Universidade Federal de Santa Catarina - Campus Blumenau
Departamento de Engenharia de Controle, Automação e Computação **Avaliação 2**
BLU3202 — **Algoritmos e Estruturas de Dados** 2022/2

Aluno(a): .....

1. Considere a seguinte sequência de operações:
   `push(8), push(3), pop(), push(5), pop(), push(7), push(9), pop(), pop(), push(1), push(4), push(2), pop()`.

   (a) A soma dos elementos remanescentes em uma *fila*, inicialmente vazia, após essa sequência de operações será _____.

   (b) A soma dos elementos remanescentes em uma *pilha*, inicialmente vazia, após essa sequência de operações será _____.

2. Seja *push(n)* a operação que insere um número n em uma *pilha*. Considere a execução desta operação, com n assumindo os valores 1, 2, 3, 4 e 5, nesta ordem. Em meio às inserções, pode haver remoções (*pop*). Indique a ordem das operações *push(n)* e *pop()* para que a sequência de números retirados da pilha com a operação *pop* seja:

   (a) 4,3,5,2,1 _____

   _____

   _____

   (b) 3,4,2,5,1: _____

   _____

   _____

   (c) 1,4,5,3,2: _____

   _____

   _____

3. Seja X uma pilha de caracteres. Seja $get(X)$ uma função cujo pseudocódigo é:

```
get(){
   // realiza a operação pop na pilha "X" e
   // armazena o elemento retirado na variável "c"
   pop(X,c);
   // imprime o elemento armazenado em "c"
   print(c);
}
```

   A combinação de operações *push* e *get* permite formar anagramas a partir de uma entrada. Por exemplo, a partir da sequência de caracteres A, R, T, S, pode-se obter o anagrama ARTS com a seguinte sequência de operações:

```
push(S);push(T);push(A);get();push(R);get();get();get();
```

   Considere a entrada, em sequência, dos caracteres A, B, C, D, E, F. Assinale os anagramas que podem ser produzidos a partir dessa entrada combinando operações *push* e *get*. Observação: cada escolha errada anula uma escolha certa.

   ( ) ACBFDE ( ) CDBEFA
   ( ) FEDCBA ( ) CFEBDA
   ( ) ABCDEF
   ( ) BDCFEA ( ) DBACEF

4. Considere as seguintes estruturas utilizadas para implementar uma fila:

```
typedef struct{          typedef struct{
   int dado;             item *inicio;
   struct item *next;    item *fim;
}item;                   }fila;
```

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

```
1  void push(fila *f, int dado){
2     item *novo =
3           malloc(sizeof(item));
4     novo->dado = dado;
5     novo->next = NULL;
6     if(f->fim!=NULL)
7        f->fim->next = novo;
8     else
9        ----------------;
10    -------- = novo;
11 }
```

   Linha 9: _____
   Linha 10: _____

5. Nas alternativas abaixo, assinale V para as verdadeiras e F para as falsas:

   ( ) Para encontrar uma chave em uma tabela hash, é examinar cada um dos elementos da estrutura até encontrar o valor procurado, de forma semelhante ao que é feito na busca por valores em vetores.

   ( ) Ao utilizar-se o método da divisão para o cálculo do *hash*, uma mesma chave será mapeada para a mesma posição em qualquer tabela hash, independente de seu tamanho.

   ( ) No pior caso, a busca por uma chave em tabelas hash em que as colisões são resolvidas por encadeamento tem custo computacional igual à busca por valores em uma lista encadeada.

   ( ) No pior caso, a busca por uma chave em tabelas hash em que as colisões são resolvidas por encadeamento tem custo computaciona igual à busca por valores em uma árvore binária de pesquisa balanceada.

   ( ) A sondagem linear aplica-se somente a tabelas hash que em que a quantidade de itens a serem armazenados é menor ou igual ao tamanho da tabela.

   ( ) No pior caso, a inserção de uma chave em uma tabela hash tem custo computacional igual em tabelas em que as colisões são resolvidas por encadeamento em comparação à solução de colisões por sondagem linear.

6. Considere a inserção das seguintes chaves na dada ordem em uma tabela hash de tamanho 7: {**282, 44, 251, 22, 129, 266, 300**} usando o método da divisão com a função hash **h(x) = x mod 7**. Mostre as tabelas resultantes resolvendo as colisões por encadeamento e por sondagem linear. Responda no seguinte formato: `0:[...], 1:[...], ...., n:[...]`

   _____

   _____

   _____

   _____

   _____

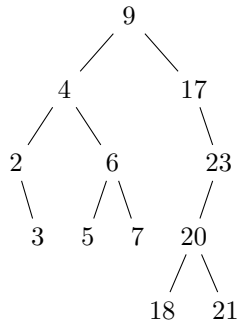
   _____

   _____

   _____

   _____

Para as questões 7 a 8, considere a árvore da figura abaixo:



7. Escreva abaixo a sequência de números que será impressa caso a árvore seja percorrida

   a) Em pré-ordem: _____

   b) Em ordem: _____

   c) Em pós-ordem: _____

8. Informe qual será a *soma* do valor das folhas da árvore após a exclusão dos nós indicados abaixo. Após a exclusão, a árvore deve continuar mantendo as propriedades de uma *árvore binária de pesquisa*.

   - 4: _____
   - 17: _____
   - 7: _____

9. Desenhe a árvore binária de pesquisa (não balanceada) gerada pela inserção da seguinte sequência de números: 50, 30, 80, 90, 60, 95, 64, 85, 65, 87, 70, 82, 35, 61, 89.

10. Seja uma árvore *completa*. São necessárias c comparações para constatar que a informação procurada não está armazenada nesta árvore. Informe a quantidade de nós desta árvore para os valores de c abaixo:

    (a) $c = 7$: _____

    (b) $c = 4$: _____

    (c) $c = 13$: _____