

① To add 2 numbers.

→

```
#include <iostream>
using namespace std;
int main()
{
    int a, b, sum = 0;
    cout << "Enter 2 numbers\n";
    cin >> a >> b;
    sum = a + b;
    cout << "sum is " << sum;
    return 0;
}
```

Output:

Enter any 2 numbers

5

6

Sum is 11

② Arithmetic operations using switch.

→

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

{

~~float~~ a, b;

char ope;

cout << "+, -, *, *";

cin >> ope;

cout << "Enter 2 numbers";

cin >> a >> b;

switch (ope)

{

```
case '*': cout << "Product is" << a*b;  
    break;  
case '+': cout << "sum is" << a+b; break;  
    break;  
case '-': cout << "Difference is" << a-b;  
    break;  
case '/': if (b != 0)  
    cout << "Division is" << a/b;  
    else  
    cout << "cant divide";  
    break;  
}  
return 0;  
}
```

Output:

+ - * / : *

Enter two numbers: 5

2

Product is 10

③ Check if the numbers is even or odd.
→

```
#include<iostream>
using namespace std;
int main()
{
    int num; //even, odd
    cout << "Enter an integer:"; //100
    cin >> num;

    if (num % 2 == 0)
    {
        cout << num << " is even";
    }
    else
        cout << num << " is odd";
    return 0;
}
```

output:

Enter an integer: 6

6 is even.

④ Print 1 to 10 using:

(i) for loop

```
→ #include <iostream>
using namespace std;
int main()
{
    int i;
    for (i=0; i<10; i++)
    {
        cout << "n" << i;
    }
    return 0;
}
```

Output: 1 2 3 4 5 6 7 8 9 10 in vertical

(ii) while loop

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

```
{
    int i;
    i=0;
    while (i<11)
    {
        cout << "n" << i;
        i++;
    }
    return 0;
}
```

Output:

1
2
3
4
5
6
7
8
9
10

5) 9) *

* *

* * *

→ #include <iostream>
int main() → using ~~main~~ namespace std;
{
int i, j, s;
for (i=1; i<=3; i++)
{
for (s=1; s<=3-i; s++)
{
cout << " ";
}
for (j=1; j<=i; j++)
{
cout << "*";
cout << " ";
}
cout << "\n";
}
return 0;
}

Output:

*

* *

* * *

b) 1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

→ #include <iostream>
int main() [→] using namespace std;
{
 int i, j;
 for (i = 1; i <= 5; i++)
 {
 for (j = 1; j <= i; j++)
 {
 cout << j;
 }
 cout << "\n";
 }
 return 0;
}

Output :

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

(1) 1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

→ #include <iostream>
int main() { using namespace std;
{
int i, j;
for (i=1; i<=5; i++)
{
for (j=1; j<=i; j++)
{
cout << i;
}
cout << "In";
}
return 0;
}

Output: 1
2 2
3 3 3
4 4 4 4
5 5 5 5 5

Qn
1318

Experiment-1

1) WAP to declare a class student having data members as roll, name. Accept & display data for one object.

```
→ #include <iostream>
using namespace std;
class student
{
private:
    string name;
    int roll;
public:
    void accept()
    {
        cout << "Enter student name\n";
        cin >> name;
        cout << "Enter roll no.\n";
        cin >> roll;
    }
    void display()
    {
        cout << "Student name:" << name;
        cout << "\n Roll no.: " << roll;
    }
};
int main()
{
    student s1;
    s1.accept();
    s1.display();
}
```

Output:

Enter student name

Sandesh

Enter roll no.

52

Student name: Sandesh

Roll no. : 52

2) WAP to declare a class Book having data members as bid, bname, bprice. Accept data for 2 books & display name of book having greater price.

```
→ #include<iostream>
using namespace std;
class book
{
private:
    string name;
    float id;
public:
    float price;
    void accept()
    {
        cout<< "Enter book id \n";
        cin>>id;
        cout<< "Enter book name \n";
        cin>>name;
        cout<< "Enter price \n";
        cin>>price;
    }
    void display()
    {
        cout<< "Book id: " << id;
        cout<< "\n Book name: " << name;
        cout<< "\n Price: " << price;
    }
};

int main()
```

{

```
book b1, b2;  
b1.accept();  
b2.accept();  
b1.display();  
b2.display();  
if (b1.price > b2.price)  
cout << "In Book 1 price is more";  
else  
cout << "In Book 2 price is more";
```

}

Output:

Enter book id

123

Enter book name

abc

Enter price

986

Enter book id

456

Enter book name

xyz

Enter price

1065

Book name: abc

Book id: 123

Price: 986

Book name: xyz

Book id: 456

Price: 1065

Book 2 price is more

M	T	W	T	F	S	S
Page No.:						
Date:						YOUVA

3) WAP to declare a class Time. Accept time in HH:MM:SS & display total time in seconds.

→

```
#include <iostream>
using namespace std;
```

```
class Time {
```

```
public:
```

```
float hr, min, sec;
```

```
float sum;
```

```
void accept () {
```

```
cout << "Enter hours:";
```

~~```
cout << cin >> hr;
```~~

```
cout << "Enter minutes:";
```

```
cin >> min;
```

```
cout << "Enter seconds:";
```

```
cin >> sec;
```

```
}
```

```
void display () {
```

```
cout << "Hours :" << hr << "\n";
```

```
cout << "Minutes :" << min << "\n";
```

~~```
cout << "Seconds :" << sec << "\n";
```~~

```
}
```

```
void tosecond () {
```

```
sum = hr * 3600 + min * 60 + sec;
```

```
cout << "Total seconds :" << sum << "\n";
```

```
}
```

```
};
```

```
int main() {  
    Time t1;  
    t1.accept();  
    t1.display();  
    t1.toSecond();  
    return 0;  
}
```

Output:

Enter hour: 1

Enter minutes: 1

Enter seconds: 1

Hour: 1

Minutes: 1

Seconds: 1

Total seconds: 3661

Q

1316

Experiment-2

- a) WAP to declare a class 'city' having data members as name and population. Accept this data for 5 cities and display name of city having highest population.

→

```
# include <iostream>
using namespace std;
```

```
class city
```

```
{
```

```
private:
```

```
string name;
```

```
public:
```

```
int pop;
```

```
void accept()
```

```
{
```

```
cout << "Name of city";
```

```
cin >> name;
```

```
cout << "Population";
```

```
cin >> pop;
```

```
}
```

```
void display()
```

```
{
```

```
cout << "Name of city:" << name;
```

```
cout << "Population:" << pop;
```

```
}
```

```
};
```

```
int main()
```

```
{
```

```
city c[5];
```

```
int i, max; = 0;
```

```
for (i=0; i<5; i++)
```

```

    {
        c[i].accept();
    }
    max = c[0].pop;
    for (i=0; i<5; i++)
    {
        if (c[i].pop > max) c[max].pop
        {
            max = i;
        }
    }
    cout << "In city with highest population" << endl;
    c[max].display();
}

```

Output :

Name of city : Pune

Population : 10000

Name of city : Mumbai

Population : 20000

Name of city : Indore

Population : 30000

Name of city : Nagpur

Population : 40000

Name of city : Kolkata

Population : 50000

Name of city : Kolkata

Population : 50000

b) WAP to declare a class 'Account' having data members as Account no. and balance. Accept this data for 10 accounts and give interest of 10%. where balance is equal or greater than 5000 and display them.

→ `#include <iostream>`
using namespace std;

```
class Account {  
    int accNo;  
    float balance;  
  
public:  
    void accept() {  
        cout << "Enter Account Number: ";  
        cin >> accNo;  
        cout << "Enter Balance: ";  
        cin >> balance;  
    }  
  
    void addinterest()  
    {  
        if (balance >= 5000) {  
            balance = balance + (balance * 10 / 100);  
        }  
    }  
  
    void display()  
    {  
        cout << "Account No: " << accNo << " Balance: "  
            << balance << endl;  
    }  
}
```

```
float getBalance ()  
{  
    return Balance;  
}  
};
```

```
int main () {  
    Account a[10];  
    for (int i=0; i<10; i++)  
    {  
        a[i]. accept();  
        a[i]. addInterest();  
    }
```

```
cout << "In Accounts with Balance >= 5000 \n";  
for (int i=0; i<10; i++)  
{  
    if (a[i]. getbalance () >= 5000)  
    {  
        a[i]. display();  
    }  
}  
return 0;
```

O/P :

```
Enter Account number: 1  
Enter Balance: 5000  
Enter Account number: 2  
Enter Balance: 6500  
Enter Account number: 3  
Enter Balance: 4000
```

Enter Account Number: 4

Enter Balance: 8900

Enter Account Number: 5

Enter Balance: 3200

Enter Account Number: 6

Enter Balance: 7800

Enter Account Number: 7

Enter Balance: 1200

Enter Account Number: 8

Enter Balance: 10000

Enter Account Number: 9

Enter Balance: 2300

Enter Account Number: 10

Enter Balance: 4999

Accounts with Balance ≥ 5000 :

Account No: 1 Balance: 5500

Account No: 2 Balance: 7150

Account No: 4 Balance: 9790

Account No: 6 Balance: 8580

Account No: 8 Balance: 11000

c) WAP to declare a class 'staff' having data members as name and post. Accept this data for 5 staff and display names of staff, who are "HOD".

→ # include <iostream>

using namespace std;

class staff

{

string name, post;

public:

void accept()

{

cout << " Enter staff Name:";

cin >> name;

cout << " Enter staff post:";

cin >> post;

}

void display()

{

cout << "\n Staff Name:" << name << "\n Staff Post:" << post << endl;

if (post == "HOD" || post == "hod")

{

cout << name << " is Head of Department.\n";

}

else

{

cout << name << " is a staff.\n";

}

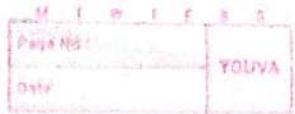
}

```
int main()
{
    staff s[5];
    int i;
    for (i=0; i<5; i++)
    {
        s[i]. accept();
    }
    for (i=0; i<5; i++)
    {
        s[i]. display();
    }
    return 0;
}
```

~~Q5~~

~~Q6~~

Experiment-3



a) WAP to declare a class 'book' containing data members as book-title, author-name, and price. Accept and display the information for one object using a pointer to that object.

```
#include<iostream>
using namespace std;
class book{
    string title;
    string a-name;
    int price;
public:
    void info () {
        cout<< "enter the book title, author name
        and price of the book:" ;
        cin >> title >> a-name >> price;
    }
    void display () {
        cout<< "Book name:" << title << "Author
        name:" << a-name << "Price:" << price;
    }
};

int main () {
    Book * n;
    Book bl;
    n = & bl;
    n -> info ();
    n -> display ();
}
```

Output:

enter the book title, author name and price
of the book:

Book 1
J.K. Rowling
950

Book name: Book 1 author name: J.K. Rowling
price: 950

2) WAP to declare a class 'student' having data members roll-no and percentage. Using 'this' pointer.

→

```
#include <iostream>
using namespace std;
class student {
```

int roll;
float perc;

public:

void info(int roll-no, float percentage) {

 this → roll = roll - no;

 this → perc = percentage;

}

void display () {

 cout << "roll no.:" << roll << " marks :" << perc;

}

}

int main () {

 student s;

 s.info(3, 90);

 s.display();

}

O/P:

roll no.: 3

marks: 90

Nested class

```
3> #include<iostream>
using namespace std;
class Marks {
public:
class Percentage {
int marks, n;
float i;
public:
void info () {
cout<<"Enter the marks you got:";
cin >> marks;
cout<<"Enter total marks:";
cin >> n;
}
void display () {
i = (float)marks/n;
n = i * 100;
cout<<"percentage:" << n;
}
};
int main () {
Marks m1;
Marks :: Percentage n1;
n1.info ();
n1.display ();
}
```

O/P :

Enter the marks you got : 67

Enter total marks : 70

Percentage : 95.71

Experiment - 9

| | | | | | | |
|-----------|---|---|---|---|---|-------|
| M | T | W | T | F | S | S |
| Page No.: | | | | | | |
| Date: | | | | | | YOUVA |

1. Swap 2 numbers from same using object as function argument.

```
→ #include <iostream>
using namespace std;
class number {
    int value;
public:
    number ( int r=0 ) {
        value = r;
    }
    void swap ( number & other ) {
        int temp = value;
        value = other. value;
        other. value = temp;
    }
    void disp () {
        cout << "Value: " << value << endl;
    }
};

int main () {
    number n1(10), n2(20);
    cout << "Before swap: " << endl;
    n1. disp ();
    n2. disp ();
    n1. swap (n2);
    cout << "After swap: " << endl;
    n1. disp ();
    n2. disp ();
    return 0;
}
```

O/P : Before swap:
Value : 10
Value : 20

After swap:
Value : 20
Value : 10

2. Swap 2 numbers from same class using friend function

```
→ #include<iostream>
using namespace std;
class AB {
    int a, b;
public:
    void info() {
        cout << "Enter 2 numbers: ";
        cin >> a >> b;
    }
    friend void swap(AB a1);
}
void swap(AB a1) {
    int temp;
    temp = a1.a;
    a1.a = a1.b;
    a1.b = temp;
    cout << "In Values after swapping: " << a1.a <<
    a1.b;
}
int main() {
    AB a1;
    a1.info();
    swap(a1);
}
```

O/P :

Enter 2 numbers : 5

9

Value after swapping : 9

5

3. Friend function swap 2 numbers different class

```
#include <iostream>
using namespace std;
class B;
class A;
int num A;
public:
(A (int val) : num A (val) {})
void disp () {
cout << "value in class A :" << num A << endl;
}
friend void swap ( A &, B & );
class B {
private:
int num B;
(B (int val) : num B (val) {})
void disp () {
cout << "value in class B :" << num B << endl;
}
friend void swap ( A &, B & );
}
void swap ( A & a, B & b ) {
int temp = a. num A;
a. num A = b. num B;
b. num B = temp;
}
```

```
int main () {
A obj A (10);
B obj B (20);
```

```

cout << "Before swapping:" << endl;
obj A. disp();
obj B. disp();
swap (obj. A, obj. B);
cout << "In After swapping:" << endl;
obj A. disp();
obj B. disp();
return 0;
}

```

O/P :

Before swapping:
value in class A : 10
value in class B : 20

After swapping:

~~Value in class A : 20~~
~~Value in class B : 10~~

4. Avg of two results.

```
→ #include <iostream>
using namespace std;
class result2;
class result{
    int a;
public:
    void accept() {
        cout << "Enter marks out of 50:";
        cin >> a;
    }
    friend void cal(result r1, result2 r2);
};
class result2 {
    int b;
public:
    void accept() {
        cout << "Enter marks out of 50:";
        cin >> b;
    }
    friend void cal(result r1, result r2);
};
void cal(result r1, result r2) {
    float avg = (float) (r1.a + r2.b) / 2;
    cout << "Avg: " << avg;
}
int main() {
    result x;
    result2 y;
    x.accept();
    y.accept();
    cal(x, y);
}
```

O/P: Enter marks out of 50: 45
 Enter marks out of 50: 46
 avg: 45.5

5. Greatest among 2 numbers (Diff class) (Friend function)
 → `#include<iostream>`

`using namespace std;`

`class B;`

`class A {`

`int a;`

`public:`

`void acc() {`

`cout << "Enter value:";`

`cin >> a;`

`}`

`friend void gr(A a1, B b1);`

`};`

`class B {`

`int b;`

`public:`

`void accept() {`

`cout << "Enter a value:";`

`cin >> b;`

`}`

`friend void gr(A a1, B b1);`

`};`

~~`void gr(A a1, B b1) {`~~

~~`if (a1.a > b1.b) {`~~

~~`cout << "First value is greater";`~~

~~`}`~~

~~`else {`~~

~~`cout << "Second value is greater";`~~

~~`}`~~

Page No.:
Date:
YOUVA

```
int main() {  
    A x;  
    B y;  
    x.acc();  
    y.acc();  
    gr(x, y);  
}
```

O/P :

Enter value: 10

Enter value: 100

Second value is greater.

Ques

1478

Experiment - 5

~~Q 9) write a CPP program to implement types of constructor~~
Experiment - 4

① Passing of object as a function argument (swap)

→ `#include <iostream>`

`using namespace std;`

`class number`

`{`

`int value;`

`public:`

`number (int v=0) {`

`value = v;`

`}`

`void swap (number & other) {`

`int temp = value;`

`value = other.value;`

~~`other.value = temp;`~~

`}`

`void disp () {`

`cout << "Value:" << value << endl;`

`}`

`int main () {`

`number n1(10), n2(20);`

`cout << "Before swap:" << endl;`

`n1.disp();`

`n2.disp();`

~~`n1.swap(n2);`~~

~~`cout << "In After swap:" << endl;`~~

`n1.disp();`

`n2.disp();`

`return 0;`

`}`

→ O/P: Before swap:

Value: 10

Value: 20

After swap:

Value: 20

Value: 10

② Friend function (swap same class)

→ `#include<iostream>`

`using namespace std;`

`class AB {`

`int a, b;`

`public:`

`void info () {`

`cout<<" Enter 2 numbers:";`

`cin>>a>>b;`

`}`

`friend void swap (AB &a1);`

`}`

`void swap (AB &a1) {`

`int temp;`

`temp = a1.a;`

`a1.a = a1.b;`

`a1.b = temp;`

`cout<<"\n values after swapping:"<<a1.a<<a1.b;`

`}`

`int main () {`

`AB a1;`

`a1.info;`

`swap (a1);`

`}`

→ O/P: Enter 2 numbers: 5

9

Values after swapping: 9 5

③ friend function (swap different class)

→ `#include <iostream>`

`using namespace std;`

`class C B;`

`class C A {`

`int num A;`

`public:`

`(A (int val) : num A (val) {}`

`void disp () {`

`cout << "Value in class A:" << num A << endl;`

`}`

`friend void swap (C A &, C B &);`

`};`

`class C B {`

`private:`

`int num B;`

`(B (int val) : num B (val) {}`

`void disp () {`

~~`cout << "Value in class B:" << num B << endl;`~~

`}`

~~`friend void swap (C A &, C B &);`~~

`};`

~~`void swap (C A & a , C B & b) {`~~

~~`int temp = a . num A ;`~~

~~`a . num A = b . num B ;`~~

~~`b . num B = temp ;`~~

`}`

Page No. : 5
Date : 10/10/2023
YUVRAJ

```
int main() {
    A obj A(10);
    B obj B(20);
    cout << "Before swapping: " << endl;
    obj A. disp();
    obj B. disp();
    swap (obj A, obj B);
    cout << "In After swapping : " << endl;
    obj A. disp();
    obj B. disp();
    return 0;
}
```

O/P:

Before swapping :

Value in class A: 10

Value in class B: 20

After swapping:

Value in class A: 20

Value in class B: 10

④ Avg of two results

→ #include <iostream>

using namespace std;

class result2;

class result1{

int a;

public:

void accept() {

cout << "Enter marks out of 50 : ";

(in >> a;

}

```
friend void cal(result r1, result2 r2);  
};  
class result2 {  
    int b;  
public:  
    void accept() {  
        cout << "Enter marks out of 50:";  
        cin >> b;  
    }  
    friend void cal(result r1, result2 r2);  
};  
void cal(result r1, result2 r2) {  
    float avg = (float)(r1.a + r2.b) / 2;  
    float avg  
    cout << "Avg: " << avg;  
}  
int main() {  
    result x;  
    result2 y;  
    x.accept();  
    y.accept();  
    cal(x, y);  
}
```

O/P:

Enter marks out of 50: 45

Enter marks out of 50: 56

Avg: 50.5

⑤ Greatest among 2 numbers (different class)
(friend function)

```
#include<iostream>
using namespace std;

class B;
class A {
    int a;
public:
    void acc() {
        cout << "Enter value: ";
        cin >> a;
    }
    friend void gr(A a1, B b1);
};

class B {
    int b;
public:
    void acc() {
        cout << "Enter a value: ";
        cin >> b;
    }
    friend void gr(A a1, B b1);
};

void gr(A a1, B b1) {
    if (a1.a > b1.b) {
        cout << "First value is greater";
    }
    else {
        cout << "Second value is greater";
    }
}

int main()
```

A x;
B y;
x. acc();
y. acc();
gr(x, y);
}

O/P:

~~Enter value: 10~~

~~Enter value: 100~~

~~Second value is greater~~

Ques
12/11

Experiment-4

* Write a C++ program to demonstrate passing of an object as function argument.

① Average of two results using 2 class.

→ #include <iostream>

using namespace std;

class result2;

class result1 {

int a;

public:

void acc() {

cout << "Enter marks out of 50 :";

cin >> a;

}

friend void cal(result1 r1, result2 r2);

}

class result2 {

int b;

public:

void acc() {

cout << "Enter marks out of 50 :";

cin >> b;

}

friend void cal(result1 r1, result2 r2);

}

void cal(result1 r1, result2 r2) {

float avg = (float) (r1.a + r2.b) / 2;

cout << "Avg : " avg << avg;

}

int main() {

result1 x;

result2 y;

```

x1.acc();
y.acc();
cal(x1,y);
}

```

D/P: Enter marks out of 50 : 45

Enter marks out of 50 : 46

avg : 45.5

friend

② Find the greater number using 2 classes and function

→ #include <iostream>

using namespace std;

class BB;

class AA {

int a;

public:

void acc() {

cout << "enter 'a' value:";

cin >> a;

}

friend void avg (AA a1, BB b1);

}

~~class BB {~~

int b;

public:

void acc() {

cout << "enter 'b' value:";

cin >> b;

}

friend void avg (AA a1, BB b1);

}

```

void avg (AA a), BB b) {
if (a.a > b.b) {
cout << "In a is the greater value";
}
else {
cout << "In b is the greater value";
}
int main() {
AA a;
BB b;
a.acc();
b.acc();
avg (a, b);
}

```

O/P : Enter 'a' value : 4

Enter 'b' value : 10

b is the greater value

③ Swap values using friend function

→ #include <iostream>
using namespace std;
class AA {
int a, b;
public:
void acc() {
cout << "enter 2 values:";
cin >> a >> b;
}
friend void swap (AA a1);
};

```
void swap (AA a1) {  
    int temp;  
    temp = a1.a;  
    a1.a = a1.b;  
    a1.b = temp;           after  
    cout << "In Values after swapping: " << a1.a  
        << a1.b;  
}  
  
int main() {  
    AA a1;  
    a1. acc();  
    swap (a1);  
}
```

D/I/P :

Enter two values : 4

5

Values after swapping : 5

4

Ques
12/11

Experiment-5

Q. Write a C++ program to implement types of constructors. a) Find sum of nos betn 1 to n using constructor where value of n will be passed to the constructor.

① Default constructor

```
#include<iostream>
```

```
using namespace std;
```

```
class sum
```

```
{
```

```
    int n;
```

```
    public:
```

```
    sum() {
```

```
        n = 5;
```

```
}
```

```
    void calc()
```

```
{
```

```
    int s = 0, i;
```

```
    for (i = 0; i <= n; i++)
```

```
{
```

```
        s = s + i;
```

```
}
```

```
    cout << "Sum is:" << s;
```

```
}
```

```
};
```

```
int main()
```

```
{
```

```
    sum s1;
```

```
    s1.calc();
```

```
}
```

O/P:

Sum is : 15

② Parameterized constructor

```
#include <iostream>
using namespace std;
class sum {
    int no;
public:
    sum(int n) {
        no = n;
    }
    void calc() {
        int s = 0, i;
        for (i = 0; i <= no; i++)
        {
            s = s + i;
        }
        cout << "sum is: " << s;
    }
};
```

```
int main()
{
    sum s1(5);
    s1.calc();
}
```

O/P:

Sum is: 15

③ copy constructor

```
#include<iostream>
using namespace std;
class sum{
    int no;
public:
    sum (int n) {
        no = n;
    }
    sum( sum &su)
    {
        no = su.no;
    }
    void calc()
    {
        int s=0, i;
        for( i=0; i<=no; i++)
        {
            s = s + i;
        }
        cout << "sum is : " << s;
    }
};
```

```
int main()
{
    sum s1(5);
    sum s2(s1);
    s1.calc();
}
```

O/P:
sum is : 15

b) WAP to declare a class "student" having data members as name and percentage. Write a constructor to initialize these data members. Accept and display data for one student.

→ `#include <iostream>` ① Default constructor

`using namespace std;`

`class student {`

`string name;`

`int m1, m2;`

`int total;`

`public:`

`student () {`

`name = "ABC";`

`m1 = 72;`

`m2 = 90;`

`total = 200;`

`}`

`void calc () {`

`float perc = (float)(m1 + m2) / total * 100;`

`cout << "Student Name:" << name << endl;`

`cout << "Percentage:" << perc << endl;`

`}`
`};`

`int main () {`

`student s1;`

`s1. calc();`

`}`

O/P :

Student Name: ABC

Percentage : 81

② Parameterized constructor

```
#include<iostream>
using namespace std;
class student{
    string name;
    int m1, m2;
    int total;
public:
    student(string n, int m1, int m2, int t) {
        name = n;
        m1 = m1;
        m2 = m2;
        total = t;
    }
    void calc()
    {
        float perc = (float)(m1 + m2) / total * 100;
        cout << "Student Name: " << name << endl;
        cout << "Percentage: " << perc << endl;
    }
};
int main()
{
    student s1("ABC", 90, 90, 200);
    s1.calc();
}
```

O/P:

Student Name: ABC
Percentage: 90

② copy constructor

```

#include <iostream>
using namespace std;
class student {
    string name;
    int m1, m2;
    int total;
public:
    student(string n, int m1, int m2, int t) {
        name = n;
        m1 = m1;
        m2 = m2;
        total = t;
    }
    student(student & st) {
        name = st.name;
        m1 = st.m1;
        m2 = st.m2;
        total = st.total;
    }
    void calc()
    {
        float perc = (float)(m1 + m2) / total * 100;
        cout << "Student Name:" << name << endl;
        cout << "Percentage:" << perc << endl;
    }
};

```

int main()

student s1("ABC", 90, 90, 200);

student s2(s1);

s1.calc();

s2.calc();

}

OIP: Student Name: ABC Student Name: ABC
Percentage: 90 Percentage: 90

Q
(2/1)

Experiment-6

1. Single inheritance

```
→ #include <iostream>
using namespace std;
class person {
protected:
    string name;
    int age;
};

class student : protected person {
public:
    void setdata (string n, int a, int r)
    {
        name = n;
        age = a;
        roll_no = r;
    }

    void display ()
    {
        cout << "Name:" << name << endl;
        cout << "Age:" << age << endl;
        cout << "Roll No:" << roll_no << endl;
    }
};

int main ()
{
    student s;
    s.setdata ("Shlok", 20, 101);
    s.display ();
    return 0;
}
```

O/P :

Name: Shlok

Age : 20

Roll No : 101

2. Multiple inheritance

→ #include <iostream>

using namespace std;

class academic {

protected:

string name;

int marks;

};

class sports {

protected:

int score;

};

class student : protected academic, protected sports {

int total;

public:

void accept () {

cout << "Enter name:";

cin >> name;

cout << "Enter marks:";

cin >> marks;

cout << "Enter score:";

cin >> score;

total = marks + score;

}

void display () {

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

cout << "Marks:" << marks << endl;

cout << "Score:" << score << endl;

cout << "Total :" << total << endl;

}

};

```
int main () {  
    student s;  
    s.accept();  
    s.display();  
    return 0;  
}
```

D/P:

Enter name: shlok

Enter marks: 89

Enter score: 7

Name: shlok

Marks: 89

Score: 7

Total: 96

3. Multilevel Inheritance

```

→ #include <iostream>
using namespace std;
class vehicle {
public:
    string model;
    string brand;
};

class car : public vehicle {
protected:
    int type;
};

class electriccar : public car {
public:
    int battery_capacity;
    void accept() {
        cout << " Enter Model:" ;
        cin >> model;
        cout << " Enter Brand:" ;
        cin >> brand;
        cout << " Enter Type (1 for Sedan, 2 for SUV):" ;
        cin >> type;
        cout << " Enter Battery Capacity (in kwh):" ;
        cin >> battery_capacity;
    }

    void display() {
        cout << " Model:" << model << endl;
        cout << " Brand:" << brand << endl;
        cout << " Type:" << (type == 1 ? "Sedan: SUV" )
            << endl;
        cout << " Battery Capacity:" << battery_capacity
            << " kwh" << endl;
    }
}

```

3
};

```
int main () {  
    electriccar  
    electriccar c;  
    c.accept();  
    c.display();  
    return 0;  
}
```

O/P:

Enter Model: X7

Enter Brand: BMW

Enter Type (1 for Sedan, 2 for SUV): 2

Enter Battery capacity (in kwh): 1000

Model: X7

Brand: BMW

Type: SUV

Battery capacity: 1000 kwh

④ Hierarchical

```

→ #include<iostream>
using namespace std;
class emp {
protected:
    string name;
    int id;
};

class man : public emp { private:
    string dep;
};

class dev : public emp {
public:
    string lang;
    string dep;
    void acc() {
        cout << "Emp name:";
        cin >> name;
        cout << "Emp ID";
        cin >> id;
        cout << "Enter programming Lang:";
        cin >> Lang;
        cout << "Enter dept:";
        cin >> dep;
    }

    void disp() {
        cout << "Name:" << name << endl << "ID" << id <<
        endl;
        cout << "Programming Lang:" << Lang << endl <<
        "Dept:" << dep;
    }
};

```

```
int main() {  
    dev d;  
    d.acc();  
    d.disp();  
}
```

O/P:

Emp name: ABC

Emp ID: 101

Enter Programming Lang: CPP

Enter dept: HOD

Name: ABC

ID : 101

Programming Lang: CPP

Dept: HOD

⑤ Hybrid

⑤ Hybrid

```

→ #include <iostream>
using namespace std;
class teacher {
protected:
    string name;
    int age;
public:
    void acc()
    cout << "Enter teacher name:" ;
    cin >> name;
    cout << "Teacher age:" ;
    cin >> age;
}
void disp()
{
    cout << "Teacher name :" << name << endl;
    cout << "Age :" << age << endl;
}
class student : public teacher {
protected:
    int roll;
    string sname;
    void acc()
    cout << "Enter student name:" ;
    cin >> sname;
    cout << "Student roll no:" ;
    cin >> roll;
}
void disp()
{
    cout << "Student name :" << sname << endl;
    cout << "Roll no :" << roll << endl;
}

```

class marks: public student {
protected:

int m1, m2, m3

public:

void accept () {

cout << "Enter marks of 3 subj: ";

cin >> m1 >> m2 >> m3;

}

void dispM () {

cout << "Marks in subj 1, 2, 3: " << m1 << endl << m2 << endl << m3 << endl;

}

class acad: public marks {

public:

int total () {

return m1 + m2 + m3;

}

int main () {

acad a;

a. acc();

a. accs();

a. accm();

a. disp();

a. dispS();

a. dispM();

int total = a. total();

cout << "Total marks: " << total << endl;

}

O/P:

Enter teacher name: ABC

Teacher age: 40

Enter student name: XYZ

Student roll no: 35

Enter marks of 3 subj: 90

90

90

Teacher name: ABC

Age: 40

student name: XYZ

Roll no: 35

Marks in subj 1, 2, 3: 90

90

90

Total marks: 270

Qn
[21/1]

Practical-7

⑥ Area of laboratory (rectangle) and area of classroom (square). (constructor overloading)

→ #include <iostream>

using namespace std;

class xyz {

public:

int l, b, s, a;

void calc(int len, int br) {

l = len;

b = br;

a = l * b;

}

void calc(int si) {

s = si;

a = s * s;

?

}

int main() {

xyz a;

cout << "Enter length and breadth of rectangle : ";

cin >> a.l >> a.b;

a.calc(a.l, a.b);

cout << "Area of rectangle : " << a.a << endl;

xyz b;

cout << "Enter side of square : ";

cin >> b.s;

b.calc(b.s);

cout << "Area of square : " << b.a << endl;

return 0;

?

→ O/P : Enter length and breadth of rectangle : 10
5

Area of rectangle : 50

Enter side of square : 5

Area of square : 25

② constructor overloading (sum of 10 integer and 5 float)

```
#include<iostream>
using namespace std;
class sum {
public:
    float fsum = 0;
    int isum = 0;
    void calc (float a) {
        fsum = fsum + a;
    }
    void calc (int a) {
        isum = isum + a;
    }
};

int main () {
    sum s;
    float f;
    int i;
    cout << "Enter 5 float values : ";
    for (int j = 0; j < 5; j++) {
        cin >> f;
        s.calc();
    }
    cout << "Sum of float values : " << s.fsum << endl;
    cout << "Enter 10 integer values : ";
}
```

```
for (int j=0; j<10; j++) {  
    cin >> i[j];  
    s.calc(i);  
}  
cout << "sum of integer values: " << s.sum << endl;  
return 0;  
}
```

③ Unary-operator

```
#include <iostream>  
using namespace std;  
class numb {
```

```
public:
```

```
int val;
```

```
void acc() {
```

```
cout << "Enter a numb: ";
```

```
cin >> val;
```

```
}
```

```
void displ() {
```

```
cout << "value: " << val;
```

```
}
```

```
void operator - () {
```

```
val = -val;
```

```
}}
```

```
int main () {
```

```
num b obj;
```

```
obj.acc();
```

```
- obj;
```

```
obj.displ();
```

```
}
```

O/P:

Enter a numb: 5

Value: -5

③ Unary ++ operator (Pre & post increment)

```

→ #include <iostream>
using namespace std;
class ab {
public:
    int num;
    void acc() {
        cout << "Enter number: ";
        cin >> num;
    }
    void disp() {
        cout << "Number: " << num << endl;
    }
    void operator++() {
        ++num;
    }
    void operator++(int) {
        num++;
    }
    int main() {
        ab obj;
        obj.acc();
        ++obj;
        obj.disp();
        return 0;
    }
}

```

O/P :

Enter number: 5

Number: 7

Pre
Post

Practical-8

① Overload '+' operator so that two strings can be concatenated.

```
#include <iostream>
#include <string>
using namespace std;
class abc {
public:
    string str;
    void acc() {
        cout << "Enter string:" ;
        cin >> str;
    }
    abc operator + (abc s) {
        abc temp;
        temp.str = str + s.str;
        return temp;
    }
    void disp() {
        cout << "Concatenated string:" << str << endl;
    }
};

int main() {
    abc s1, s2, r;
    s1.acc();
    s2.acc();
    r = s1 + s2;
    r.disp();
    return 0;
}
```

O/P :

Enter string : Hello

Enter string : World

Concatenated string : Hello World

②

```

→ #include <iostream>
using namespace std;
class ILogin {
protected:
    string name, password;
public:
    void accept() {
        cout << "Name: ";
        cin >> name;
        cout << "Password: ";
        cin >> password;
    }
};

class EmailLogin: virtual public ILogin {
public:
    void showEmail() {
        cout << name << " " << password << endl;
    }
};

class MembershipLogin: virtual public ILogin {
public:
    void showMembership() {
        cout << name << " " << password << endl;
    }
};

class Employee: public EmailLogin, public
MembershipLogin {
public:
    void input() {
        accept();
    }
};

```

```
void display() {  
    show Email();  
    show Membership();  
}  
};  
int main() {  
    Employee e;  
    e.input();  
    e.display();  
    return 0;  
}
```

O/P :

Name : ABC

Password : 123

ABC 123

ABC 123

Qn

1211

Practical-9

①

```
→ #include <iostream>
# include <fstream>
using namespace std;
```

```
int main() {
    fstream f1, f2;
    char ch;
    f1.open ("First.txt", ios::in);
    f2.open ("Second.txt", ios::out);
    while (f1.get(ch)) {
        f2.put(ch);
    }
```

```
cout << "File copied successfully." << endl;
```

```
f1.close();
f2.close();
return 0;
}
```

O/P:
File copied successfully.

②

```
#include <iostream>
#include <fstream>
using namespace std;
```

```
int main() {
    fstream file;
    file.open("file.txt", ios::in);
```

```
    if (!file) {
        cout << "Unable to open file";
        return 1;
    }
```

```
    char ch;
```

```
    int digit_count = 0, space_count = 0;
```

```
    while (file.get(ch)) {
```

```
        if (ch >= '0' && ch <= '9') {
            digit_count++;
        }
    }
```

```
    else if (ch == ' ') {
```

```
        space_count++;
    }
```

```
}
```

```
    cout << "digits: " << digit_count << endl;
```

```
    cout << "spaces: " << space_count << endl;
```

```
    file.close();
```

```
    return 0;
}
```

O/P:

Unable to open file

③

```

→ #include <iostream>
#include <fstream>
using namespace std;
int main () {
fstream File;
file.open ("File.txt", ios::in);
if (!File) {
cout << "Unable to open file" << endl;
return 1;
}
char ch;
int word_count = 0;
int prev_type = 0;
while (File.get (ch)) {
int current_type = ((ch >= 'a') && ch <= 'z') ||
(ch >= 'A') && ch <= 'Z') ||
(ch >= '0') && ch <= '9') ? 1 : 0;
if (current_type == 1 && prev_type == 0) {
word_count++;
}
prev_type = current_type;
}
cout << "Word count:" << word_count << endl;
file.close ();
return 0;
}

```

O/P:

Unable to open file

Qn
12/11

Experiment-10

Page No.:

Date:

YOUVA

```
#include <iostream>
using namespace std;
template <typename T>
T sum (T a[], int n) {
    T s = 0;
    for (int i=0; i<n; i++)
        s = s + a[i];
    return s;
}
int main () {
    int i[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};
    float f[10] = {1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9, 10.10};
    cout << sum (i, 10) << endl;
    cout << sum (f, 10) << endl;
}
```

O/P:

55

59.6

②

```

→ #include <iostream>
# include <string>
using namespace std;
template <typename T>
T square (T x) {
    return x * x;
}
template <>
string square <string> (string s) {
    return s + s;
}
int main() {
    cout << square (5) << endl;
    cout << square (string ("Aaa")) << endl;
    return 0;
}

```

O/P:

25

AaaAaa

Q
2/11

③

```
#include<iostream>
using namespace std;
template <class T>
class calculator {
public:
    T add (T a, T b) { return a+b; }
    T sub (T a, T b) { return a-b; }
    T mul (T a, T b) { return a*b; }
    T div (T a, T b) { return (b==0)?0:a/b; }
    T mod (T a, T b) { return (b==0)?0:a%b; }
    T max (T a, T b) { return (a>b)?a:b; }
    T min (T a, T b) { return (a<b)?a:b; }
    T square (T a) { return a*a; }
    T avg (T a, T b) { return (a+b)/2; }
    T cube (T a) { return a*a*a; }
};
```

```
int main() {
    calculator<int> c;
    int ch;
    while (1) {
        cout << "Enter choice:\n";
        cout << "1. Add\n 2. Subtract\n 3. Multiply\n";
        cout << "4. Divide\n";
        cout << "5. Modulus\n 6. Maximum\n 7. Minimum\n";
        cout << "8. Square\n";
        cout << "9. Average\n 10. Cube\n 11. Exit\n";
        cout << "choice: ";
        cin >> ch;
        switch (ch) {
```

```

case 1: {
    int a, b;
    cout << "Enter two numbers: ";
    cin >> a >> b;
    cout << "Result: " << c.add(a, b) << endl;
    break;
}

case 2: {
    int a, b;
    cout << "Enter two numbers: ";
    cin >> a >> b; break;
    cout << "Result: " << c.sub(a, b) << endl;
}

case 3: { break;
    int a, b;
    cout << "Enter two numbers: ";
    cin >> a >> b;
    cout << "Result: " << c.mul(a, b) << endl;
}

case 4: { break;
    int a, b;
    cout << "Enter two numbers: ";
    cin >> a >> b;
    cout << "Result: " << c.div(a, b) << endl;
}

case 5: { break;
    int a, b;
    cout << "Enter two numbers: ";
    cin >> a >> b;
    cout << "Result: " << c.mod(a, b) << endl;
    break;
}

case 6: {
    int a, b;
    cout << "Enter two numbers: ";
    cin >> a >> b;
}

```

cout << "Result:" << c.max(a, b) << endl;
break;
}
case 7: {
int a, b;
cout << "Enter two numbers:";
(in >> a >> b);
cout << "Result:" << c.min(a, b) << endl;
break;
}
case 8: {
int a;
cout << "Enter number:";
(in >> a);
cout << "Result:" << c.square(a) << endl;
break;
}
case 9: {
int a, b;
cout << "Enter two numbers:";
(in >> a >> b);
cout << "Result:" << c.avg(a, b) << endl;
break;
}
case 10: {
int a;
cout << "Enter number:";
(in >> a);
cout << "Result:" << c.cube(a) << endl;
break;
}
case 11: {
return 0;
}

default:

cout << "Invalid choice or exit requested. ";

return 0;

}

}

return 0;

}

O/P:

choice : 1

Enter two numbers: 9

5

Result : 9

Qn
12/11