Universidade Federal de Santa Catarina - Campus Blumenau
Departamento de Engenharias
BLU3202 — **Algoritmos e Estruturas de Dados**

Avaliação - REC
2018/1

Aluno(a): .....

1. O que será impresso pelo programa abaixo?

```
1 #include <stdio.h>
2
3 int main(){
4    int v[5] = {3,5,4,7,1};
5    for(int i=1;i<5;i++){
6       printf("%d-", *(v+i)+*(v+i-1));
7    }
8 }
```

Resposta: _____

2. O que será impresso pelo programa abaixo?

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5    int i,*p;
6    p = malloc(sizeof(int)*3);
7    for(i=0;i<3;i++){
8     *(p+i) = i+1;
9    }
10   for(i=0;i<3;i++){
11      printf("%d", *(p+i)+1);
12   }
13 }
```

Resposta: _____

3. O que será impresso pelo programa abaixo?

```
1 #include <stdio.h>
2
3 int main(){
4    int v[5] = {3,5,4,7,1};
5    for(int i=1;i<5;i++){
6       printf("%d-", *(v+i)+*(v+i-1));
7    }
8 }
```

4. O programa abaixo deve armazenar inteiros em um vetor `v` cujo tamanho é informado pelo usuário na linha 7. Informe como devem ser preenchidas as lacunas nas linhas 8, 11 e 14 para que o vetor seja criado, para que os números informados pelo usuário sejam armazenados no vetor e para que so valores lidos sejam impressos na ordem em que foram lidos.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main(){
5    int *v, q, i;
6    printf("Informe␣o␣tamanho␣do␣vetor:␣");
7    scanf("%d",&q);
8    v = malloc(sizeof(int)*q);
9    for(int i=0;i<q;i++){
10      printf("Informe␣o␣%do␣valor:␣",i+1);
11      scanf("%d",&v[i]);
12   }
13   for(i=0;i<q;i++){
14      printf("%d\n", v[i]);
15   }
16 }
```

Respostas:
– Linha 8: _____
– Linha 11: _____
– Linha 14: _____

5. Qual será o retorno da função `f` abaixo se ela receber parâmetros `x=4`, `y=5` e `w=0`?

```
1 int f(int x, int y){
2    if(y==2)
3       return  x * x;
4    else
5       return  x * f(x,y-1);
6 }
```

Resposta: _____

6. O que será impresso pelo programa abaixo?

```
1 #include<stdio.h>
2
3 int f(int *v, int s){
4    if(s==1)
5       return *v;
6    else
7       return *v + f(v+1,s-1);
8 }
9
10 int main(){
11    int v[5] = {3,6,2,6,4};
12    printf("%d",f(v,5));
13 }
```

Resposta: _____

Para as questões..., considere a seguinte estrutura:

```
typedef struct lstItem{
   int dado;
   struct lstItem *next;
} listaItem;
```

7. Complete as lacunas das linhas 4 e 5 de modo que a função abaixo faça a inserção de um elemento no início da lista:

```
1 listaItem *ins_inicio(listaItem *lista, int dado){
2    listaItem *novo = malloc(sizeof(listaItem));
3    novo->dado = dado;
4    ------------------- ;
5    ------------------- ;
6    return lista;
7 }
```

Linha 4: _____
Linha 5: _____

8. Complete as lacunas das linhas 10 e 13 de modo que a função abaixo faça a inserção de um elemento no início da lista:

```
1 listaItem *ins_fim(listaItem *lista, int dado){
2    listaItem *novo = malloc(sizeof(listaItem));
3    novo->dado = dado;
4    novo->next = NULL;
5    if(lista==NULL){
6       lista = novo;
7    }
8    else{
9        listaItem *ultimo = lista;
10       while(__________________){
11        ultimo = ultimo->next;
12       }
13       ---------------------- ;
14   }
15   return lista;
16 }
```

Linha 10: _____
Linha 13: _____

9. Considere as descrições enumeradas a seguir e numere os itens a seguir.

(a) Estrutura de dados que pode ser percorrida em ambos os sentidos pois cada um de seus elementos aponta tanto para seu predecessor quanto para o sucessor.

(b) Pode-se acessar o primeiro elemento da lista diretamente a partir do último elemento, sem percorrer toda a lista.

(c) A partir de um determinado elemento da lista, só é possível acessar os seus sucessores pois os elementos não possuem apontamento para seus predecessores.

(d) Requer que os elementos sejam movidos para posições anteriores ou posteriores no caso de exclusões e inclusões de elementos da lista;

( ) Listas baseadas em vetores dinâmicos
( ) Listas encadeadas
( ) Listas duplamente encadeadas
( ) Listas circulares

10. Sobre listas encadeadas (ligadas), é correto afirmar que:

(a) Seus elementos podem ser acessados através do uso de índices.

(b) A remoção de elementos pode implicar em ocupação de espaço de memória desnecessário.

(c) O número máximo de elementos da lista deve ser definido quando o programa é escrito.

(d) Não é necessário movimentar elementos nas operações de inserção e remoção.

(e) Cada um dos elementos da lista ocupa menos espaço de memória do que ocuparia em uma lista implementada através de vetor.

11. Qual das seguintes expressões referenciam o valor do terceiro elemento (elemento de índice 2) do vetor `v`?

(a) `*(v + 2)` (c) `v + 4` (e) `v++;`
(b) `*(v + 4)` (d) `v + 2`

12. Quanto à alocação dinâmica de memória, considere as seguintes afirmações:

I A alocação dinâmica permite trabalhar com estruturas de dados cujo tamanho só será conhecido durante a execução do programa.

II A alocação dinâmica baseia-se na alocação de espaços contíguos de memória, cujo tamanho total é definido quando o programa é escrito.

III Alterações feitas em variáveis passadas por referência (na forma de ponteiros) para uma função não afetam o conteúdo dessas variáveis no programa principal.

São corretas as afirmações:

(a) I, II e III (c) II e III (e) II
(b) I e II (d) I

13. Considere as seguintes operações executadas em sequência e a seguir responda:

(a) `push(7)` (d) `push(2)` (g) `push(1)`
(b) `push(5)` (e) `pop()` (h) `pop`
(c) `pop()` (f) `push(7)` (i) `push(9)`

4.1 Quais serão os elementos contidos em uma *fila*, inicialmente vazia, ao final da sequência de operações acima? _____

4.2 Quais serão os elementos contidos em uma *pilha*, inicialmente vazia, ao final da sequência de operações acima? _____

14. Considere o problema de pesquisar por um número em um vetor ordenado utilizando o método de pesquisa binária. O menor número de comparações que nos permite concluir que um número não está presente em arrays de 10, 20, e 30 elementos é, respectivamente:

(a) 3, 4, 5
(b) 3, 3, 4
(c) 2, 3, 3
(d) 3, 4, 4
(e) 4, 4, 5

15. (1.5 ponto) Selecione a opçao cujas palavras preenchem corretamente as lacunas da informação a seguir:
O _____ é o método de ordenação que tem por fundamento selecionar um elemento em um vetor, chamado de *pivô* e, em sucessivas operações, posicionar todos os elementos _____ que o pivô à sua _____ e todos os elementos _____ que o pivô à sua _____.

(a) quicksort, menores, esquerda, maiores, direita

(b) mergesort, menores, esquerda, maiores, direita

(c) quicksort, menores, direita, maiores, esquerda

(d) mergesort, menores, direita, maiores, esquerda

(e) insertion sort, menores, direita, maiores, esquerda

16. **Filas e pilhas** Sejam `push(n)` e `pop()` as operações que, respectivamente, inserem um número inteiro n removem elementos em filas e pilhas. Considere a seguinte sequência de operações: `push(7), push(2), pop(), push(4), pop(), push(5), push(8), pop(), pop(), push(3)`.

- A soma dos elementos remanescentes em uma *fila* após essa sequência de operações será _____.
- A soma dos elementos remanescentes em uma *pilha* após essa sequência de operações será _____.

17. **Filas e pilhas** 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)`.

I Após a realização das operações, a soma dos elementos remanescentes na fila será maior que a soma dos elementos remanescentes na fila.

II 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.

III 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.

IV 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:

(a) I e II (c) I e IV (e) I, II, III e IV
(b) II e IV (d) II e III

18. **hash** Sobre tabelas hash, pode-se afirmar que

(a) Na resolução de colisões por encadeamento, a busca, no pior caso, terá complexidade semelhante à listas encadeadas

(b) 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.

(c) Em tabelas hash livres de colisões, a busca por um elemento tem, no pior caso, complexidade igual à complexidade da busca em árvores binárias de pesquisa.

(d) Em uma tabela livre 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 inserido ocupará a primeira posição da tabela, o segundo elemento ocupará o segundo e assim por diante.

19. **hash**. 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:

- $t = 4$: _____
- $t = 7$: _____
- $t = 12$: _____

20. **hash**. Considere uma tabela hash em que as chaves são *strings* de tamanho ilimitado formadas exclusivamente pelos caracteres a, b, c e d. Para cada um desses caracteres, atribui-se um valor inteiro da seguinte maneira: $a = 1$, $b = 2$, $c = 3$ e $d = 4$ e, assim, pode-se calcular um valor k para cada string, consistindo da soma dos valores atribuídos a cada um dos caracteres. Por exemplo, $k('abc') = 6$.

O cálculo do hash para uma string s é dado por $h(s) = k\%t$, onde t é o tamanho da tabela hash e k é um número inteiro obtido através da soma dos valores atribuídos a cada um dos caracteres da chave, send

21. **fila** Considere as seguintes estruturas, utilizadas para implementar uma fila:

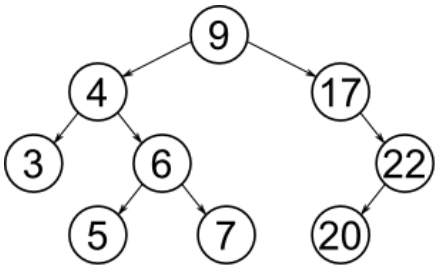
```
typedef struct{          typedef struct{
   int dado;             item *inicio;
   struct item *next;    item *fim;
}item;                   }fila;
```

Complete o código abaixo para implementar a função *pop* em uma fila.

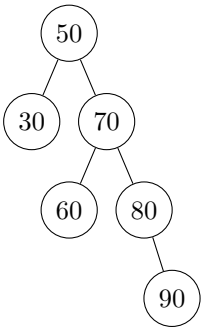
```
1 int pop(fila *f){
2    int dado = f->inicio->dado;
3    ____________________;
4    if(f->inicio==NULL)
5      ____________________;
6    return dado;
7 }
```

Linha 3: _____
Linha 5: _____

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



1. Escreva abaixo a sequência de números que será impressa caso a árvore seja percorrida
   - Em pré-ordem: _____
   - Em ordem: _____
   - Em pós-ordem: _____
2. Caso o nó armazena o número 9 seja excluído, a soma das folhas da árvore será _____.
3. Transforme a árvore abaixo em uma árvore AVL.



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