementos de cada extremidade da lista; se $d = 1$, duplica-se o elemento do início e o do fim da lista; se $d = 2$, duplicam-se dois elementos do início e dois elementos do fim; e assim por diante.
     Exemplo:
     - **Lista de entrada**: [A, B, C, D, E]
     - Para $d = 2$:
       - **Lista resultante**: [A, B, A, B, C, D, E, D, E]

     Sugestão de protótipo: `void shell(int d);`

   - **Para alunos com $\xi = 2$**, pede-se, na **parte 1**: a remoção de r elementos de cada extremidade da lista; se $r = 1$, remove-se o elemento do início e o do fim da lista; se $r = 2$, remove-se dois elementos do início e dois elementos do fim; e assim por diante; e, na sequência, na **parte 2**: criar um espelho dos elementos existentes.
     Exemplo:
     - **Lista de entrada**: [A, B, C, D, E, F, G]
     - Para $r = 2$:
       - **Lista resultante da parte 1**: [C, D, E]
       - **Lista resultante da parte 2**: [C, D, E, E, D, C]

     Sugestão de protótipo: `void peel_mirror(int r);`

   - **Para alunos com $\xi = 3$**, pede-se a replicação em x vezes de cada um dos elementos da lista.
     Exemplo:
     - **Lista de entrada**: [A, B, C, D]
     - Para $x = 3$:
       - **Lista resultante**: [A, A, A, B, B, B, C, C, C, D, D, D]

     Sugestão de protótipo: `void multi_item(int x);`

2. Uma aplicação exige um vetor com todos os dígitos de sua matrícula para construir um fila de pilhas e, em seguida, efetuar um determinado processamento. Segue código completo:

```
1  #include <iostream>
2  #include "array_stack.h"
3  #include "array_queue.h"
4
5  using namespace structures;
6
```

```
7
8  ArrayQueue< ArrayStack<int>* > *fila_de_pilhas(int vet[8]) {
9      int v;
10
11     ArrayStack<int> *p1 = new ArrayStack<int>(8);
12     ArrayStack<int> *p2 = new ArrayStack<int>(8);
13
14     for (int i = 0; i < 8; i = i + 2) {
15         p1->push(vet[i]);
16         p2->push(vet[i+1]);
17     }
18
19     ArrayQueue< ArrayStack<int>* > *fila = new ArrayQueue< ArrayStack<int>* >(3);
20
21     fila->enqueue(p1);
22     fila->enqueue(p2);
23
24     return fila;
25 }
26
```

```
27
28 int processa( ArrayQueue< ArrayStack<int>* > *fila ) {
29     int v[2] = {0, 0};
30     int i = 0;
31     while ( ! fila->empty() ) {
32         ArrayStack<int> *pilha;
33         pilha = fila->dequeue();
34         while ( ! pilha->empty() ) {
35             v[i] = v[i] + pilha->pop();
36         }
37         i++;
38     }
39     return v[0] - v[1];
40 }
41
```

```
42
43 int main() {
44     int vet[8] = { ___ , ___ , ___ , ___ , ___ , ___ , ___ , ___ };
45
46     auto fila = fila_de_pilhas(vet);
47     int v = processa(fila);
48
49     std::cout << "--> " << v << std::endl;
50
51     return 0;
52 }
```

coloque os dígitos de sua matrícula

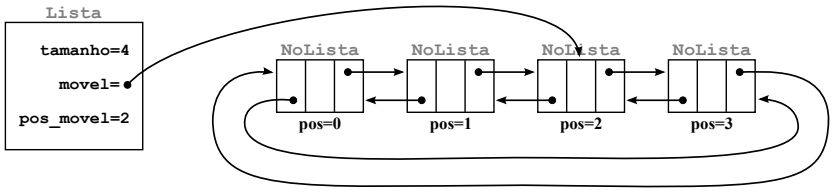
Pede-se:

(a) *(1,0pt)* Faça um desenho da fila de pilhas `fila` criada pela função `fila_de_pilhas()`.

(b) *(1,0pt)* Determine o valor de `v` calculado pela função `processa()`.

(c) *(1,0pt)* Explique as linhas 8 a 40 do código.

3. *(2,5pt)* Considere a classe `Lista` do desenho a seguir para uma lista dinâmica duplamente encadeada e circular, contendo um inteiro para a quantidade atual de elementos (`tamanho`), um ponteiro para um nó qualquer (`movel`) e um inteiro para indicar a posição desse nó (`pos_movel`), de forma que o único ponteiro (`movel`): • em uma operação de busca, aponte para o nó procurado (a figura abaixo refere-se à busca pelo dado na posição 2); • em uma inserção, aponte para o nó recém inserido; e, • em uma remoção, aponte para o nó à direita (se houver) do nó removido. A estrutura `NoLista` de nó da lista possui um tipo `T` (`dado`), ponteiros para anterior (`ant`) e próximo (`prox`).



Considerando que o ponteiro `movel` pode se encontrar em qualquer lugar e **não se pode acrescentar nenhum outro ponteiro à classe** `Lista`, implemente a operação de:

- **Para alunos com $\xi = 0$:**
  Remoção do fim: $\Rightarrow$ `T removeDoFim();`
- **Para alunos com $\xi = 1$:**
  Remoção do início: $\Rightarrow$ `T removeDoInicio();`
- **Para alunos com $\xi = 2$:**
  Inserção no início: $\Rightarrow$ `void insereNoInicio(T dado);`
- **Para alunos com $\xi = 3$:**
  Inserção no fim: $\Rightarrow$ `void insereNoFim(T dado);`

4. *(2,5pt)* Um usuário quer selecionar k elementos de posições lógicas consecutivas (p, $p+1$, $p+2$, ..., $p+k-1$) existentes em uma lista, reposicionando-os no fim da mesma (na mesma ordem original). Exemplo:

- **Lista de entrada**: [A, B, C, D, E, F, G, H]
- Para $p = 2$, $k = 5$:
  - **Elementos selecionados**: [C, D, E, F, G]
  - **Lista resultante**: [A, B, H, C, D, E, F, G]

Pede-se:

(a) Implemente um método da Classe Lista para esses k reposicionamentos a partir do índice p. Documente o código com o tratamento de todas as condições de contorno adotadas.
Sugestão de protótipo:
`void reposicionaSubLista(int p, int k);`

(b) Para o código do item (a), em função no número total de elementos n da lista e dos valores p e k definidos, escreva uma equação que descreva o número de operações (considere cada operação como sendo a atualização de um ponteiro para nó de lista), se for utilizada uma:

i. Lista dinâmica simplesmente encadeada com um único ponteiro para o início;
ii. Lista dinâmica simplesmente encadeada com um ponteiro para o início e outro para o fim;
iii. Lista dinâmica duplamente encadeada com um ponteiro para o início e outro para o fim
iv. Lista dinâmica duplamente encadeada com um único ponteiro móvel, conforme o exercício anterior

Boa prova!