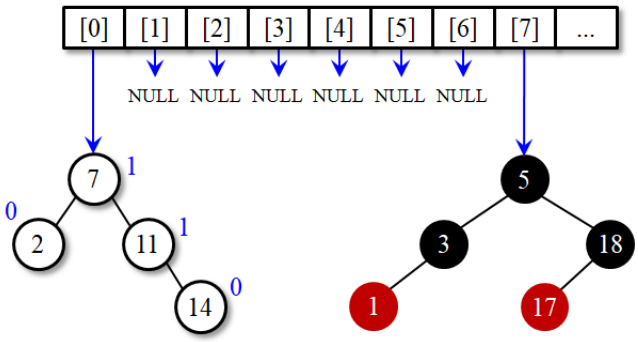
Avaliação Substitutiva

(individual – entregue via Run Codes)

Exercício Único: escreva, em linguagem C pura (sem nenhum comando próprio de C++), um programa que leia um valor N e aloque um vetor de N posições para implementação de uma tabela hash usando open hashing (closed addressing). A função de espalhamento é o método da divisão e, para o tratamento de colisões, cada posição do vetor define uma árvore ao invés de uma lista. Se a posição do vetor for par, a estrutura deve ser uma árvore AVL (Aula 6), enquanto que se a posição do vetor for ímpar, a estrutura deve ser uma árvore Left Leaning Red-Black (Aula 7), como mostrado na figura a seguir:



Após alocar dinamicamente o vetor, leia outro número inteiro representando uma opção do usuário. As ações a serem executadas dependerão da opção lida, sendo:

- Se a opção lida for 1, seu programa deve ler um novo valor inteiro e inserir na tabela hash. Note que a função de espalhamento indica a posição do vetor, sendo que o valor lido deve ser corretamente inserido na árvore correspondente.
- Se a opção lida for 2, seu programa deve ler um novo valor inteiro e pesquisá-lo na tabela hash. Se o valor pesquisado estiver presente na árvore correspondente, imprima-o. Se o valor pesquisado não estiver presente, imprima um "x" minúsculo como resposta. Pule uma linha após imprimir a resposta.
- Se a opção lida for 3, seu programa deve ler um novo valor inteiro e removê-lo da tabela hash (isto é, da árvore correspondente).
- Se a opção lida for 4, para cada posição da tabela hash, seu programa deve executar o percurso em pré-ordem na árvore correspondente e imprimir os valores dos elementos em uma única linha, separados por um único espaço, de acordo com o seguinte formato (que representa a resposta da figura acima):

```
[0]: 7(1) 2(0) 11(1) 14(0)
[1]:
[2]:
[3]:
[4]:
[5]:
[6]:
[7]: 5(B) 3(B) 1(R) 18(B) 17(R)
etc...
```

  Entre colchetes, está a posição do vetor. Para cada posição, imprima os elementos daquela árvore. Os parênteses indicam, no caso da AVL, o fator de balanceamento daquele nó e, no caso da Red-Black, a cor daquele nó, sendo R para vermelho (red) e B para preto (black). Pule uma linha ao finalizar.
- Se a opção lida for 9, seu programa deve finalizar. Note que seu programa encerra apenas nesta opção. Ao terminar de processar qualquer outra opção, o programa deve retornar ao loop inicial para ler a próxima ação que o usuário deseja executar.

Observação 1: este exercício substitui a menor nota entre as atividades dadas ao longo da disciplina.

Observação 2: você já tem todas as funções (inserção, remoção e busca) da AVL prontas (tema dos exercícios da Aula 6). Você já tem as funções de inserção e busca da Red-Black prontas (tema dos exercícios da Aula 7). Você deve, portanto, completar com as funções de remoção na árvore Red-Black.

Observação 3: você também já tem o método da divisão e a inserção em tabelas de espalhamento do tipo open hashing prontos (tema dos exercícios da Aula 9). Sua tarefa é simplesmente substituir a lista encadeada por uma árvore e fazer com que as três estruturas (hash, AVL e Red-Black) trabalhem em conjunto.

Observação 4: no caso da árvore AVL, lembrar de calcular o fator de balanceamento de cada nó como: *FB = Altura direita – Altura esquerda*

Observação 5 (muito importante!): não deixe para a última hora! Não é difícil, porém, é trabalhoso fazer todas as estruturas funcionarem em conjunto. Lembre-se que imprevistos acontecem. O servidor do Run Codes pode cair ou apresentar instabilidades, a energia elétrica pode acabar, você pode ficar sem internet, etc... É por sua conta e risco! Não haverá prorrogação de prazo.
