INSTITUTO FEDERAL
Goiano
Campus Rio Verde

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA GOIANO – CAMPUS RIO VERDE
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO
ESTRUTURAS DE DADOS I - 1ª AVALIAÇÃO

Data: 13/06/2022

Aluno (a):

VALOR TOTAL DA AVALIAÇÃO: 100 Peso: 0,4 NOTA:_____

QUESTÕES

Questão 1 - (total 10) Assinale a afirmativa **FALSA**, considerando a seguinte sequência de instruções escrita em linguagem C:

```
...
int *pti;
int i = 10;
pti = &i;
...
```

a. ( ) pti armazena o endereço de i.
b. ( ) a expressão i == *pti resulta em falso.
c. ( ) ao se alterar o valor de i, *pti será modificado.
d. ( ) *pti é igual a 10.
e. ( ) ao se executar *pti = 20; i passará a ter o valor 20.

Questão 2 - (total 10) Seja vet um vetor de 4 elementos: TIPO vet[4]. Supor que depois da declaração, vet esteja armazenado no endereço de memória 1024 (endereço de vet[0]). Supor também que o espaço de memória para armazenar uma variável do tipo char ocupa 1 byte, do tipo int ocupa 2 bytes, do tipo float ocupa 4 bytes e do tipo double ocupa 8 bytes. Escolha a opção que representa o resultado para a expressão vet+2, considerando que vet pudesse ter sido declarado como: char, int, float ou double, nessa ordem:

a. ( ) 1026, 1027, 1028, 1029
b. ( ) 1024, 1028, 1032, 1036
c. ( ) 1026, 1032, 1032, 1040
d. ( ) 1024, 1028, 1032, 1040
e. ( ) 1026, 1028, 1032, 1040

Questão 3 - (total 15) Considerando os conceitos de listas encadeadas dinamicamente, marque a opção que representa a sequência correta para as afirmações abaixo:

1) O acesso a qualquer elemento de uma Lista Linear Simplesmente Encadeada (LLSE) é realizado a partir da cabeça da lista, porém em uma Lista Linear Duplamente Encadeada (LLDE) é natural o acesso a qualquer elemento sem a necessidade de passar inicialmente pela cabeça da lista.

2) Considere um aplicação responsável por pesquisar um elemento em uma estrutura de lista encadeada dinamicamente. A implementação desta aplicação usando uma estrutura de LLDE obterá um desempenho superior a mesma aplicação implementada usando uma LLSE.

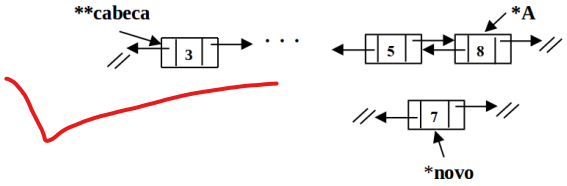
3) Comparando uma lista encadeada, contendo N elementos, implementada com LLSE e outra lista implementada com LLDE, é possível notar que a implementação com LLDE possui maior flexibilidade, uma vez que cada nó permite acesso ao nó anterior, porém a quantidade de espaço necessária em memória é maior.

4) O uso de um descritor apontando para o primeiro e último elemento da lista permite que uma implementação com LLDE tenha desempenho superior a uma implementação que use o mesmo descritor com LLSE, considerando a operação de remover o último elemento da lista.

V=Verdadeiro e F=Falso

a. ( ) 1-V, 2-F, 3-F, 4-V.
b. ( ) 1-F, 2-V, 3-V, 4-V.
c. ( ) 1-F, 2-V, 3-V, 4-F.
d. ( ) 1-F, 2-F, 3-V, 4-F.
e. ( ) 1-F, 2-F, 3-V, 4-V.

Questão 4 - (total 15) A figura ao lado representa de forma gráfica os nós de uma LLDE, contendo os campos ant, dado e prox (anterior, dado e próximo). Considere que o nó apontado pelo ponteiro "novo" deve ser inserido ANTES do elemento apontado por "A" (entre o nó que contém o número 5 e o nó que contém o número 8). Apresente o código, em linguagem C, responsável por fazer o encadeamento do nó novo.



Questão 5 - (total 30) Considere uma implementação de Lista Linear Simplesmente Encadeada (LLSE). Abaixo é possível ver a estrutura definida para a LLSE.

```
typedef struct no{
        int dado;
        struct no *prox;
}No;
```

a) (valor 15) Construa uma função responsável por receber duas listas em seu parâmetro e fazer a concatenação das duas listas, de forma que a lista1 fique com o resultado da concatenação e a lista2 fique vazia ao término da operação.
b) (valor 15) Construa uma função responsável por receber uma lista em seu parâmetro e identificar se um determinado valor está presente na lista. Se o valor estiver presente na lista a função deverá retornar verdadeiro (1), caso contrário deverá retornar falso (0).

Questão 6 - (total 20) Considere uma implementação de Lista Linear Duplamente Encadeada (LLDE). Considere também que a implementação utiliza um descritor com os campos de ini (ponteiro para o início da lista), fim (ponteiro para o último elemento da lista), qtd (armazena a quantidade de elementos presentes na lista) e maior (ponteiro para o nó com o maior elemento presente na lista). Abaixo é possível visualizar as estruturas usadas na implementação.

```
typedef struct no{
        int dado;
        struct no *ant, *prox;
}No;

typedef struct desc{
        int qtd;
        No *ini, *fim, *maior;
}Desc;
```

Desenvolva uma função responsável por remover um elemento no final da lista. A função deve receber em seu parâmetro o descritor.