Mini-teste Rubro negra

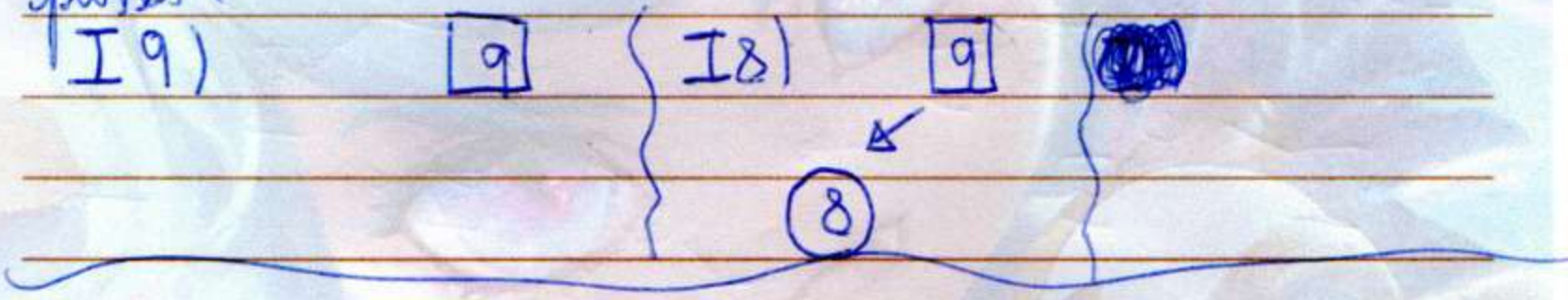
R = remover
I = inserir

1)

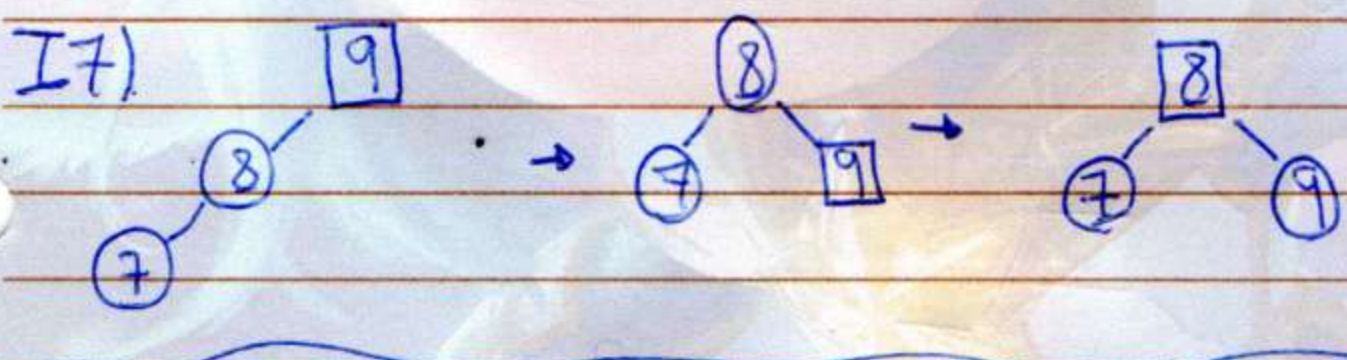
Legenda: □ = negro
○ = vermelho

passos:

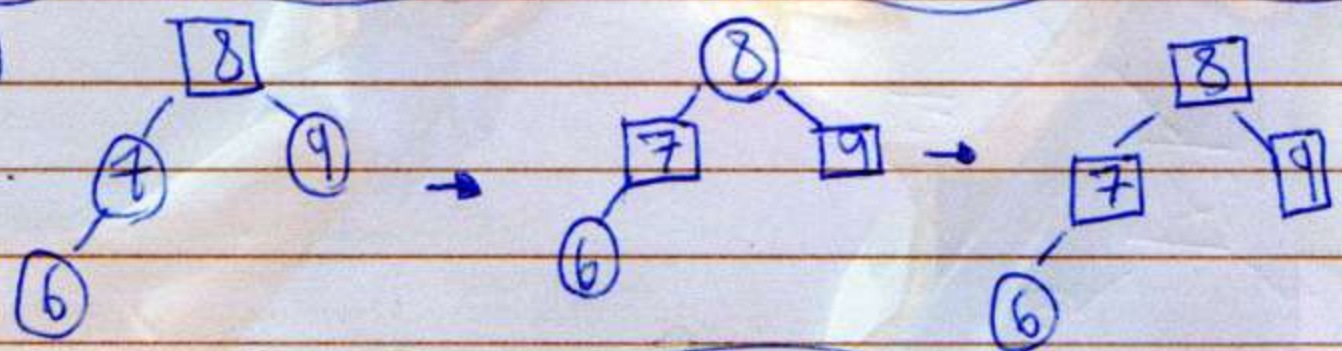
I 9)



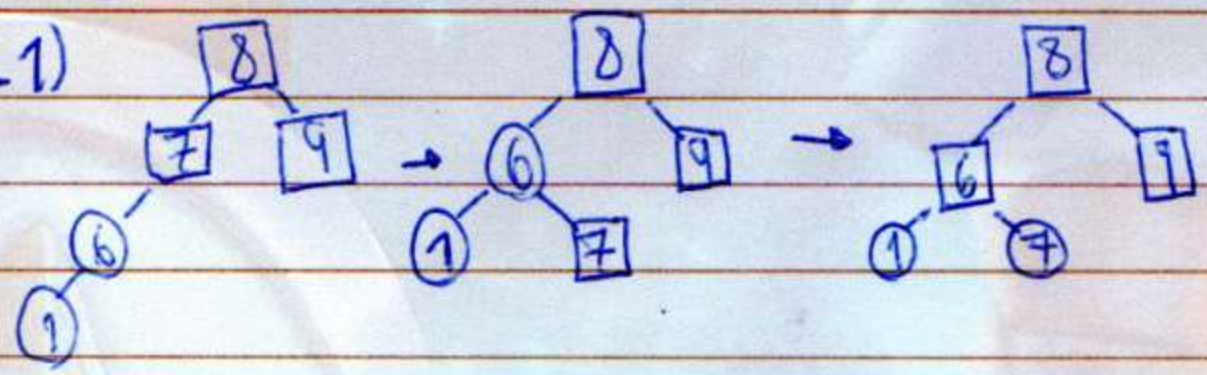
I 7)



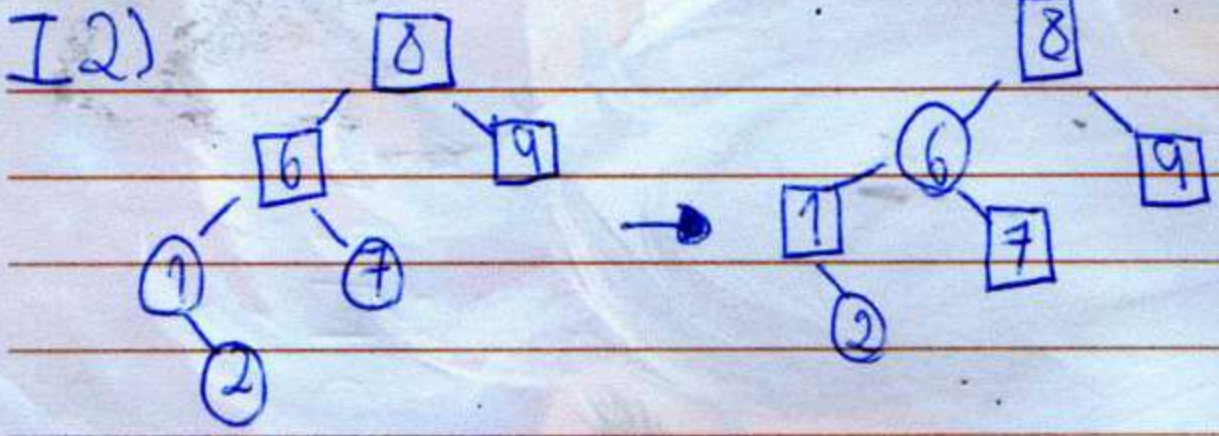
I 6)



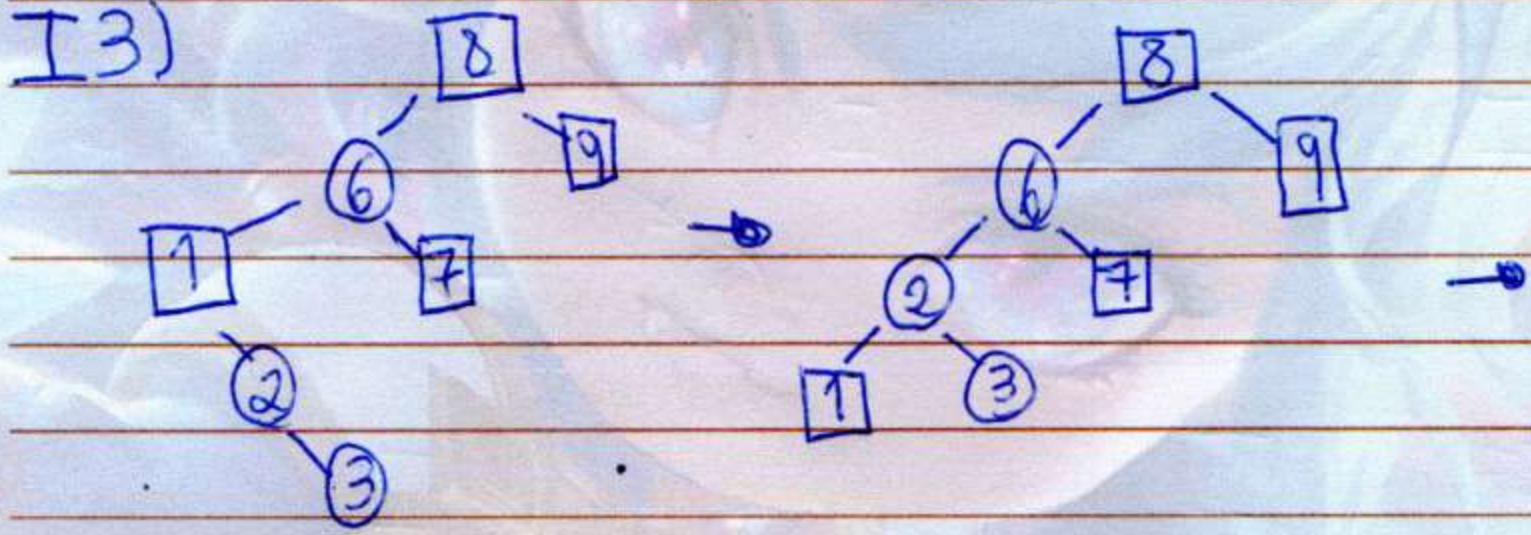
I 1)

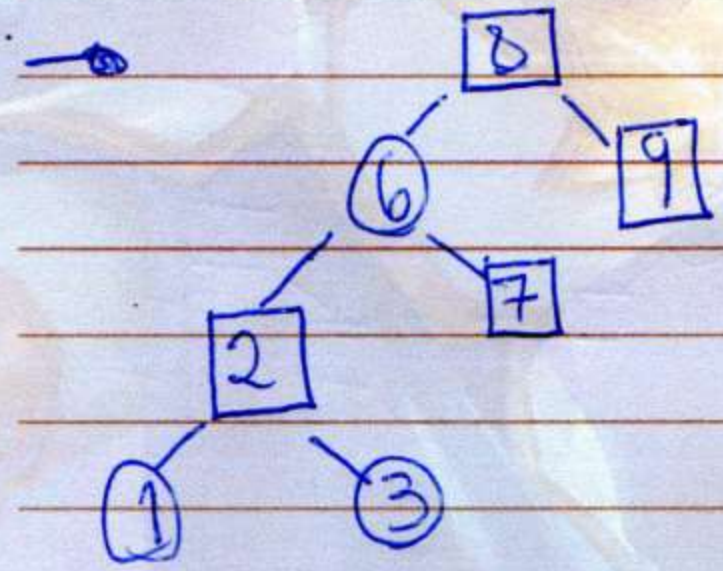


I2)



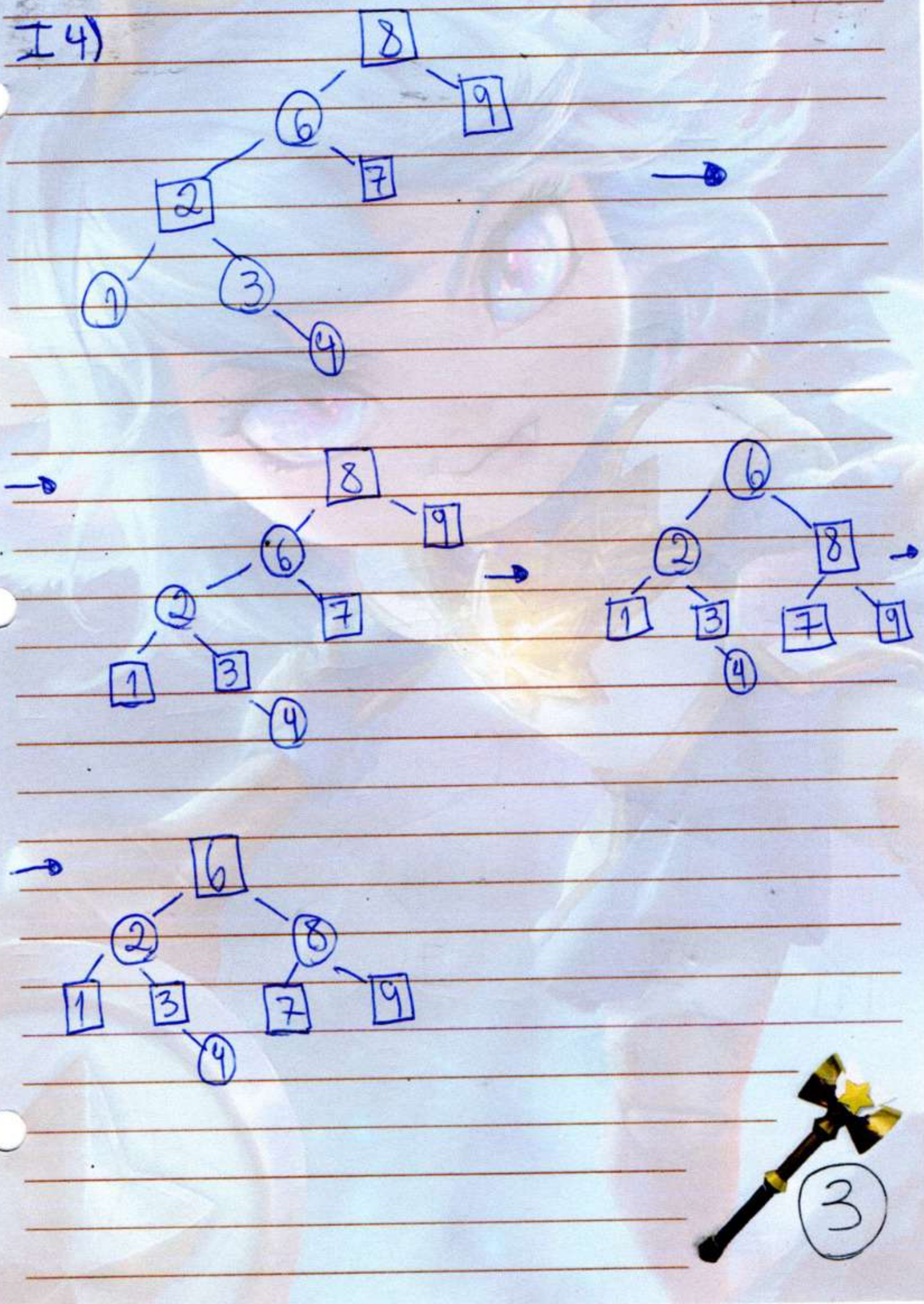
I3)



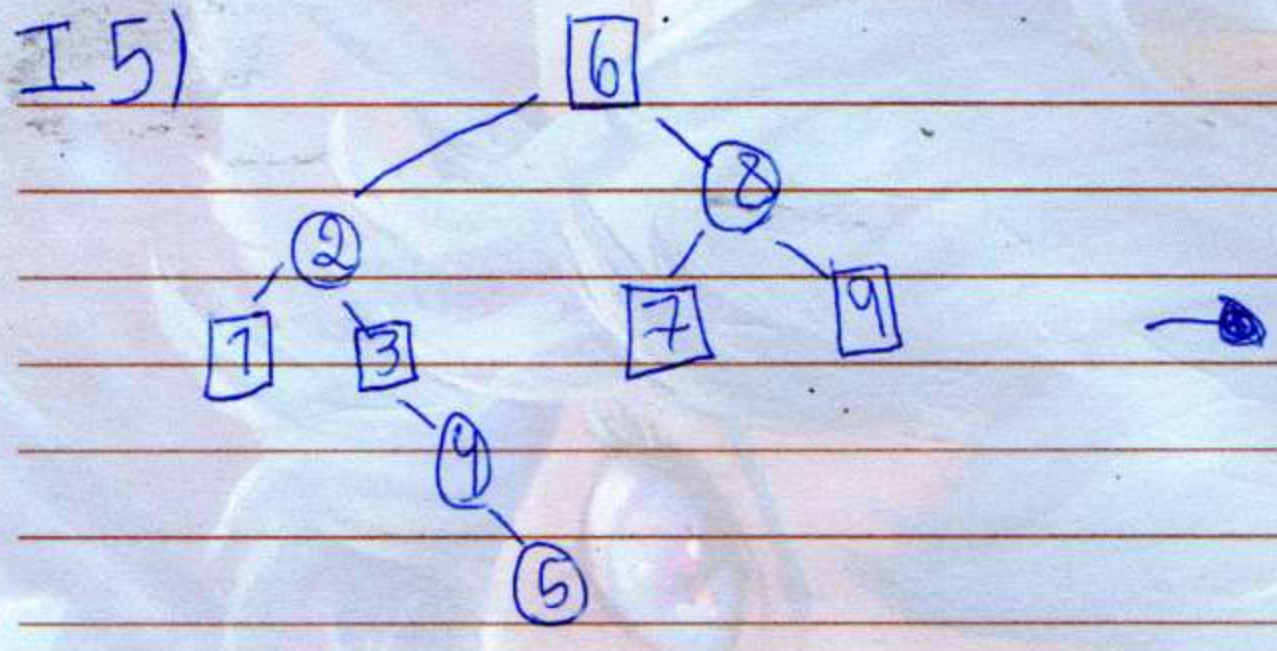


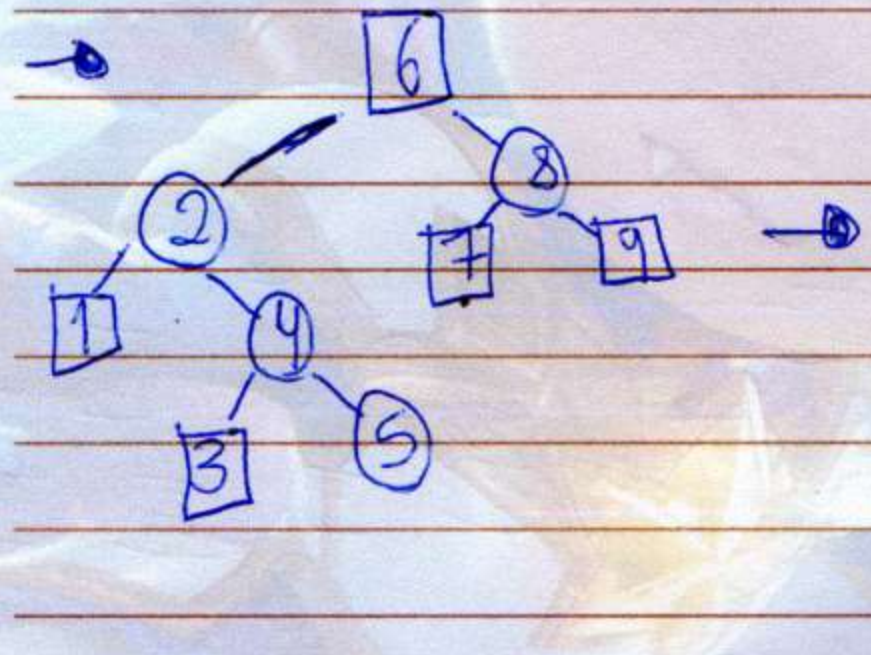
(2)

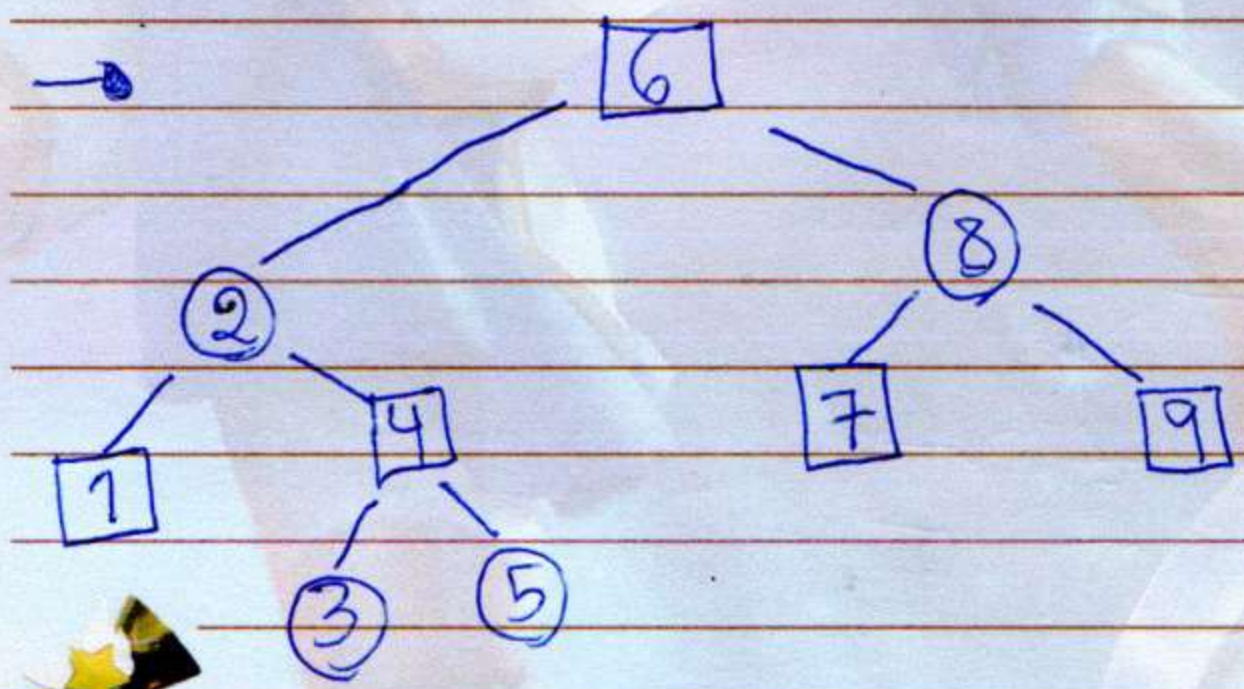
I 4)



I 5)

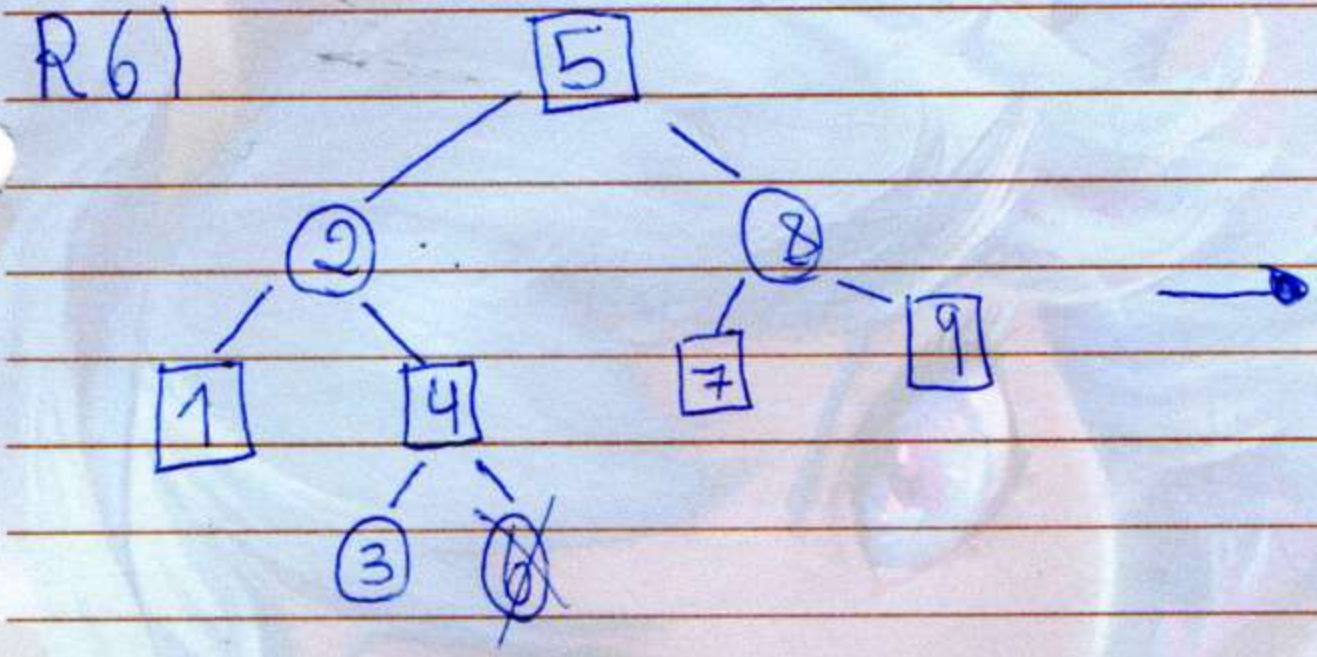


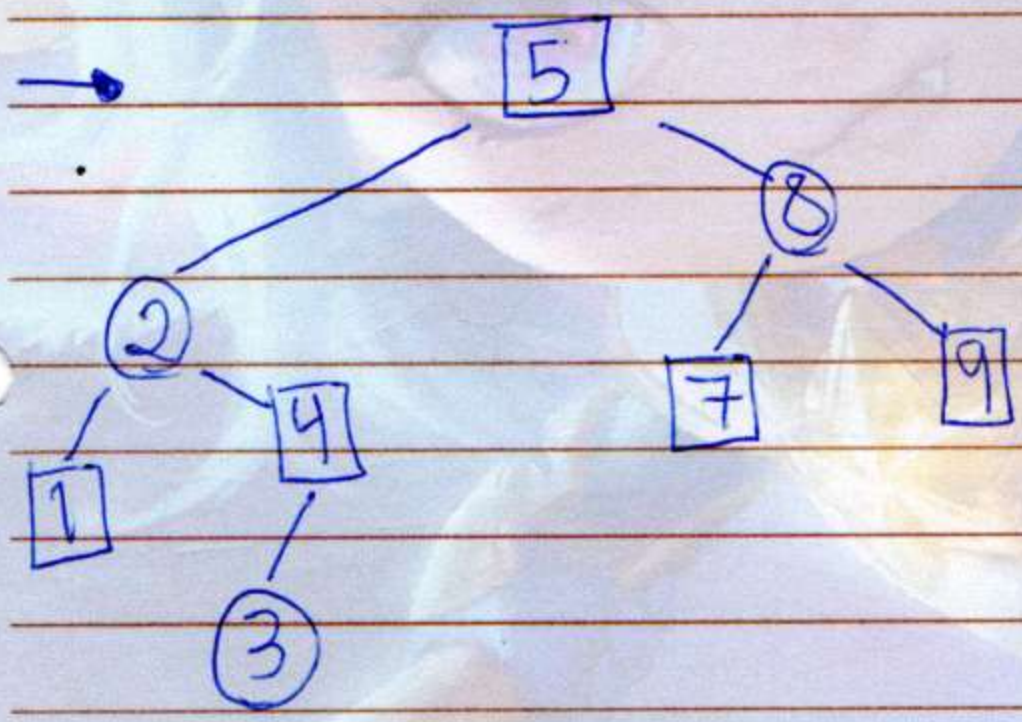




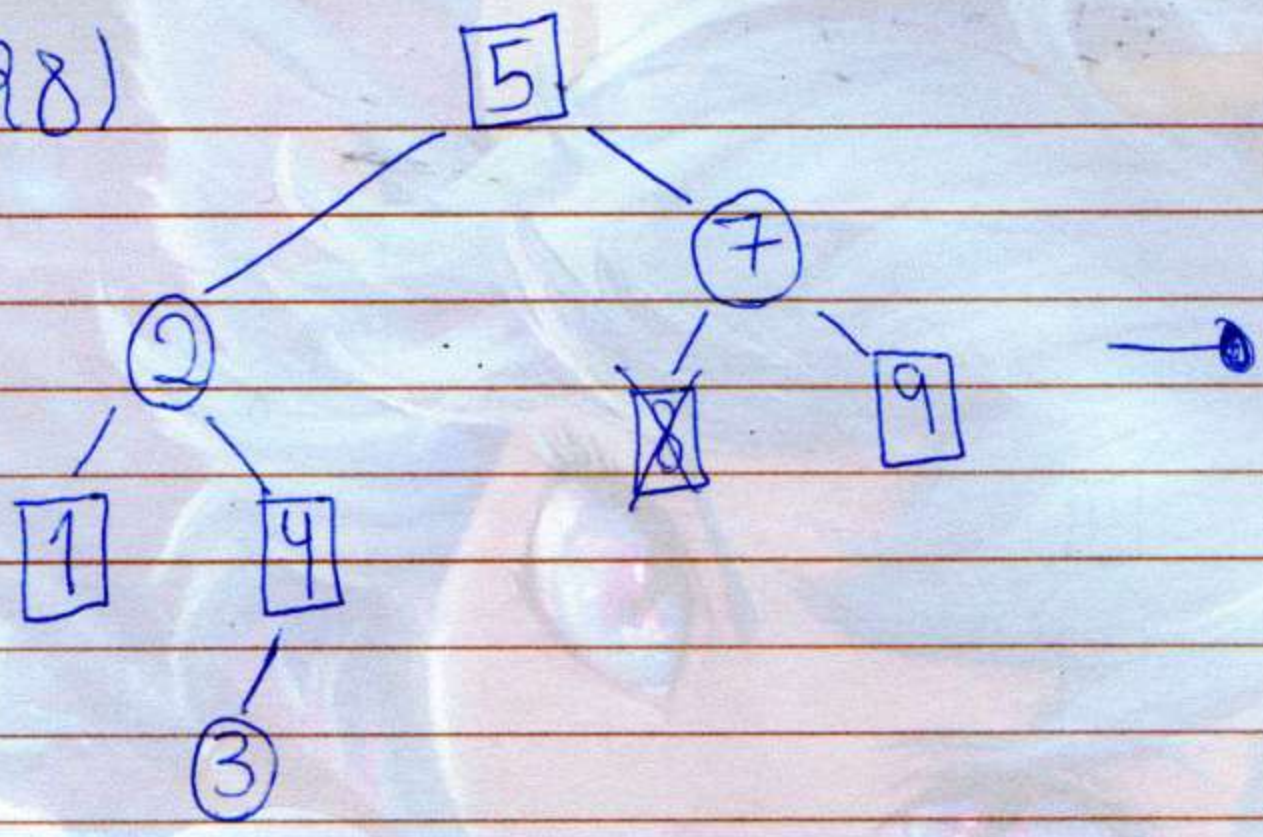
4

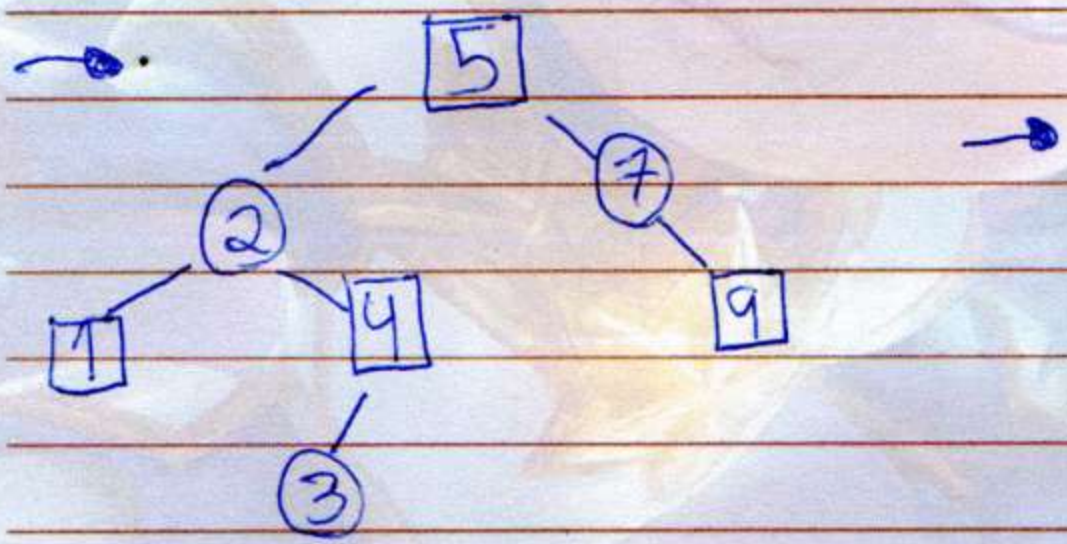
R6)

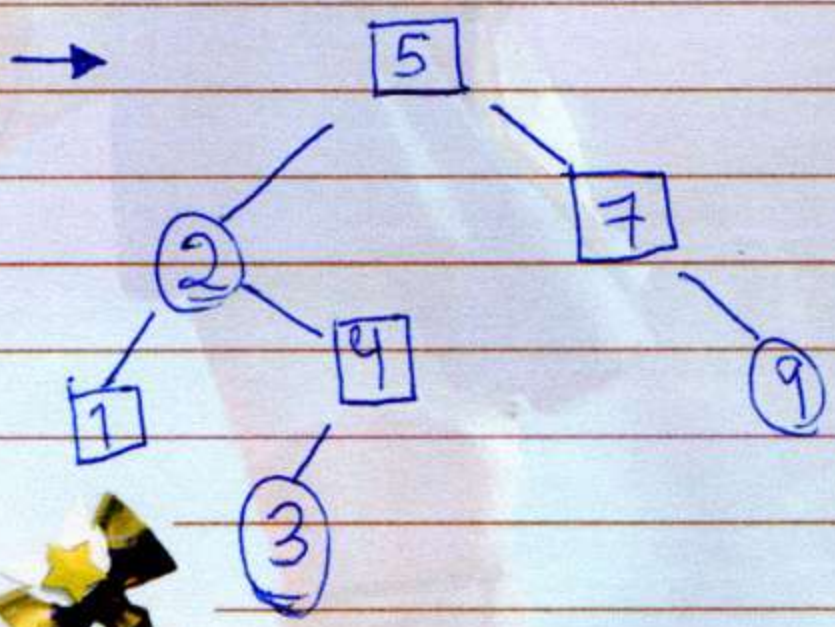




A8)







2)

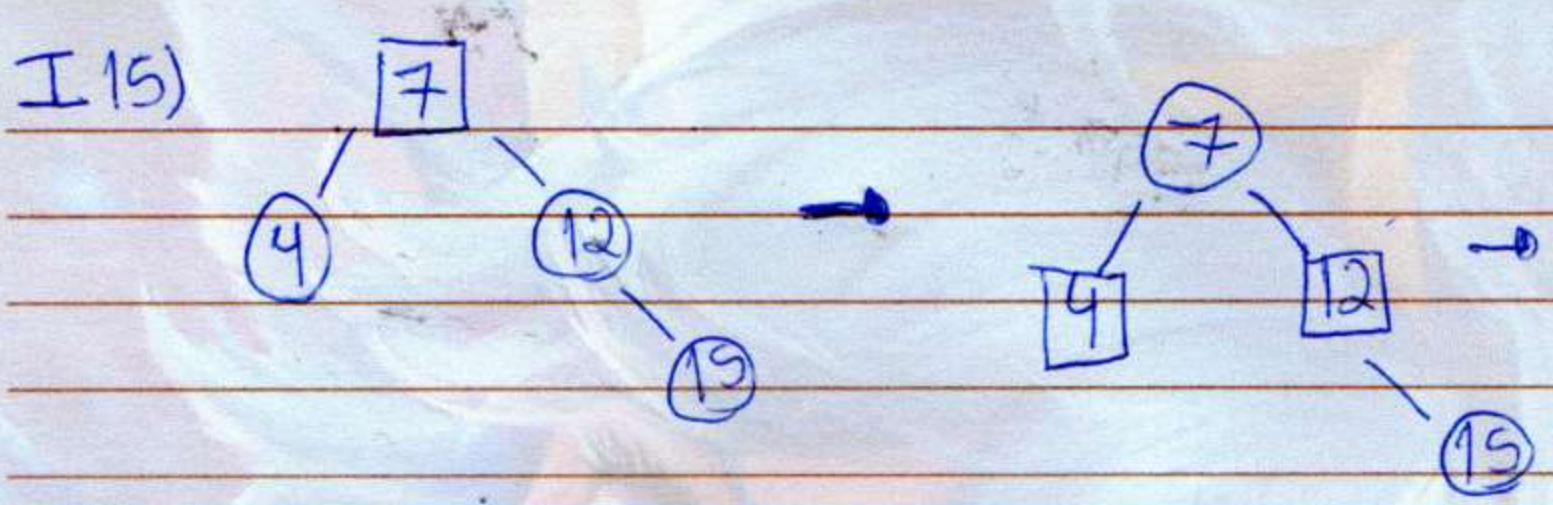
passos:

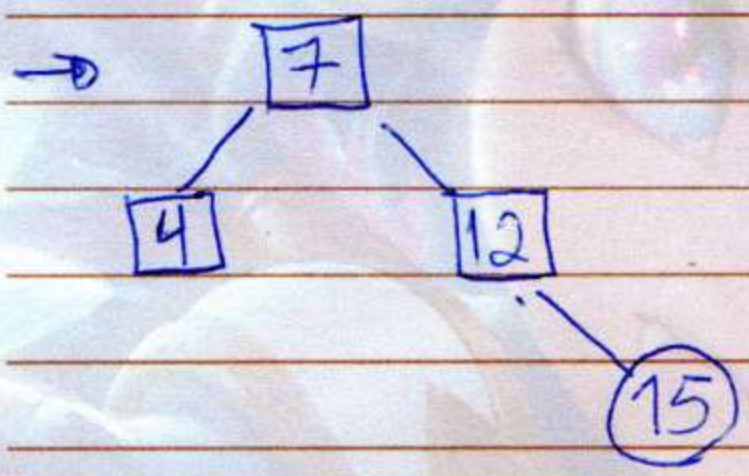
I 4) [4]

I 7) [4] — (7)

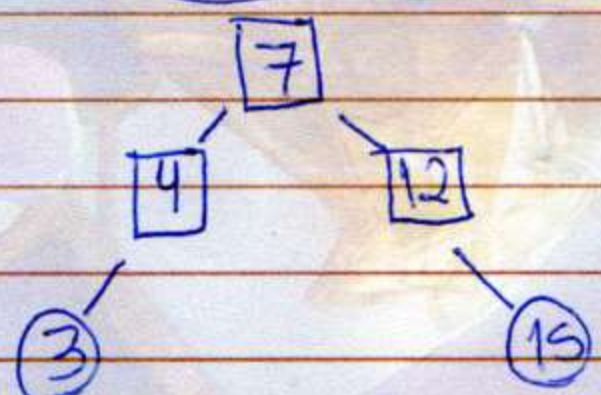
I 12) [4] — (7)
(7) → [4] — (12)
→ [7]
(4) — (12)

I 15)

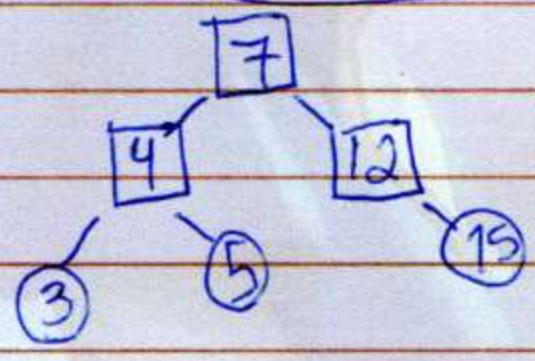




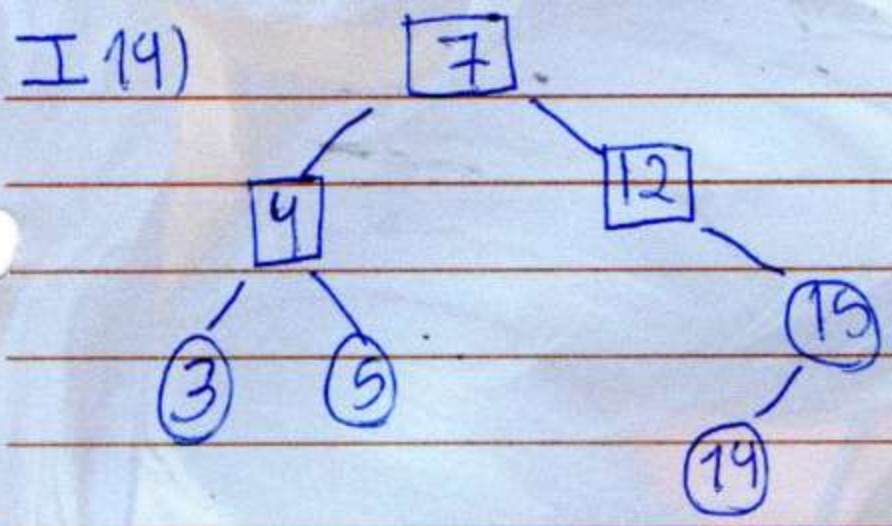
I 3)

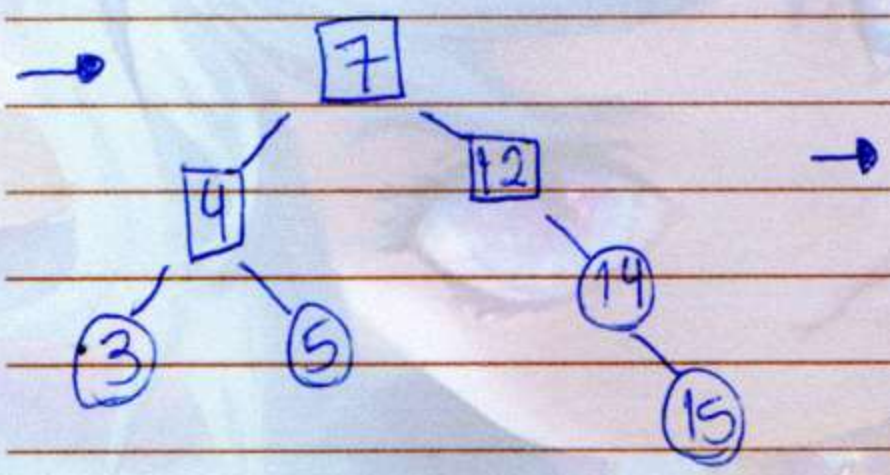


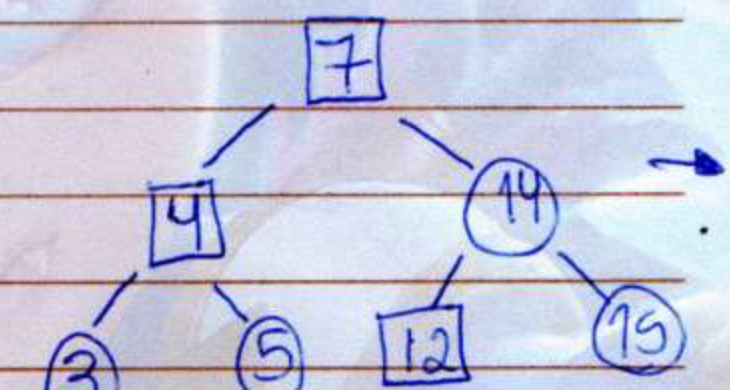
I 5)

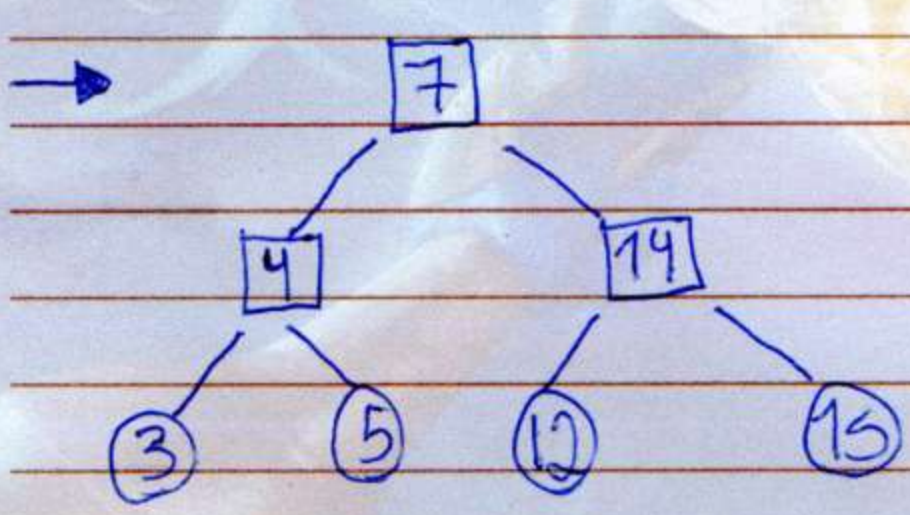


I 14)

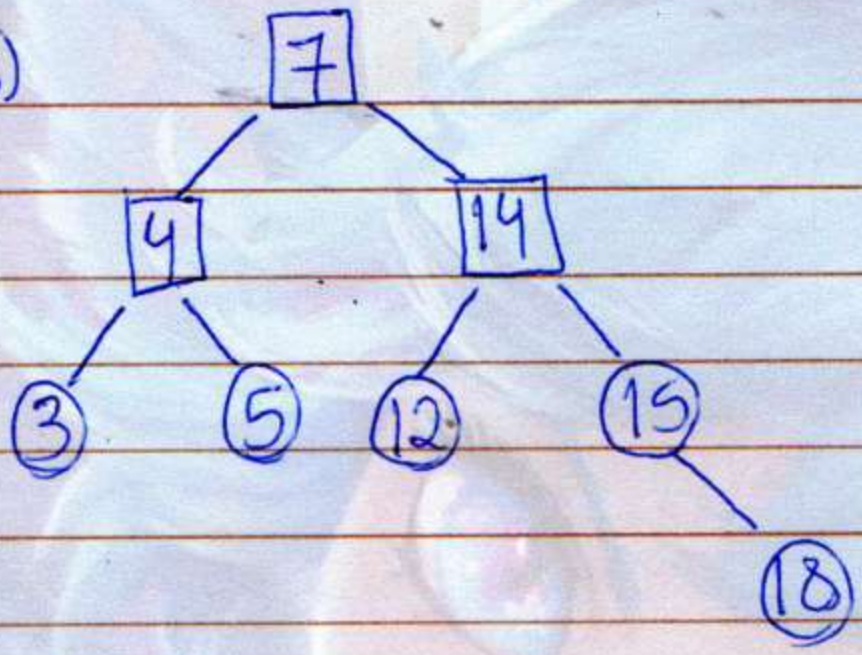


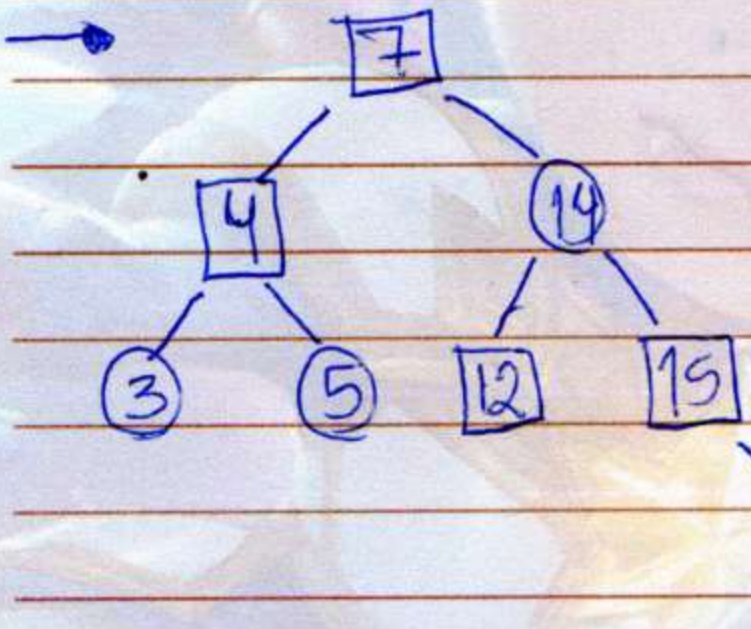






I 18)





Miniteste - Árvore rubro-negra

3 - A raiz deve ser sempre preta, logo seus filhos serão vermelhos com exceção das folhas. As folhas são sempre pretas. Logo, qualquer filho da raiz será vermelho.

4- O procedimento de remoção de um nó em árvores rubro-negras se inicia com uma etapa de busca e remoção como nas árvores binárias de busca convencionais. Porém, se alguma propriedade vermelho-preta for violada, a árvore deve ser rebalanceada.
Se a remoção for de um nó vermelho, esta é realizada sem problemas, pois todas as propriedades da árvore se manterão intactas. Se o nó for preto, a quantidade de nós pretos em pelo menos um dos caminhos da árvore será alterada, o que implica que algumas operações de rotação e/ou alteração de cor sejam feitas para manter o balanceamento da árvore.
Para ocorrer a remoção de um nó em uma árvore rubro-negra é necessário que esse nó tenha no máximo um filho que não seja folha, a maior preocupação na remoção está em não quebrar as regras básicas. O primeiro caso verifica se o pai do nó não é nulo, se for vai para o segundo caso. No segundo caso, se o nó e seu pai forem pretos e seu irmão for vermelho o pai deve ser pintado de vermelho e o irmão de preto e então se o nó for filho esquerdo, faz a rotação à esquerda de seu pai e vai pro próximo caso, se for filho direito, rotaciona o pai à direita e vai pro próximo caso.Nesse caso, se o pai do nó, o irmão, o filho esquerdo e direito do irmão forem todos pretos, pinta o irmão de vermelho e volte para o primeiro caso com o pai do nó, se não forem vai pro próximo caso. No quarto caso, se o irmão e o filho esquerdo e direito do irmão forem pretos e o pai do nó for vermelho, deve pintar o irmão de vermelho e o pai do nó de preto, se não deve prosseguir para o próximo caso. No quinto caso, se o nó for filho esquerdo e o filho direito do irmão for preto deverá pintar o irmão de vermelho e o filho esquerdo do irmão de preto e aí sim rotacionar à direita o irmão, mas se o nó for filho direito deverá pintar o irmão de vermelho e o filho direito do irmão de preto e então rotacionar para esquerda o irmão, indo para o último caso. Ao chegar no último caso deverá pintar o pai do nó de preto, caso o nó seja filho esquerdo, pinta o filho direito do irmão do nó de preto e rotaciona o pai do nó para a esquerda, se o nó for filho direito, pinta o filho esquerdo do irmão de preto e rotaciona o pai para direita. Ao sair do encadeamento de casos poderá ser feita a remoção do nó naturalmente. O algoritmo de remoção é O(log n).

5 - Tanto o caminho mais curto S quanto o caminho mais longo S terão o mesmo número de nós pretos, conforme a propriedade 5. Por causa da propriedade 4, cada nó vermelho deve ter um pai preto e filhos pretos. Isso significa que o número de nós vermelhos deve ser menor ou igual ao número de nós pretos em qualquer caminho em uma árvore rubro-negra válida.
A maior diferença, então, pode ser obtida se S contém apenas nós pretos e L contém o mesmo número de nós pretos com o máximo possível de nós vermelhos adicionados a ele, que é 2S.