Q.1 STRUCTURE WITHOUT MEMBER FUNCTON; INPUT :

```
#include <iostream>

using namespace std;

int num;

void Set(int temp) { num = temp; }

void display() { cout << "num=" << num; }

};

int main()

{

marks M1; m1.Set(9); m1.display();

}
```

OUTPUT :

```
num=9
```

Q.2 STRUCTRE WITH MEMBER FUNCTON :

INPUT :

```
#include <iostream> using namespace std; struct Person {

char name[50];

int age;

float salary;

};

void displayData(Person);

int main(){

Person p;

cout << "Enter Full name: ";

cin.get(p.name, 50);

cout << "Enter age: ";

cin >> p.age; cout << "Enter salary: ";

cin >> p.salary;

displayData(p);

return 0;

}
```

```
void displayData(Person p) {

cout << "\nDisplaying Information."<<end1;

cout << "Name: " << p.name << endl;

cout <<"Age: " << p.age << endl;

cout << "Salary: " << p.salary;

}
```

OUTPUT :

```
Enter Full name: Bill Jobs

Enter age: 55

Enter salary: 34233.4

Displaying Information.

Name: Bill Jobs

Age: 55

Salary: 34233.4
```

Q.3 CREATE A STUDENT CLASS,READ AND PRINTN STUDENT'S DETAILS :

INPUT :

```
#include <iostream> using namespace std; #define MAX 10

class student

{

private:

char name[30]; int rollNo;

int total;

float perc; public:

void getDetails(void); void putDetails(void);

};

void student::getDetails(void){

cout << "Enter name: " ;

cin >> name;

cout << "Enter roll number: ";

cin >> rollNo;

cout << "Enter total marks outof 500: "; cin >> total;
perc=(float)total/500*100;

}

void student::putDetails(void){

cout << "Student details:\n";

cout << "Name:"<< name << ",Roll Num<< ",Total<< total<< ",Percentage:" << perc;

} int main()

{ student std[MAX]; //array of objects creation int n,loop; cout << "Enter total number of students: "; cin >> n; for(loop=0;loop<
n; loop++){ cout << "Enter details of student " << loop+1 << ":\n"; std[loop].getDetails();

}

cout << endl; for(loop=0;loop< n; loop++){ cout << "Details of student " << (loop+1) << ":\n"; std[loop].putDetails();

}

return 0;

}
```

OUTPUT :

```
Enter total number of students: 2

Enter details of student 1:

Enter name: Mike

Enter roll number: 101 Enter total marks outof 500: 456

Enter details of student 2:

Enter name: Mock

Enter roll number: 102 Enter total marks outof 500: 398

Details of student 1:

Student details:

Name:Mike,Roll Number:101,Total:456,Percentage:91.2Details of student 2:

Student details:

Name:Mock,Roll Number:102,Total:398,Percentage:79.6
```

Q.4 USE OF SCOPE RESOLUTION OPERATOR :

INPUT:

```
#include<iostream>

using namespace std;

int num = 30; // Initializing a global variable num int main()

{
```

int num = 10; // Initializing the local variable num cout << "Value of global num is " << ::num; cout << "nValue of local num is " << num; return 0;

}

Output :

Moving on with this article on Scope Resolution Operator In C++

Q.5 SWAP THE NO. BY POINTER AND REFERENCE :

INPUT :

# include <iostream> using namespace std; void swap(int& a, int& b)

{ int c=a; a=b; b=c;

}

int main(void)

{ int i=5,j=7;

cout<<"Before swap"<<endl; cout<<"I:"<<i<<"J:"<<j<<endl; swap(i,j); cout<<"After swap"<<endl; cout<<"I:"<<i<<"J:"<<j<<endl; return 0;

}

Q.6 USE FUNCTION AS A VALUE USING REFERENCE VARIABLE :

INPUT :

#include <iostream> using namespace std; void swap(int &x, int &y); int main () {

int a = 100; int b = 200; cout << "Before swap, value of a :" << a << endl; cout << "Before swap, value of b :" << b << endl; swap(a, b); cout << "After swap, value of a :" << a << endl; cout << "After swap, value of b :" << b << endl; return 0;

}

OUTPUT :

Before swap, value of a :100

Before swap, value of b :200

After swap, value of a :200

After swap, value of b :100

Q.7 EXAMPLE OF A NEW AND DELETE OPERATOR :

INPUT :

#include <iostream> using namespace std; int main ()

{ int* p = NULL; p = new(nothrow) int;

if (!p)

cout << "allocation of memory failed\n"; else

```
{

*p = 29; cout << "Value of p: " << *p << endl;

}

float *r = new float(75.25); cout << "Value of r: " << *r << endl; int n = 5;

int *q = new(nothrow) int[n];

if (!q)

cout << "allocation of memory failed\n"; else

{

for (int i = 0; i < n; i++) q[i] = i+1;

cout << "Value store in block of memory: "; for (int i = 0; i < n; i++) cout << q[i] << " ";

}

delete p; delete r; delete[] q; return 0;

}
```

Output:

```
Value of p: 29

Value of r: 75.25

Value store in block of memory: 1 2 3 4 5
```

Q.8 INLINE FUNCTION :

INPUT :

```
#include <iostream> using namespace std; inline int cube(int s)

{ return s*s*s;

}

int main()

{ cout << "The cube of 3 is: " << cube(3) << "\n"; return 0;

}
```

Output:

```
The cube of 3 is: 27
```

Q.9 EXAMPLE OF FUNCTION OVERLOADING :

INPUT :

```
#include <iostream> using namespace std; float absolute(float var){ if (var < 0.0) var = -var; return var;

} int absolute(int var) {
```

```
if (var < 0) var = -var; return var;
} int main() { cout << "Absolute value of -5 = " << absolute(-5) << endl; cout << "Absolute value of 5.5 = " << absolute(5.5f) << endl; return 0;
}
```

OUTPUT :

```
Absolute value of -5 = 5
Absolute value of 5.5 = 5.5
```

Q.10 EXAMPLE OF DEFAULT ARGUMENT FUNCTION:

INPUT :

```
#include <iostream> using namespace std;
int sum(int x, int y, int z = 0, int w = 0)
{ return (x + y + z + w);
} int main() { cout << sum(10, 15) << endl; cout << sum(10, 15, 25) << endl; cout << sum(10, 15, 25, 30) << endl; return 0;
}
```

Output :

```
25
50
80
```

Q.11 EXAMPLE OF DEFAULT CONSTRUCTOR OR NO ARGUMENTS :

INPUT :

```
#include <iostream> using namespace std; class DemoDC {
private: int num1, num2 ; public:
DemoDC() { num1 = 10; num2 = 20;
}
void display() { cout<<"num1 = "<< num1 <<endl; cout<<"num2 = "<< num2 <<endl;
}
};
int main() {
DemoDC obj;
obj.display(); return 0;
}
```

Output :

```
num1 = 10 num2 = 20
```

Q.12 EXAMPLE OF PARAMETERIZED CONSTRUCTOR:

INPUT :

```
#include <iostream> using namespace std; class Point
{ private:
int x, y;
public:
// Parameterized Constructor
Point(int x1, int y1)
{ x = x1; y = y1;
}
int getX()
{
return x;
}
int getY()
{
return y;
}
};
int main()
{
Point p1(10, 15); cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY(); return 0;
}
```

Output:

```
p1.x = 10, p1.y = 15
```

Q.13 EXAMPLE OF COPY CONSTRUCTOR :

INPUT :

```
#include<iostream> using namespace std; class Point
{ private:
int x, y;
public:

Point(int x1, int y1) { x = x1; y = y1; }

// Copy constructor

Point(const Point &p1) {x = p1.x; y = p1.y; }

int getX() { return x; } int getY() { return y; }

}; int main()

{

Point p1(10, 15); Point p2 = p1; cout << "p1.x = " << p1.getX() << ", p1.y = " << p1.getY(); cout << "\np2.x = " << p2.getX()
<< ", p2.y = " << p2.getY(); return 0;

}
```

Output:

```
p1.x = 10, p1.y = 15 p2.x = 10, p2.y = 15
```

Q.14 EXAMPLE OF CONSTRUCTOR OVERLOADING :

INPUT :

```
#include <iostream>

using namespace std; class construct

{ public: float area; construct()

{

area = 0;

}

construct(int a, int b)

{

area = a * b;

}

void disp()

{

cout<< area<< endl;

}

};

int main()

{ construct o; construct o2( 10, 20);

o.disp(); o2.disp(); return 1;
```

}

Output:

0

200

Q.15 EAMPLE OF DESTRUTOR :

INPUT :

```
class String { private: char* s; int size;

public:

String(char*); // constructor

~String(); // destructor

};

String::String(char* c)

{ size = strlen(c); s = new char[size + 1]; strcpy(s, c);

}

String::~String() { delete[] s; }
```

Q.16 EXAMPLE OF FREIND FUNCION WITH CLASS:

INPUT :

```
#include <iostream> class A { private:

int a;

public:

A() { a = 0; } friend class B; // Friend Class

}; class B { private:

int b;

public: void showA(A& x)

{

std::cout << "A::a=" << x.a;

}

};

int main()

{
```

- a;
- b;

```
b.showA(a); return 0;

}
```

Output:

```
A::a=0
```

Q.17 EXAMPLE OF CONSTRUTOR WITH DEFAULT ARGUMENTS :

INPUT :

```
#include<iostream>

using namespace std; class Simple{ int data1; int data2; int data3; public:

Simple(int a, int b=9, int c=8){ data1 = a; data2 = b; data3 = c;

}

void printData();

};

void Simple :: printData(){

cout<<"The value of data1, data2 and data3 is "<<data1<<", "<< data2<<" and "<< data3<<endl;

}
```

Q.18 STATIC DATA MEMBER :

INPUT :

```
#include <iostream>

using namespace std; class A

{ public:

A() { cout << "A's Constructor Called " << endl; }

};

class B

{ static A a; public:

B() { cout << "B's Constructor Called " << endl; }

};

int main()

{

B b; return 0;

}
```

Output:

```
B's Constructor Called
```

Q.19 STATIC FUCTION CANNOT ACCESS NON-STATIC MEMBERS :

INPUT :

```
class A{ static int b; public: static int GetValue(){ b = 10; return b;
} };
int A::b =50; int main(){ cout<< A::GetValue(); return 0;
}
```

OUTPUT:

```
10
```

Q.20 OVERLOAD UNARY INCREMENT(+) OPERATER :

INPUT :

```
#include <bits/stdc++.h> using namespace std; class Integer { private:
int i;
public:
Integer(int i = 0)
{ this->i = i;
}
Integer operator++()
{
Integer temp; temp.i = ++i; return temp;
}
void display()
{
cout << "i = " << i << endl;
}
};
int main()
{
Integer i1(3); cout << "Before increment: ";
i1.display(); Integer i2 = ++i1; cout << "After pre increment: ";
i2.display();
```

}

Output:

```
Before increment: i = 3

After pre increment: i = 4
```

Q.21 OVERLOAD BINARY OPERATOR :

INPUT :

```
#include<iostream> using namespace std; struct X { void operator*(int) { } void operator*(X, float) { } int main() { X x; int y = 10; float z = 10; x * y; x * z; }
```