1. Estrutura de pilha se basea em uma struct onde tem somente um array e o topo. Quando estamos falando da função push se basea em dar topo++ e colocar o valor dentro do array e quando damos pop basicamente você da topo— e segue a vida não precisando desvincular o valor, pois ele sera sobescrito quando ocorrer outro push.
2. Listra Simplesmente encadeada: Embora sua estrutura seja mais simples e ocupe menos memoria comparada a duplamente, ela basicamente possui um valor e um ponteiro para nó(ou seja a própria estrutura) apontando para a próxima. Ela tem um problema onde se ela não for circular não podemos pegar o valor no meio da fila para conferir se ele está contido na lista, obrigando a apenas começar a verificação pelo primeiro termo.   
   Listra Duplamente encadeada: Já na duplamente ela pode voltar e além de possuir uma memoria que demanda mais um ponteiro para guardar a volta, permitindo percorrer a lista de onde vc quiser, e pode ser mais eficiente em questões de busca como comentado.
3. A lista encadeada comparada ao array é o maior beneficiário da alocação dinâmica permitindo que ela possua um quantidade de termos variável em quanto o array a quantidade tem que ser declarada na sua inicialização.
4. Quando criamos um ponteiro para struct nos podemos usar o -> para se referenciar ao valor presente na memoria da struct nesse atributo que está guardado na mesma, também podemos fazer assim (\*struct).atributo funciona também. A principal importância de aprender ponteiros, além de podem manipular e retornar 2 coisas em uma função é poder ter eficiência de memoria e de agilidade na hora de codar permitindo a diminuição de uso de variáveis secundarias. NÃO LIGANDO PARA ESCOPO
5. O ponteiro em c é quando vc fala que uma variável está se referencando a um local na memoria seja na heap ou stack e lá possui um valor:  
   int \*x= &Y

Ou seja o x guarda o endereço de y e \*x guarda o valor que está presente em y podendo, fazer referencia ao valor e modifica-lo mesmo fora da função de origem. Um caso comum de ponteiros é quando você deseja retornar mais deu uma coisa na função.

1. Uma struct em C similar a um objeto onde vc permite que tipos diferentes de dados possam andar juntos e serem criados e referenciados junto criando, assim um novo tipo de dado em C. A principal importância é que eles não se perdem, por estarem unidos em uma estrutura so além de ser mais fácil de manipular vc pode fazer uma instanciação de ambos de uma vez só mantendo assim como dito novamente unidas. Um exemplo seria Alunos ou Livros onde ela pode ter matricula um vetor de char e no caso de livros um autor.
2. A função malloc serve para fazer uma alocamento dinâmico de um ponteiro para heap onde lá guarda o valor desejado:  
   int \*i = (\*int) malloc(sizeof (int) \* tamanho) Além disso permite que quando a variável não tem mais valor no contexto do código ela pode ter sua memoria desalocada no caso a função free. Quando fazemos a alocação temos que verificar se o ponteiro não apontou para um lixo na memoria fazendo, assim que tenha um mal funcionamento do ponteiro por isso deve ser feito uma verificação.