1. Cite dois exemplos práticos de PILHA ao qual utilizamos no dia a dia na área computacional.

2. Em uma pilha necessitamos definir uma variável de ponteiro independente dos demais dados contidos no nó. Qual tipo deve ser o ponteiro e quanto ele consumirá de memória? Justifique sua resposta.

3. Quais as possíveis operações que podemos efetuar em uma pilha? Explique.

4. Em uma pilha, o que acontece se atribuirmos NULL para o ponteiro que aponta para o topo da pilha?

5. Analise e apresente as diferenças entre um vetor estático, um vetor dinâmico e uma lista simplesmente encadeada.

6. Desenvolva um programa denominado **PILHA1** em linguagem C que implemente as operações de uma pilha (pop, push e imprimir) cujo nó deverá conter nome[30] e idade.

7. Com base no exercício anterior (**PILHA1**), construa uma função que retorne a quantidade de elementos na pilha.

8. Ainda com base no programa **PILHA1**, faça a alteração onde o campo nome deverá ter tamanho dinâmico.

9. Construa um programa em linguagem C que implemente uma pilha através de 10 elementos gerados randomicamente (não repetidos). Posteriormente, apresente o conteúdo da pilha.

10. Dada a sequência de operações abaixo, indique o conteúdo final da pilha (do topo para a base), supondo que ela se inicie vazia: push('A'), push('B'), pop(), push('C'), push('D'), pop(), pop(), push('E').

11. Considere a pilha armazenada no vetor int S[5] = {3, 8, 2, 7, 1} com topo = 2 (ou seja, o elemento no índice 2 é o topo). Quais elementos ainda podem ser removidos por chamadas sucessivas de *pop* sem provocar *underflow*? Explique.

12. Como funciona as operações (inserção e remoção) em uma LISTA simplesmente encadeada?

13. Quais são as características de uma lista duplamente encadeada?

14. Utilizando a linguagem c, apresente a estutura mínima de um nó em uma lista simplesmente encadeada.

15. Qual a diferença básica entre uma lista simplesmente encadeada e uma lista duplamente encadeada? Explique.

16. Apresente em linguagem C, apenas a estrutura do nó de uma lista circular.

17. Analise a estrutura abaixo:

```
typedef struct no{
  int valor;
  struct no *proximo;
}No;

typedef struct{
  No *inicio;
  int tam;
}Lista;
```

Para a estrutura acima, de uma lista simplesmente encadeada, o desenvolvedor fez o seguinte algoritmo de busca sequencial:

```
1  No* buscar(Lista *lista, int num){
2    No *aux, *no = NULL;
3
4    aux = lista->inicio;
5    while (aux && aux- > valor != num)
6          aux = aux->proximo;
7    if (aux)
8        no = aux;
9    return no;
10
11 }
```

Analisando as estruturas acima, explique a lógica implementada na linha 5 da função **buscar**.

18. Analisando o código abaixo, apresente exatamente o resultado que será apresentado ao usuário após a execução completa do programa.

```c
#include <stdio.h>
#include <stdlib.h>
typedef struct no{
  int valor;
  struct no *proximo;
}No;
typedef struct{
  No *inicio;
}Lista;
//-----------------------------------------------
void inserirMeio(Lista *lista,int num,int ant){
 No *aux, *novo = malloc(sizeof(No));
 if(novo){
   novo->valor = num;
   if(lista->inicio == NULL){
     novo->proximo = NULL;
     lista->inicio = novo;
   }else{
     aux = lista->inicio;
     while(aux->valor != ant && aux->proximo)
       aux = aux->proximo;
     novo->proximo = aux->proximo;
     aux->proximo = novo;
   }
 }else printf("Erro ao locar memoria.\n");
}
```

```c
void imprimir(Lista lista){
 No *no = lista.inicio;
 while(no){
   printf("%d ", no->valor);
   no = no->proximo;
 }
}
//----------------------------
int main(void) {
 int opcao, valor, anterior;
 No *no;
 Lista lista;
 lista.inicio=NULL;
 inserirMeio(&lista, 10, 15);
 inserirMeio(&lista, 20, 0);
 inserirMeio(&lista, 30, 10);
 inserirMeio(&lista, 40, 20);
 inserirMeio(&lista, 50, 40);
 imprimir(lista);
 system("pause");
 return 0;
}
```

19. Utilizando linguagem C, apresente apenas uma estrutura de exemplo de um nó de uma lista circular.

20. Utilizando linguagem C, apresente apenas uma estrutura de exemplo de um nó de uma lista duplamente encadeada.

21. Apresente um exemplo do dia a dia na área computacional, do uso de LISTAS.

22. Qual a principal diferença entre uma lista simplesmente encadeada e uma lista duplamente encadeada? Explique

23. considere uma lista simplesmente encadeada que contém os seguintes elementos: 10 → 20 → 30 → 40 → 50.

Indique a sequência da lista após a lista receber NULL e a remoção do elemento 30, justificando sua resposta.

24. A respeito de vazamento de memória, analise o código abaixo e desenvolva uma função denominada **sair()** que seja responsável por liberar corretamente toda a memória alocada dinamicamente antes da finalização do programa.

```c
#include <stdlib.h>
#include <stdio.h>

typedef struct apelido_no{
   int dado;
   struct apelido_no *proximo;
}no;
no *top=NULL;
//-------------------------------------
int entrada_dados(){
   int valor;
   printf("\nValor a empilhar: ");
   scanf("%d",&valor);
   return valor;
}
//-------------------------------------
void push(int item){
   no *novo=malloc(sizeof(no));
   //verificar se há memoria
   novo->dado=item;
   novo->proximo=top;
   top=novo;
   system("pause");
}
//-------------------------------------
void pop(){
   if (top==NULL)
      printf("A pilha esta vazia\n");
   else{
     no *temp;
     temp=top;
     top=top->proximo;
     free(temp);
   }
   system("pause");
}
//-------------------------------------
int main(){
   int n,opcao;
   do{
     system("cls");
     printf("\nMenu\n1. Empilha");
     printf("\n2. Desempilha\n3. Sair")
     printf("\nOpcao (0-3):");
     scanf("%d",&opcao);
     switch (opcao){
       case 1:
         n=entrada_dados();
         push(n); //empilhar
         break;
       case 2:
         pop(); //desempilhar
         break;
       case 3:
         //sair();
         break;
     }
   }while (opcao!=3);
   system("pause");
   return 0;}
```

25. Apresente as características sobre de Listas Circulares, explicando o seu funcionamento.

26. Apresente três desenhos que representem graficamente:

a. Lista simplesmente encadeada
b. Lista duplamente encadeada
c. Lista circular

27. Quais são as regras para inserção e remoção de elementos (nós) em uma lista?

28. Analise o código abaixo desenvolvido em linguagem C e apresente a representação gráfica (desenho da estrutura) após a execução completa do programa, inclusive sinalizando onde a variável ***p** estará apontando, bem como a que estrutura se refere (pilha, fila, lista simplesmente encadeada, lista duplamente encadeada ou lista circular).

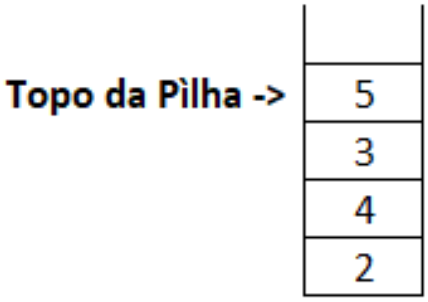
```c
#include <stdlib.h>

typedef struct no{
  int dado;
  struct no *proximo;
}NO;

NO *p=NULL;
//------------------------------
void acao_1(int v){
  NO *novo=malloc(sizeof(NO));
  novo->dado=v;
  novo->proximo=p;
  p=novo;
}
//------------------------------
void acao_2(){
  if (p==NULL)
     printf("      vazia\n");
  else{
    NO *temp;
    temp=p;
    p=p->proximo;
    free(temp);
  }
}
```

```c
//------------------------------
int main(){
  int n,opcao;
  system("cls");
  acao_1(71);
  acao_1(8);
  acao_1(15);
  acao_2();
  acao_1(19);
  system("pause");
  return 0;
}
```

29. Como podemos determinar o tamanho de uma lista? Explique.

30. Crie uma função que verifica se um número é par ou ímpar. Em seguida, percorra a lista e, para cada nó, verifique se o valor é par ou ímpar. Caso seja par, insere em uma lista de números pares, caso contrário (ímpar), insere em uma lista de números ímpares.

31. Considere a seguinte estrutura de dados do tipo Pilha, na qual existem quatro valores armazenados e cujo topo é indicado pelo ponteiro Topo da pilha.



A sequência de instruções expressas a seguir na forma de um pseudocódigo deve ser executada com base no estado atual da pilha.

As instruções PUSH e POP são instruções típicas de estruturas de dados do tipo Pilha, e representam, respectivamente, inserção e exclusão.

1. Soma ← 0;
2. POP(x);
3. Soma ← Soma + x;
4. x ← 10;
5. PUSH(x);
6. x ← 12;
7. PUSH(x);
8. POP(x);
9. POP(x);
10. Soma ← Soma + x;

Com base nessa sequência de instruções, apresente o valor final da variável soma.