

Aluno(a):

- Considere a seguinte sequência de operações:
push(7), push(2), pop(), push(4), pop(), push(5),
push(8), pop(), pop(), push(3), push(1), pop().
 - A soma dos elementos remanescentes em uma *fila*, inicialmente vazia, após essa sequência de operações será
 - A soma dos elementos remanescentes em uma *pilha*, inicialmente vazia, após essa sequência de operações será

- Considere a seguinte sequência de operações aplicadas tanto a uma fila quanto a uma pilha: push(5), push(7), pop(), push(1), pop(), push(6), pop() e push(4).

- Após a realização das operações, a soma dos elementos remanescentes na fila será menor do que a soma dos elementos remanescentes na pilha.
- Ao longo da execução das operações, a soma dos elementos armazenados na fila nunca será menor do que a soma dos elementos armazenados na pilha.
- Ao longo da execução das operações, a soma dos elementos armazenados na pilha nunca será menor do que a soma dos elementos armazenados na fila.
- A diferença entre a soma dos elementos presentes na pilha e na fila nunca ultrapassa 5.

A respeito das afirmações acima, pode-se dizer que estão corretas as alternativas:

- I e II
- II e IV
- I e IV
- II e III
- I, II, III e IV

- Considere as seguintes estruturas, utilizadas para implementar uma fila:

```
typedef struct{
    int dado;
    struct item *next;
}item;

typedef struct{
    item *inicio;
    item *fim;
}fila;
```

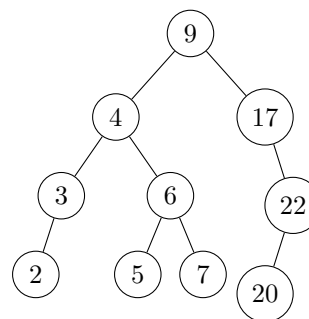
Complete o código abaixo para implementar a função *pop* em uma fila, assumindo que o parâmetro **f* é um ponteiro para o primeiro elemento da estrutura.

```
1 int pop(fila *f){
2     int dado = f->inicio->dado;
3     _____;
4     if(f->inicio==NULL)
5         _____;
6     return dado;
7 }
```

Linha 3: _____
Linha 5: _____

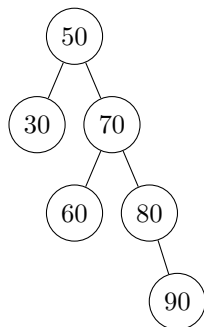
- Sobre tabelas hash, pode-se afirmar que
 - Sendo n_1 um número par e n_2 um número ímpar tal que $n_1 < n_2$, a probabilidade de colisões é menor para tabelas hash de tamanho n_1 do que para tabelas de tamanho n_2 .
 - Em tabelas livres de colisões, a busca por um elemento tem, no pior caso, complexidade igual à complexidade da busca em árvores binárias de pesquisa.
 - Na resolução de colisões por encadeamento, a busca por um elemento, no pior caso, terá complexidade semelhante à busca em listas encadeadas.
 - Em tabelas livres de colisões, tem-se a garantia, independente da função hash utilizada, de que os elementos estarão dispostos na ordem em que foram inseridos. O primeiro elemento a ser ocupará a primeira posição da tabela, o segundo elemento ocupará a segunda posição e assim por diante.
- Considere uma tabela hash de tamanho t que armazena chaves cujos valores cujas chaves são números inteiros. O cálculo do hash para uma chave k é dado por $h(k) = k \% t$ (método da divisão). Assumindo que a tabela está inicialmente vazia, qual será o número de colisões caso sejam inseridos elementos com as chaves 24, 28, 30 e 36 em tabelas com os seguintes tamanhos:
 - $t = 4$: _____
 - $t = 7$: _____
 - $t = 12$: _____

Para as questões 6 a 7, considere a árvore da figura abaixo:



- Escreva abaixo a sequência de números que será impressa caso a árvore seja percorrida
 - Em pré-ordem: _____
 - Em ordem: _____
 - Em pós-ordem: _____
- Caso o nó armazena o número 4 seja excluído e a árvore permaneça sendo uma árvore binária de pesquisa, a do valor de suas folhas será _____.

8. Transforme a árvore abaixo em uma árvore AVL.



9. Desenhe a árvore binária de pesquisa (não balanceada) gerada pela inserção da seguinte sequência de números: 50, 30, 80, 20, 70, 40, 90, 5, 25, 73, 2, 75.

10. Desenhe a árvore AVL gerada pela inserção da seguinte sequência de números: 50, 30, 80, 20, 70, 40, 90, 5, 25, 73, 2, 75.