Programação avançada
UFPE 2023.2
Lista de exercícios: estruturas de dados

- Cada aluno deve resolver os exercícios. Não é um trabalho em dupla.
- O código deve compilar e rodar sem erros.

Exercício 1: implementação de pilha com arranjo

- Complete o arquivo `pilha_arranjo.cpp` para implementar uma pilha usando um arranjo.
- A função `main` não pôde ser modificada.
- A função `main` chama as funções `push` e `pop` aleatoriamente.
- A chamada das funções deve gerar as seguintes mensagens no terminal:

  ”42 pushed to the stack”

  ”42 popped from the stack”

  ”Empty stack”

  ”Full stack”

Exercício 2: implementação de fila com lista ligada

- Complete o arquivo `fila_lista.cpp` para implementar uma fila usando uma lista ligada.
- A função `main` não pôde ser modificada.
- A função `main` chama as funções `enQueue` e `deQueue` aleatoriamente.
- A chamada das funções deve gerar as seguintes mensagens no terminal:

  ”Enqueue 142”

  ”Dequeue 42”

  ”Empty queue”

Exercício 3: lista ligada sem números duplicados

- Complete o arquivo `lista_duplicado.cpp` para implementar uma lista ligada que não contém números duplicados.
- A função `main` não pôde ser modificada. As classes `Node`, `List` e `Hashtable` devem ser implementadas.
- Na primeira parte da função `main`, uma lista ligada de 100 nodos é criada a partir de números escolhidos aleatoriamente.
- A lista é **duplamente** ligada.

- Na segunda parte da função `main`, a função `removeDuplicates` é chamada para remover os nós com números duplicados. Essa função usa uma **tabela de dispersão** (hash table) para identificar os números duplicados.
- A tabela de dispersão usa o método da divisão (101 é o menor número primo maior que 100) e o endereçamento aberto.

Exercício 4: Árvore de busca binária

O objetivo deste execicio é implementar uma árvore de busca binária.
A árvore de busca binária deve ser implementada com as seguintes funções:

- Adicionando novo dado.
- Busca de dado a partir de uma chave.
- Eliminação de dado.
- Impressão dos dados contidos na árvore.

Usando estas funções básicas, o programa deve:

- Construir uma árvore de busca binária contendo todos os dados contidos no arquivo `data.txt`.
- Procurar na árvore de busca binária os dados armazenados no arquivo `search.txt`.
- Eliminar os dados armazenados no arquivo `delete.txt`.

Exercício 5: Algoritmo de Dijkstra

- O grafo deve ser representado por uma matriz de adjacências.
- O arquivo `graph.txt` contém o grafo.
- A chamada da função `printPath(0,)` deve gerar as seguintes mensagens no terminal:

  ”Path from 0 to node 2”

  ”0 3 1 2”