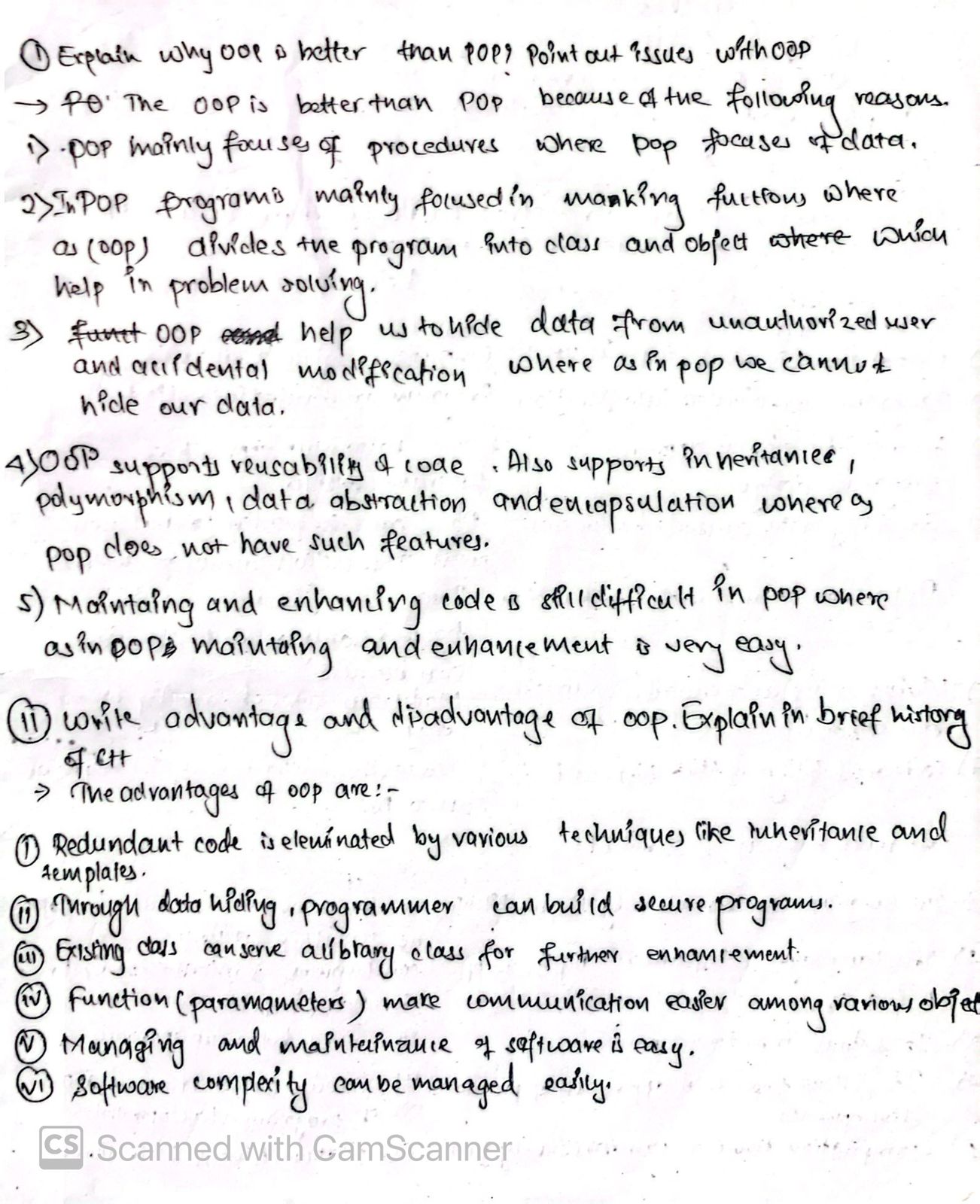
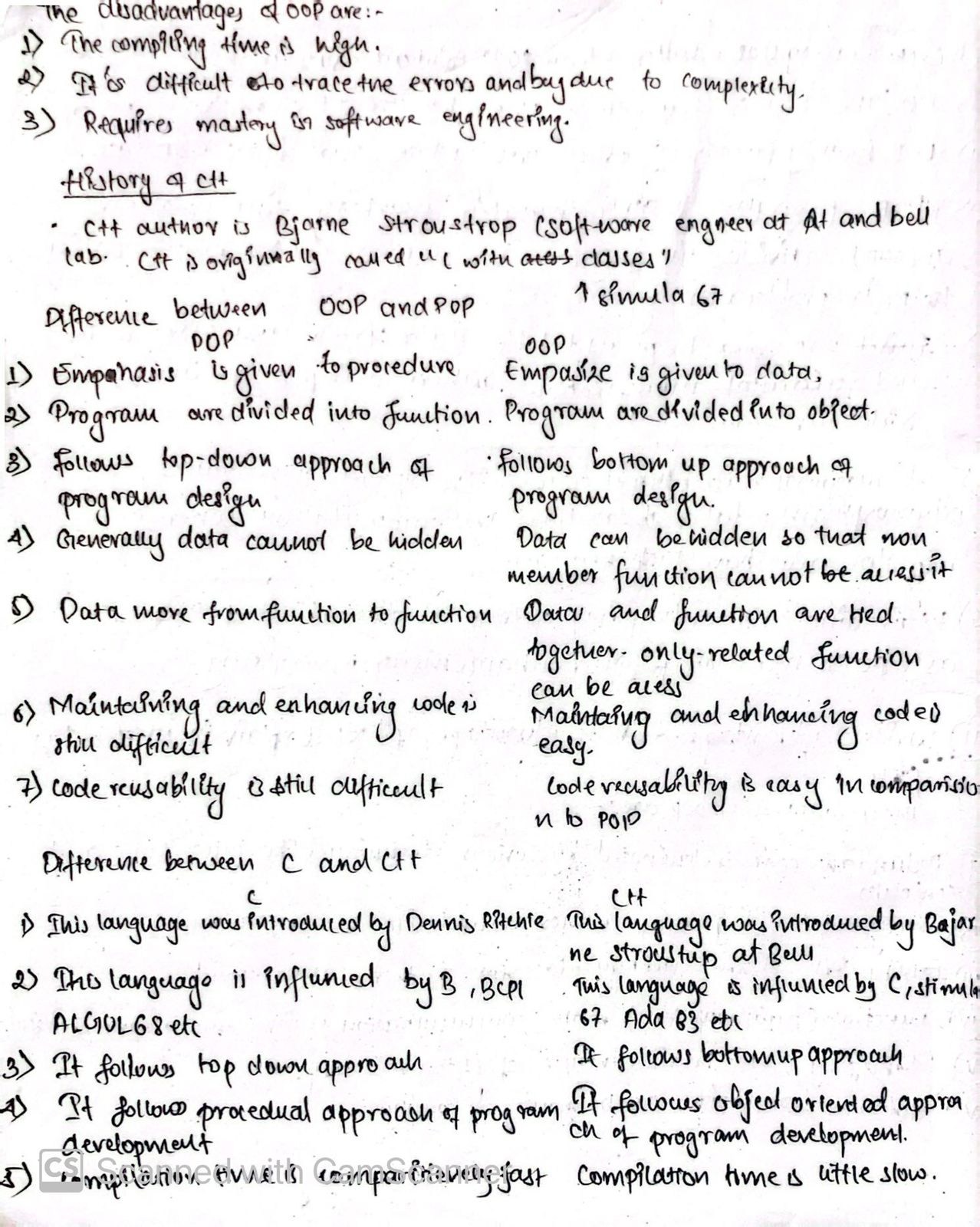
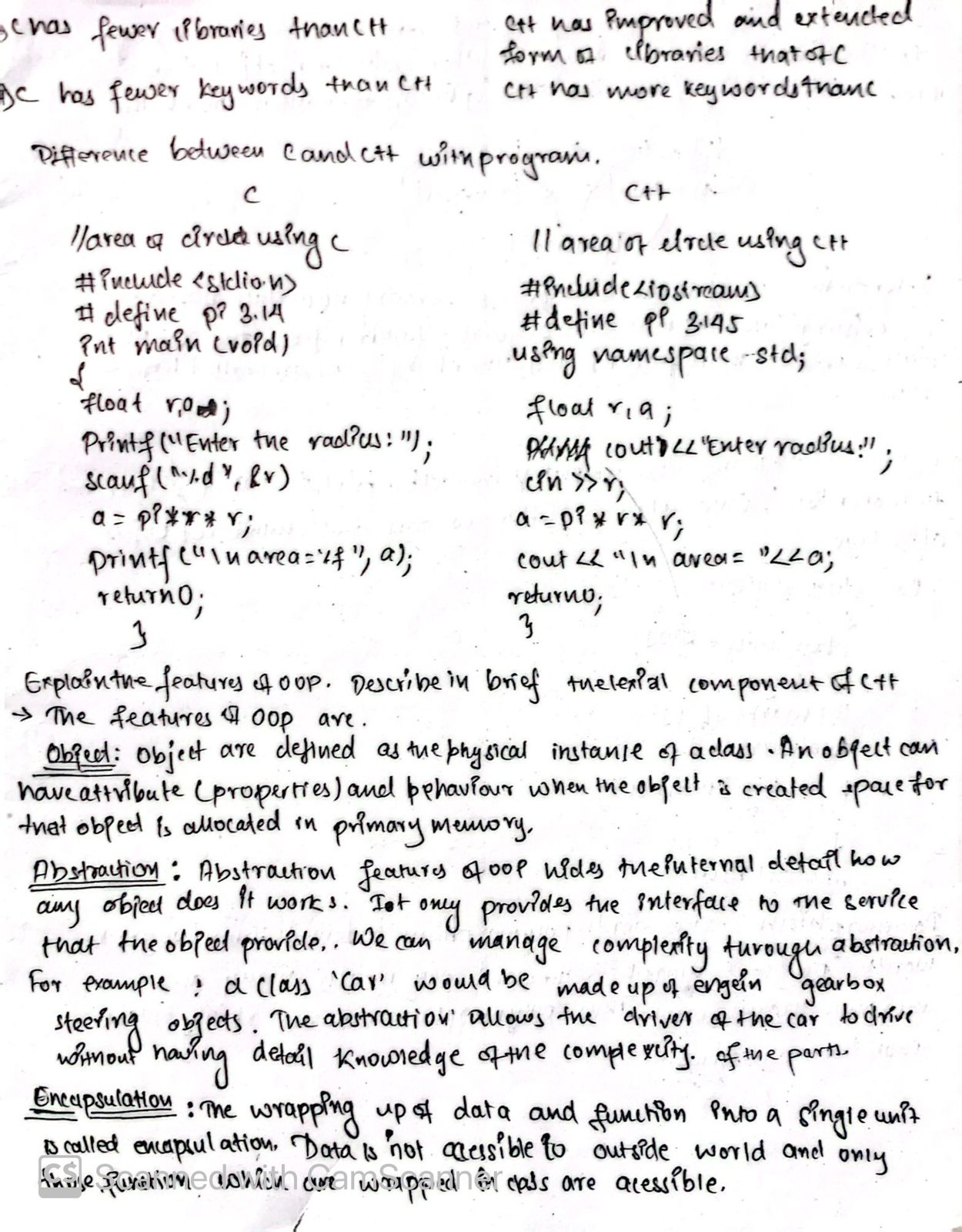
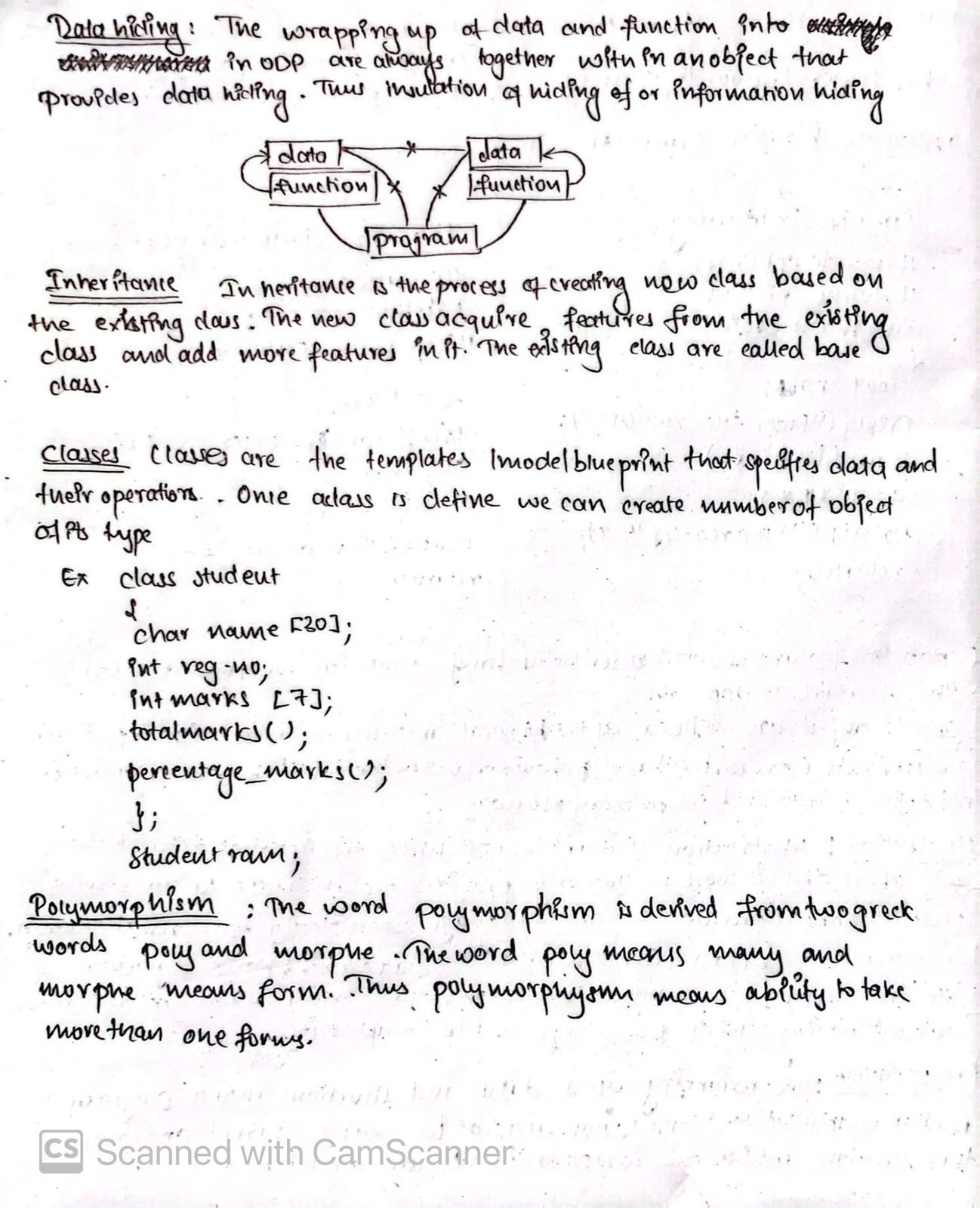
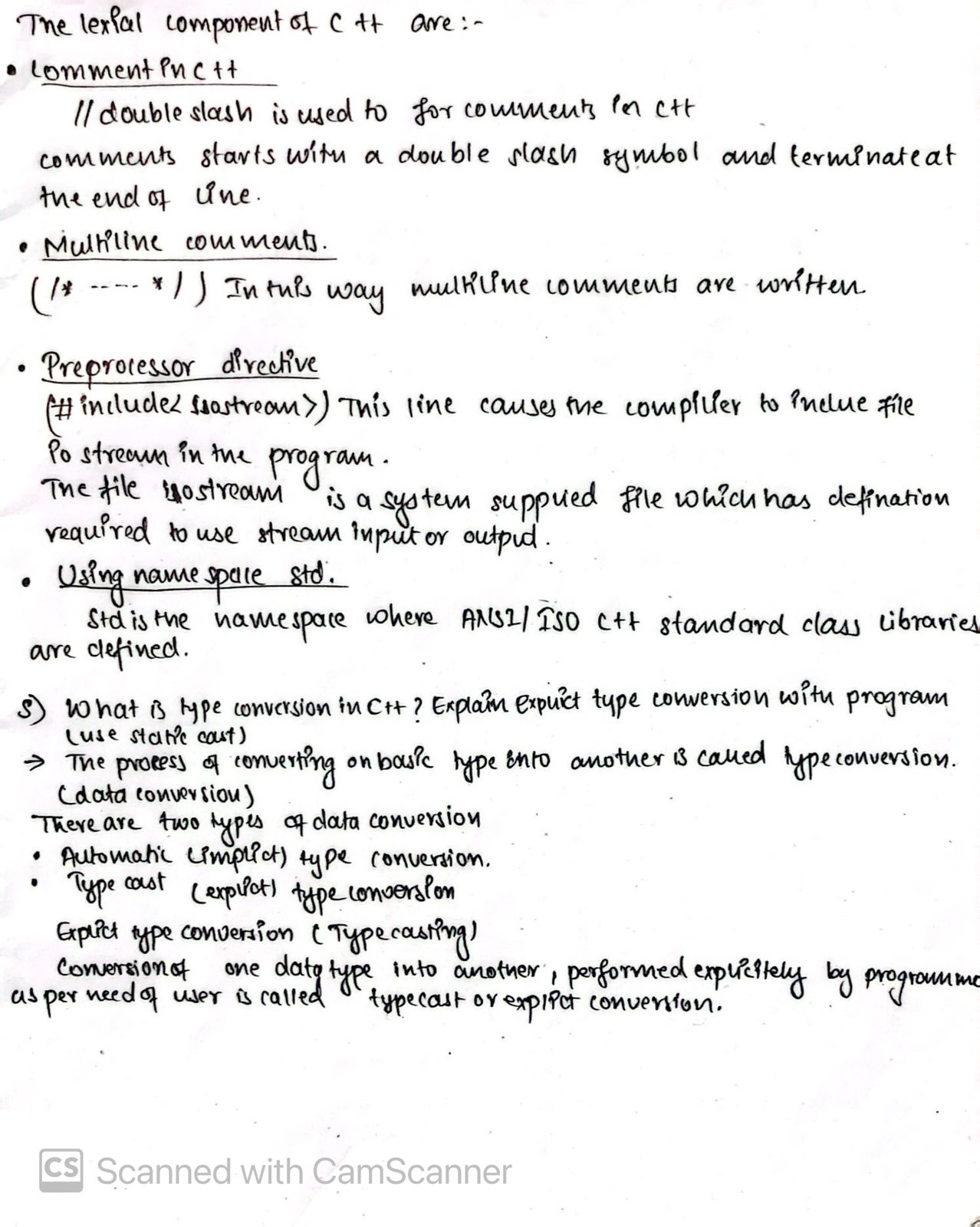
****

****

****

****

****

**6)What is namespace? Why do we need this? Write a program to demonstrate namespace using “using” keyword and scope resolution operator.**

Namespace is the mechanism is used for the logical grouping of variables, classes and function in c++.We need namespace as it is a container for variable , function ,classes and other identifiers that avoids conflicts residing in different scopes .

Syntax:

Namespace namespace\_name

{

// declaration of variables, classes, function etc

}

A program to demonstrate namespace using “using” keyword and scope resolution operator.

**#include<iostream>**

**using namespace std;**

**namespace first**

**{**

**float f=5.5;**

**int i=100;**

**}**

**int main ()**

**{**

**using first::f ;**

**using first::i ;**

**cout<< “f=\t” <<f<< “\n”;**

**cout<< “i=\t” <<i<< “\n”;**

**return 0;**

**}**

**7)How do we define user defined constant? Show by program.**

We can use user defined constants using the keyword ‘const’. This indicates that the value does not change during

program execution .Since we cannot change the value of the constant we must initialized it.

examples ;

const int max =1000;

const int arr[]={1,2,3,4};

using in program:

**#include<iostream>**

**using namespace std;**

**int main()**

**{**

**const int g=10;**

**int u,output;**

**cout<<"enter the value of u:"<<endl;**

**cin>>u;**

**output=g\*u;**

**cout<<"the output is :"<<endl<<output;**

**return 0;**

**}**

**8) What are manipulators? How can we use them? Show by program.**

Manipulator are special variables and function used to modify as user‘s requirements .Some of the commonly used manipulator are endl, setw(), setfill() , setprecession()etc.

We can use them by writing a header <iomanip.h> (standard masnipulator).

**#include<iostream>**

**#include<iomanip>**

**using namespace std;**

**int main()**

**{**

**long n=1234567890;**

**int u=12345;**

**float m=50.123;**

**cout<<setw(10)<<n<<endl; //setw() manipulator**

**cout<<setw(10)<<setfill('\*')<<u<<endl; //setfill() manipulator**

**cout<<"num="<<setprecision(3)<<m<<endl; //setprecision() manipulator**

**return 0;**

**}**

**9) What is dynamic memory allocation (DMA)? How can we implement this? Write program to display DMA.**

The process of allocating and de allocating memory at run time is known as dynamic memory allocation. C++ provide “new” for dynamic memory allocation and “delete” for dynamic memory de allocation.

The “new” operator obtains memory during memory allocation from the memory heap of the operating system and return address of obtained memory. The memory reserved by “new” in any scope remains as it is even after the program control goes out of scope.

**void function()**

**{ int\* ptr = new int; // heap-allocated integer**

**// ...**

**delete ptr; // deallocate the memory }**

Additionally, if you allocate an array of objects on the heap, you should use delete[] to deallocate it properly:

**void function()**

**{ int\* arr = new int[10];// heap-allocated array of integers**

**// ...**

**delete[] arr; // deallocate the array**

**}**

**10) What is function overloading? Overload a function with :**

**a)different number of arguments**

**b)different types of arguments**

Some function conceptually perform the same task on object of different types of number .In such case it is convenient to give them the same name . When the same name is used for different operation then It is called function overloading .When an overloaded function is called, the function with matching argument is evoked.

example

void display ();

void display (int);

void display (float);

void display (int,float);

**Program with different number of argument**

**#include<iostream>**

**using namespace std;**

**void display (int);**

**void display (int,int);**

**int main()**

**{**

**int a=5,b=6;**

**cout<<"the argument function:";**

**display(a);**

**cout<<"two argument function:";**

**display(a,b);**

**return 0;**

**}**

**void display(int a)**

**{**

**cout<<a<<endl;**

**}**

**void display (int a,int b)**

**{**

**cout<<a<<"and"<<b<<endl;**

**}**

**Program with different types of argument**

**#include<iostream>**

**using namespace std;**

**void display (int);**

**void display(char);**

**int main()**

**{**

**char ch='a';**

**int num=20;**

**cout<<"char function";**

**display(ch);**

**cout<<"int num:";**

**display(int num);**

**return 0;**

**}**

**11)What is inline function? WAP to implement inline function. What is the advantage of inline function? Where we cannot use inline function?**

Inline function is the function is used to save memory space and reduce size of the code.

Use of function adds extra overheads .The calling of a function require activities like jumping to the function definition, saving registers, pushing, argument into the stack and returning result to the calling function.

**Program of in line function:**

**#include<iostream>**

**using namespace std;**

**inline float intrest (float p, float t, float r)**

**{**

**return((p\*t\*r)/100);**

**}**

**int main()**

**{**

**float result ;**

**result=intrest(1000,5,10);**

**cout<< "the intrest is :"<<endl<<result;**

**return 0;**

**}**

The advantages of inline function:

1. Increases performance in program.
2. It does not require function calling overhead.
3. It saves overhead of variables push on the stack, while function calling.
4. It saves overhead of return of result from called function to calling function.

There are some situation where inline function cannot be used:

1. for large function.
2. for function returning values ,if a loop ,a switch or a goto exists.
3. if inline function are recursive.

**12) What is default argument? What is the rule to define default arguments? Show with program.**

In **c++** there is a provision of supplying less number of argument than the actual number of parameter.

This mechanism is supported by default arguments. So with the help of default argument a function without specifying all its arguments. In such cases the function assigns a default value to the parameter which does not have a matching argument in the function call.

Default values are specified when the function is declared. The compiler looks at the prototype to see how many arguments a function uses and alert the program for possible values .

**Program:**

**#include<iostream>**

**using namespace std;**

**void marks(int m1=40,int m2=40 ,int m3 =40);**

**int main()**

**{**

**marks();**

**marks(55);**

**marks(66,77);**

**marks(77,85,92);**

**return 0;**

**}**

**void marks(int m1,int m2,int m3)**

**{**

**cout<<"total:"<<endl<<(m1+m2+m2);**

**}**

**13)What is pass by value and pass by reference in function call? Show in program.**

**Call by value**

When values of actual argument are passed to a function as argument of each arguments it is known as function call by value. In this call, the value of each actual argument of the function definition. The content of the arguments in the calling function are not alternated even if they are changed in the called function.

**#include<iostream>**

**using namespace std;**

**void swap(int , int);**

**int main()**

**{**

**int a,b;**

**a=99;b=89;**

**cout<<"before function call "<<a<<b;**

**swap(a,b);**

**cout<<"after function call"<<a<<b;**

**return 0;**

**}**

**void swap(int x,int y)**

**{**

**int temp;**

**temp=x;**

**x=y;**

**y=temp;**

**cout<<"the values with in function :"<<x<<y;**

**}**

**Call by reference**

In this type of function call the address of variable on argument is pass to the function as argument instead of

Actual value of variable .So the variable is passed as argument during the function call are changed by the called function.

**#include<iostream>**

**using namespace std;**

**void swap(int&,int&)**

**int main()**

**{**

**int a=5, int b=6;**

**cout<<"before swapping :"<<a<<"and b"<<b<<endl;**

**swap a,b;**

**cout<<"after swapping "<<a<<"and b"<<b<<endl;**

**return 0;**

**}**

**void swap(int&x,int&y)**

**{**

**int temp;**

**temp=x;**

**x=y;**

**y=temp;**

**}**

**14) What is reference variable? Show return by reference by the help of example.**

A reference variable is another name for a variable. It is declared using & operator.

Syntax:

datatype & reference\_variable =orginal value;

example:

**#include<iostream>**

**using namespace std;**

**void main()**

**{**

**int x =100;**

**int & ref=x;**

**cout<< “x=”<<x\t ref=”<<ref;**

**}**

1. **Differentiate between:**
2. **C structure and C++ structure.**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **C structure** | **C++ structure** | |
| **Syntax** | Defined using the struct keyword. | Defined using the struct keyword (same as C), but can use the class keyword for more advanced features. | |
| **Members:** | All members are public by default. | All members are public by default, but can use private and protected access specifiers. | |
| **Member Functions:** | | Cannot have member functions | | Can have member functions. | |
| **Inheritance:** | | Does not support inheritance. | | Supports inheritance. | |
| **Access Modifiers:** | | No support for access control; everything is public. | | Can use access specifiers (public, private, protected). | |

1. **C++ structure and class.**

|  |  |  |
| --- | --- | --- |
|  | **C++ Structure** | **C++ Class** |
| **Default Access:** | Members are public by default. | Members are private by default. |
| **Purpose** | Typically used for simple data structures. | Typically used for more complex data types and encapsulation |
| **Inheritance** | Can be used in inheritance, but by default, inheritance is public. | Can be used in inheritance, and by default, inheritance is private. |

1. **Macros and inline functions.**

|  |  |  |
| --- | --- | --- |
|  | **Macros:** | **Inline Functions:** |
| **Definition** | Preprocessor directives defined using #define. | Functions defined using the inline keyword. |
| **Compilation:** | Replaced by the preprocessor before compilation. | Integrated into the code during compilation. |
| **Type Safety:** | No type checking or scope control. | Type checking is performed. |
| **Debugging** | Harder to debug due to text substitution. | Easier to debug as they are regular functions. |
| **Usage:** | Typically used for constants or simple code snippets. | Used for small functions to avoid the overhead of function calls. |

1. **C++ structure and union.**

|  |  |  |
| --- | --- | --- |
|  | **C++ Structure:** | **C++ Union:** |
| **Definition:** | A data type that can hold multiple members, where each member has its own memory location. | A data type that can hold multiple members, but only one member can occupy the memory at any one time. |
| **Memory Allocation:** | Each member has its own memory space, so the total memory used is the sum of all members' sizes. | Only enough memory to hold the largest member is allocated. |
| **Access:** | All members can be accessed simultaneously. | Only one member can be accessed at a time, and updating one member may affect the others. |

**16) What are enumerations (enum) in C++?**

Enumerations (enum) in C++ are a user-defined data type that consists of a set of named integral constants.

**#include<iostream>**

**using namespace std;**

**enum Day { Sunday, Monday, Tuesday, Wednesday, Friday, Saturday };**

**int main()**

**{**

**Day today = Monday;**

**cout << "The day is: " << today << endl;**

**return 0;**

**}**

**17)WAP to calculate simple interest by using concept of default argument with default value of rate = 15%.**

**#include <iostream>**

**using namespace std;**

**double SI(float p,float t, float r = 15.0)**

**{**

**return (p \* t \* r) / 100;**

**}**

**int main() { float p, t;**

**cout << "Enter the principal amount: ";**

**cin >> p;**

**cout << "Enter the time (in years): ";**

**cin >> t;**

**float i = SI(p, t);**

**cout << "Simple Interest with defaultrate(15%): " << i << endl;**

**return 0;**

**}**

**18) WAP to calculate area of circle, rectangle, and triangle using concept of function overloading.**

**#include <iostream>**

**#define PI 3.14**

**using namespace std;**

**float area(int r)**

**{**

**return (PI \* r \* r);**

**}**

**float area(int l, int w)**

**{**

**return (l \* w);**

**}**

**float area(float b, float h)**

**{**

**return 0.5 \* b \* h;**

**}**

**int main() {**

**float radius, length, width, base, height;**

**cout << "Enter the radius of the circle: ";**

**cin >> radius;**

**cout << "Area of the circle: " << area(radius) << endl;**

**cout << "Enter the length and width of the rectangle: ";**

**cin >> length >> width;**

**cout << "Area of the rectangle: " << area(length, width) << endl;**

**cout << "Enter the base and height of the triangle: ";**

**cin >> base >> height;**

**cout << "Area of the triangle: " << area(base, height) << endl;**

**return 0;**

**}**

**19) WAP to calculate the volume of cube, cuboid, and cylinder using the concept of function overloading.**

**#include <iostream>**

**#define PI 3.14**

**using namespace std;**

**float volume(float side)**

**{**

**return (side \* side \* side);**

**}**

**float volume(float length, float width, float height)**

**{**

**return( length \* width \* height);**

**}**

**float volume(float radius, float height)**

**{**

**return (PI \* radius \* radius \* height);**

**}**

**int main()**

**{**

**float side, length, width, height, radius;**

**cout << "Enter the side length of the cube: ";**

**cin >> side;**

**cout << "Volume of the cube: " << volume(side) << endl;**

**cout << "Enter the length, width, and height of the cuboid: ";**

**cin >> length >> width >> height;**

**cout << "Volume of the cuboid: " << volume(length, width, height) << endl;**

**cout << "Enter the radius and height of the cylinder: ";**

**cin >> radius >> height;**

**cout << "Volume of the cylinder: " << volume(radius, height) << endl;**

**return 0;**

**}**

**20) WAP to display n number of character by using default argument for both parameters. Assume that the function takes two arguments one character to be printed and other number of character to be printed.**

**#include <iostream>**

**using namespace std;**

**void display(char character = 'N', int count = 5)**

**{**

**for (int i = 0; i < count; ++i)**

**{**

**cout << character;**

**}**

**cout << endl;**

**}**

**int main()**

**{**

**display();**

**display('x');**

**display('\*', 10);**

**return 0;**

**}**