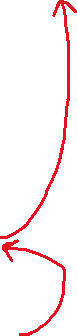
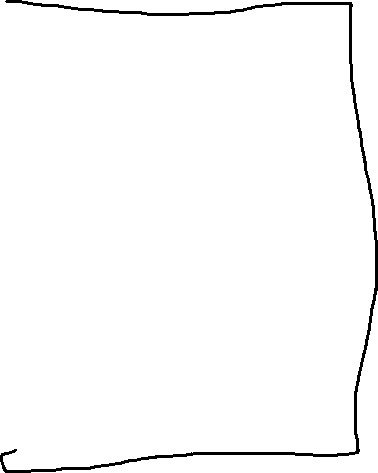
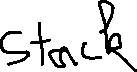
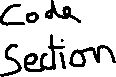
**Section 2: Essential C and C++ Concepts**

**3. Array Basics:**

* Collection of similar data.

**7. Pointers:**

* Pointer is an address variable used to store address of a data.
* Memory is divided as follows:



* Pointers are used for:
  + Accessing heap
  + Accessing resources
  + Parameter passing
* Declare pointer:   
  int\* pointerToAnInteger;  
  pointerToAnInteger = &anIntegerVariable; // & - Address of  
  cout << \*pointerToAnInteger; // \* - Dereference operator
* To get memory in heap: pointer = new int[5];
* All pointers regardless of what datatype the point to, take up the same space (usually 8 bytes).

**9. Reference in C++**

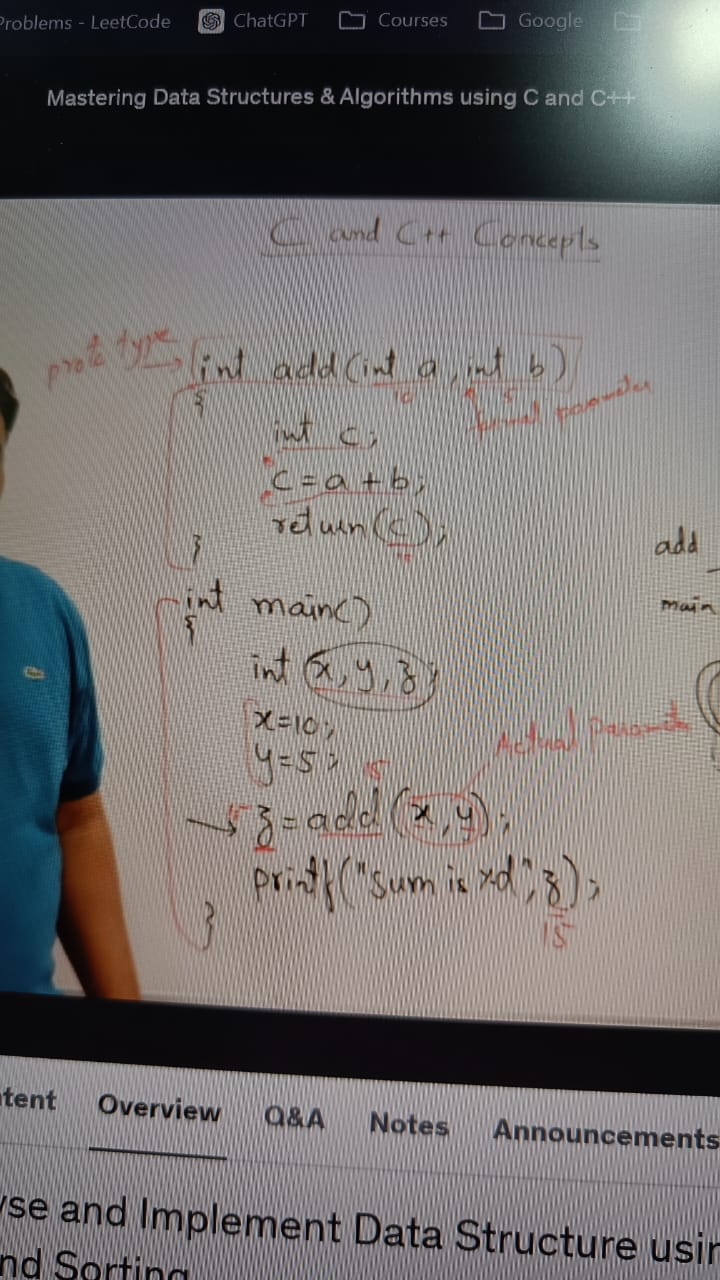
* Alias given to a variable.
* How?  
  int a = 10;  
  int &r = a;

Now “r” can be used in place of a.

* Useful in parameter passing.

**13. Functions:**

* Function is a piece of code (related instructions) that performs a specific task.
* Used for modular/procedural programming.



**15. Parameter Passing Methods**

#include <bits/stdc++.h>

using namespace std;

void swap\_byValue(int a, int b) {

    int temporary = a;

    a = b;

    b = temporary;

}

void swap\_byAddress(int\* a, int \*b) {

    int temporary = \*a;

    \*a = \*b;

    \*b = temporary;

}

void swap\_byReference(int& a, int& b) {

    int temporary = a;

    a = b;

    b = temporary;

}

int main() {

    int a = 1, b = 2;

    swap\_byValue(a, b); //Doesn't affect a and b

    cout << a << " " << b << endl;

    swap\_byAddress(&a, &b); //Affects a and b as addresses are passed; Values at addresses are swapped.

    cout << a << " " << b << endl;

    swap\_byReference(a, b); // Affects a and b as their references are passed

    cout << a << " " << b << endl;

}