Vidyalekhan
DATE
PAGE

Experiment : 1

a) WAP to declare a class student having data members as name, roll no, Accept and display data for one student.

```
#include <iostream>
#include <string>
using namespace std;
class student
{
    int r;
    string n;
public
    void accept()
    {
        cout << " Enter name and roll number: ";
        cin >> n >> r;
    }
    void display()
    {
        cout << " Name = " << n << endl;
        cout << " Roll no = " << r << endl;
    }
};
int main()
{
    student;
```

```
    s1.accept();
    s1.display();
    return 0;
}
```

out-put -

```
Enter name and roll number : ABC
33
Name = ABC
Roll number = 33
```

b) WAP to declare a class Book having data members as id, name, price etc. Accept data for 2 books and display data of book having greater price.

```
#include <iostream>
#include <string>
using namespace std;
class Book
{
public:
        int bp, bpgs;
        string n;
        void accept()
    {
        cout << " Enter name and price : ";
        cin >> n >> bp;
        cout << " Enter pages : ";
```

```
    cin >> bpgs;
}
    void display()
{
        cout << " Book name = " << endl;
        cout << " Book price = " << bp << endl;
        cout << " Book pages = " << bpgs << endl;
}
};

    int main()
{
    Book b1, b2;
    b1.accept();
    b2.accept();
    if (b1.bp > b2.bp) {
        cout << " Book with higher price:";
        b1.display();
    }
    return 0;
}
```

Output :

```
Enter name and price : ABC 50
Enter pages : 200
Enter name and price : XYZ 100
Enter pages : 300

Book with higher price :
Book name = XYZ
```

Book price = 100
Book pages = 300

c) WAP to declare a class time having data members as H, M and S. Accept data for one object and display total time in sec

```
#include <iostream>
#include <string>
using namespace std;
class time
{
    int h, m, s, t;
public:
    void accept()
    {
        cout << "Enter hours:";
        cin >> h;
        cout << "Enter minutes:";
        cin >> m;
        cout << "Enter seconds:";
        cin >> s;
    }
    void display()
    {
        t = (h * 3600) + (m * 60) + s;
        cout << "Total time =" << t;
    }
};
```

```
int main ()
{
    Time t1;
    t1 . accept ();
    t2 . display ();
    return 0;
}
```

Output -

```
Enter hours : 4
Enter minutes : 6
Enter seconds : 89
Total Time : 14849
```

19/9

Experiment 2 :

a) WAP to declare class CITY having data members as name and population. Accept data for 5 cities and display the one with higher population

```
#include <iostream>
#include <string>
using namespace std;
class City
{
  public:
        String n;
        int p;
        void accept()
    {
        cout << " Enter city name and
        population : ";
        cin >> n >> p;
    }
      void disp() const
          cout << " City : " << n << ",
          population : " << p;
};
  int main()
{
      City c [5];
      cout << " Enter data of 5 cities : \n";
```

```
    for (int i = 0 ; i < 5 ; i++)
{
        cout << " City " << i + 1 << ":\n";
        cin [i]. accept ();

}
    int m = 0;
    for (int i = 1 ; i < 5; i ++)
{
      if ( c [i] . p > c [m] . p)
        {
            m = i;
        }
}
    cout << " City with the highest populati
                -on :; c [m] . disp ();
    return 0;

}
```

Output -

```
Enter data for 5 cities :
City 1 :
Enter city name and population : ABC 100
City 2 :
Enter City name and population : DEF 250
City 3 :
Enter city name and population : GHI 60
City 4 :
Enter City name and population : JKL 420
```

City 5:
Enter city name and population : XYZ 1000

City with the highest population :
City : XYZ , population : 1000

b) WAP to declare a class 'Account' having data members account no and balance. Accept data for 10 people

```
#Include <iostream>
using namespace std;
class account
{
  public :
     int a ;
     ~~doug~~ double b;
     void input ()
     {
        Cout << " Enter account no & balance : "; Cin >> a >> b;
     }
     void add ()
     {
        If (b >= 5000)
           b+ = b * 0.10 ;
     }
void disp ()
   Cout << " Account no : " << a << ",
      Balance : << b << end l;
```

```
};
int main ()
{
    Account A [10];
    for (int i = 0 ; i < 10 ; i ++)
    {
        Cout << " Enter details for account " <<
        (i + 1) << " : \n";
        A [i] . Input ();
    }
    Cout << " Added balance : " ;
    for (int i = 0 ; i < 10 ; i ++)
    {
        A [i] . b >= 5000)
            A [i] . disp ();
    }
    return 0;
}
```

~~output~~

Output -

```
Enter details for account -
1:
1
3456
2:
2
7890
3:
3
9475
4:
4
6578
```

```
Accounts with added balance :
Account no : 2 , 8679
Account no : 3 , 10422.5
Account no : 4 , 7235.8
Account no : 8 , 10744.8
Account no : 9 , 6245.8
Account no : 10 , 9900
```

5:
5
4980
6:
6
4321
7:
7
4999
8:
8
9768
9:
9
5678
10:
10
9000

c) WAP in C++ to define class staff for 5 people with data members as name and post and display data of those having post HOD.

```
#include <iostream>
#include <string>
using namespace std;
class staff
{
  public:
    string n, p;
    void getdata()
    {
```

```
    cout << " Enter name and post ";
    cin >> n >> p;
}
    void disp ()
        cout << " Name : " << n << ", Post : "
            << p;
};
int main ()
{
    staff s [10];
    cout << " Enter data for 5 people : ";
    for (int i = 0 ; i < 5 ; i++)
    {
        cout << " Name " << i + 1 << " : In ";
        s[i] . getData ();
    }

    int m = 0;
    for (int i = 0 ; i < 5 ; i++)
    {
        if ( s[i] . p == " HOD ")
        {
            m = i;
        }
        cout << " People who are HODs : ";
        s[m] . display ();
        return 0;
    }
}
```

Output -

```
Enter data for 5 people :
Name 1 :
Enter name : A
Post : Intern
Name 2 :
Enter name : B
Post : Principle
Name 3 :
Enter name : C
Post : HR
Name 4 :
Enter Name : D
Post : HOD
Name 5 :
Enter name : E
Post : Intern

People who are HODs :
Name : D , Post : HOD
```

Q2

19/9

Experiment 3

Q1) WAP to declare a class 'book' containing data members as book title and author name.

```
#include <iostream>
#include <string.h>
using namespace std;
class Book
{
    private :
        string book_Title
        string author_Name;
        float price;
    Public :
      void accept ()
    {
        cout << " enter book title :";
        getline ( cin , book title )
        cout << " Enter author name :"
        cin >> author name;
        cout << "enter price :";
        cin >> price;
}
    void display ()
{
    cout << " Book title :" << book_title
    << endl;
    cout << " Author name" : <<
                author_name << endl
    cout << " Price :" << price << endl;
} return 0;
}
```

Q2) WAP to declare a class 'Student' having data members roll_no and percentage using this pointer invoke member function to accept and display for 1.

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

class student {
public :
    int roll no ;
    Float percentage ;
    void accept () {
    Cout << " enter roll no : ";
    Cin >> roll no ;
    Cout << " enter percentage : ";
    }
    {  void display ()
    {
        Cout << " In student INFO \n ";
        Cout << " roll no : " << roll no <<
        endl;
        Cout << " percentage : " << percentage
    }
}
Int main(){
    Student . S1
    S1. accept();
    S1. display ();
    Return 0
}
```

Output :

enter roll no : 1
enter percentage : 90

Student Info
roll no. : 1
percentage : 90

Q3) WAP to demonstrate the use of nested class.

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

class Outerclass {
private :
        int outervalue = 100;
  }

class innerclass {
public :
        void display (outerclass and object) {
                cout << " This is the Inner class."
                << endl;
                cout << " Accessing outer class value
                : " << obj.outervalue << endl; }
        };

void outermethod () {
        cout << " This is the outer class." << end
        l;
```

```
    Innerclass inner;
    inner.display(*this);
  }
};

int main(){
      Outerclass outer;
      outer.outermethod();
      return 0;
  }
```

```
This is the outer class.
This is the inner class
Accessing Outer class value: 100
```

9/9

Experiment : 4

a) WAP to create two classes result 1 and result 2 to store marks and compute ~~average~~ average.

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

class Result2 ;

class Result 1 {
public :
        int marks 1;
        void get-data () {
            Cout << " Enter marks of student (Result 1):
            ";
            Cin >> marks 1;
        }
};

class Result 2 {
public :
        int marks 2;
        void get-data () {
            Cout << " Enter marks of student (Result 2):
            ";
            Cin >> marks 2;
        }
};

void average ( Result r1 , Result r2) {
            Float avg = (r1.marks + r2.marks)/2.0;
            Cout << " Average of two results = " <<
```

```
    avg << endl;
}

int main () {
    result r1;
    result r2;

    r1.getdata();
    r2.getdata();

    average (r1, r2);
    return 0;
}
```

Output :

Enter marks of ~~absent~~ student (Result 1) : 80
Enter marks of student (Result 2) : 90
Average of two results = 85

b) WAP to find greatest number among two numbers from two different classes using friend function

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

class B;

class A {
    int num1;
public:
    void getdata () {
        cout << " Enter number for class A: ";
```

```
    cin >> num1;
}
  friend void Findgreatest (A, B);
};

class B {
    int num2;
public:
    void getdata () {
        cout << "Enter number for class B: ";
        cin >> num1;
    }
    friend void Findgreatest (A, B);
};

void Findgreatest (A a, B b) {
    if (a.num1 > b.num2)
        cout << "Greatest number is :" << a.
        num1 << endl;
    else
        cout << "Greatest number is : << b.num2
        << endl;
}

int main () {
    A obj1;
    B obj2;

    Obj1 . getData ();
    Obj2 . getData ();

    Findgreatest (Obj1, obj2);
    return 0;
}
```

c) WAP for swapping contents of two variables of same class using friend function.

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

class sample {
        int value;
public:
    void getData() {
        cout << "Enter value : ";
        cin >> value;
    }
    void display() {
        cout << value << endl;
    }
    friend void swapvalues(sample and, sample and);
};

void swapvalues(sample and s1, sample and);

void swapvalues(sample and s1, sample and s2) {
        int temp = s1.value;
        s1.value = s2.value;
        s2.value = temp;
}

int main() {
        sample obj1, obj2;
        cout << "Enter data for object : ";
```

```
Obj.getdata();
cout << "Enter data for object 2 : ";
obj2.getdata();

cout << "\nBefore swapping :\n";
cout << "Object 1 = "; obj1.display();
cout << "Object 2 = "; obj2.display();

swapvalues(obj1, obj2);

cout << "\nAfter swapping :\n";
cout << "Object 1 = "; obj1.display();
cout << "Object = "; obj2.display();

return 0;
}
```

d) WAP to swap two numbers from same class using object as function argument. Write swap function as member function.

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

class Number {
    int a, b;

public:
    void getdata() {
        cout << "Enter two numbers : ";
        cin >> a >> b;
    }
```

```
void display(){
    cout << " a = " << a << " , b = " << b <<
    endl;
}

void swapnumbers (Number and obj){
        int temp = obj.a;
        obj.a = obj.b;
        obj.b = temp;
    }
};

int main (){
    Number n;

    n.getdata();

    cout << " Before swapping : ";
    n.display();

    n.swapnumbers(n);

    cout << " After swapping : ";
    n.display();

    return 0;
}
```

Output :

```
Enter two numbers : 10 20
Before swapping : a = 10 , b = 20
After swapping : a = 20 , b = 10
```

e) WAP to swap two numbers from different class using friend function.

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

class classB;

class classA; {
    int num1;

public:
        void getdata () {
            cout << " Enter number for class A: ";
            cin >> num1;
        }

        void display () {
            cout << " class A num1 : " << num1
             << endl;
        }

        friend void swapnumbers (classA &a, classB
        &b); };

class classB {
    int num2;

public:
        void getdata () {
            cout << " Enter number for class B:
            "; cin
            cin >> num2
        }
```

```
void display () {
    cout << " Class B num2 = " << num2 << endl;
}

friend void swapnumbers (Class &a, class B &b);
};

void swapnumbers (Class &a, class B &b) {
    int temp = a.num1;
    a.num1 = b.num2;
    b.num2 = temp;
}

int main () {
    ClassA objA;
    ClassB objB;

    objA.getData ();
    objB.getData ();

    cout << "\n Before swapping :" endl;
    objA.display ();
    objB.display ();

    Swap numbers (objA, objB);

    cout << "\n After swapping :" << endl;
    obj.display ();
    obj.display ();

    return 0;
}
```

Vidyalekhan
DATE
PAGE

Output:

```
Enter number for class A: 5
Enter number for class B: 15

Before swapping:
Class A num1 = 5
Class B num2 = 15

After swapping:
Class A num1 = 15
Class B num2 = 5
```

Extra Question

Q1) Create two classes, class A and class B, each with a private integer. Write a friend function sum() that can ~~does~~ access private data from both classes and return the sum.

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

class class B;

class class A {
    int a;
public:
    void setdata(int x) {a = x;}
    friend int sum(class A, class B);
};

class class B {
```

```
    int b;

public :
        void setdata (int y) { b = y; }
        Friend int sum (class A, class B);
    };

int sum (~~te~~ class A obj A , Class B obj B) {
            return obj A . a + obj B . b;
        }

int main () {
        class A; class B;
        A. setdata (5);
        B. Set data (10);

        Cout << " sum (A, B) << endl ;
        return 0;
    }
```

O/P :
Sum = 15

Q2) WAP with a class Number that contains a private integer. Use a Friend function swap Numbers (Number &, Number &) to swap the private value of two Number objects.

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

Class Number {
        int value;
public :
```

```
void setdata(int v) { value = v; }
void display() { cout << value << endl; }
friend void swapnumbers(Number &, Number &);
};

void swapnumbers(Number &n1, Number &n2){
    int temp = n1.value;
    n1.value = n2.value;
    n2.value = temp
}

int main(){
    Number n1, n2;
    n1.setdata(10);
    n2.setdata(20);

    cout << "Before swap:"; n1.display();
    n2.display();

    swapnumbers(n1, n2);

    cout << "After swap:"; n1.display();
    n2.display();

    return 0;
}
```

Q3) Define two classes Box and Cube, each having a private volume. Write a friend function findGreater(Box, Cube) that determines which object has a larger volume.

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

class Cube;

class Box {
    int volume;
public:
    void setvolume(int v) { volume = v; }
    friend void FindGreater(Box, Cube);
};

class Cube {
    int volume;
public:
    void setvolume(int v) { volume = v; }
    friend void FindGreater(Box, Cube);
};

void FindGreater(Box b, Cube c) {
    if (b.volume > c.volume)
        cout << "Box has greater volume"
        << endl;

    else if (c.volume > b.volume)
        cout << "Cube has greater volume
        " << endl;

    else
        cout << "Both have equal volume"
        << endl;
}
```

```
int main() {
    Box b; Cube c;
    b.setvolume(30);
    c.setvolume(50);

    Find greater (b, c);
    return 0;
}
```

O/P:

Cube has greater volume

Q4) Create a class Complex with real and imaginary parts as private members. Use a friend function to add two complex numbers and return the result as a new complex object.

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

class Complex {
        int real, imag;
public:
    Complex(int r = 0, int i = 0): real(r), imag(i) {}
    void display() { cout << real << " + " << imag << "i" << endl; }
    friend Complex add(Complex, Complex);
};

Complex add(Complex c1, Complex c2) {
    return Complex(c1.real + c2.real, c1.imag + c2.imag);
```

```
}

int main(){
        Complex c1(3,4), c2(5,6);
        Complex c3 = add(c1, c2);

        Cout << " Sum of complex numbers :";
        c3.display();
        return 0;
}
```

Output:-

Sum of complex numbers : 8 + 10i

Q5) Create a class student with private data members : name and three subject marks. Write a friend function CalculateAverage(Student) that calculates and display the average marks.

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

Class Student {
    String name;
    int m1, m2, m3;
public:
        void setdata(String n, int a, int b, int c)
                {
        name = n; m1 = a; m2 = b; m3 = c
        }
    friend void Calculateaverage(Student);
};
```

```
void calculateAverage(Student s) {
    float avg = (s.m1 + s.m2 + s.m3)/3.0;
    cout << "Average marks of " << s.name <<
    " = " << avg << endl;
}

int main(){
    Student s;
    s.setdata("Amit", 80, 90, 70);
    calculateAverage(s);
    return 0;
}
```

Output :

Average marks of Amit = 80

Q6) Create three classes : Alpha, Beta, and Gamma, each with private data member. Write a single friend function that can access all three and print their sum.

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

class Beta; class Gamma;

class Alpha {
    int a;
public:
    Alpha(int x) { a = x; }
    friend void total(Alpha, Beta, Gamma);
};
```

```
};

void total (Alpha A, Beta B, Gamma G) {
    cout << " Sum = " << (A.a + B.b + G.c)
     << endl;
}

int main () {
    Alpha a(10);
    Beta b(20);
    Gamma g(30)
    total (a, b, g);
    return 0;
}
```

O/P

Sum = 60

Q7) Create a class point with private members x and y. Write a friend function that calculate and returns the distance between two point objects.

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

class Point {
    int x, y;
public:
    Point (int a, int b) { x = a; y = b; }
    friend double distance ( Point , Point);
};
```

```
int main(){

point p1(0,0), p2(3,4);
 cout << "Distance =" << distance(p1, p2) << endl;
  return 0;
}
```

Output:

Distance = 5

Q8) Create two classes : BankAccount and Audit. BankAccount holds private balance information. Write a friend function in Audit that accesses and prints balance information for auditing.

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

class Audit;

class BankAccount {
      int balance {
public:
      BankAccount(int b) { balance = b; }
      friend void auditBalance(BankAccount, Audit
      );
};
```

```
class Audit {
    string auditorName;

public:
    Audit(string name) { auditName = name; }
    friend void auditBalance(BankAccount, Audit);
};

void auditBalance(BankAccount acc, Audit a) {
    cout << "Auditor " << a.auditName
         << " Checked balance: " acc.balance
         << endl;
}

int main() {
    BankAccount acc(5000);
    Audit aud("Rani");
    auditBalance(acc, aud);
    return 0;
}
```

Auditor Rani checked balance : 5000

19/9

Unit 2 : Functions in C++

Member Functions

Definition : Functions defined inside a class to operate on its objects.

- Can access private / protected data directly
- Declared inside class, defined inside / outside

Syntax :

```
Class Classname {
public :
    void Functionname (); // declaration
};

Void Classname :: Functionname () {
}                                  // definition
```

Passing Arguments to Functions

(a) Passing Variables

- Normal values are passed

```
void add(int a, int b) { cout << a + b; }
```

(b) Passing objects

- Whole object passed as parameter
- By value → Copy passed, original not changed.

- By reference → actual object passed, changes affect original.

(c) Pass by value

```
void fun(int x) { x = 10; } // no effect on caller
```

(d) Pass by reference

```
void fun(int &x) { x = 10; } // changes caller's value
```

(e) Passing constant

- Use const → object/variable cannot be modified.

```
void display(const int x);
```

(f) Default Arguments

- Provide default values at declaration

```
void fun(int a, int b = 10);
```

Nesting of Member Functions

Definition: One member function calls another member function of the same class.

```
class Example {
    int a;
    void read() { cin >> a; }
public:
    void input() { read(); } // nesting
};
```

Returning Values from Functions

(a) Return Statement - returns value to caller

```
int sum(int a, int b) { return a+b; }
```

(b) Returning by reference - Function returns reference of variable

```
int& fun() { static int x = 10; return x; }
```

(c) Returning object - function can return an object

```
class {
    int a;
public:
    Test(int x) {a = x;}
    Test add(Test t) {return Test(a + t.a);}
    };
```

Inline Function

Function expanded at compile time instead of making a call.

- Used for small, frequently used functions

```
inline int square(int x) { return x * x; }
```

Friend Function

Non-Member function that can access private/protected data of class.

- Declared inside class with Friend keyword

```
class A {
    int x
public :
    friend void show(A);
};
```

Pointer to object

Pointer variable storing address of an object
- Members accessed using ->.

```
Class Test { int a; public : void show()
    { cout << a ; } };

Test obj, *ptr ;
ptr = &obj ;
ptr -> show();
```

This pointer

implicit pointer inside all member functions pointing to the calling object.

```
class Test {
    int a;
public :
    void set (int a) { this -> a = a; }
};
```

Nest / Inner Class

class defined inside another class

. Used for grouping, encapsulation

```
class Outer {
public:
    class Inner {
        void display () { cout << " Inner class "; }
    };
};
```

some pts missing

Experiment 5

Q1) WAP to find the sum of numbers between 1 to n using a constructor where the value of n will be passed to the constructor.

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

class number {
        int num;
public:
        number () {
            cout << " Enter number : " << endl;
            cin >> num;

            int sum = 0;
            for (int i = 1; i <= num; i++) {
                sum = sum + i;
            }

            cout << " Sum of numbers upto
                << num << " is : " sum <<
                endl;
        }
};

int main () {
    number n;

    return n;
}
```

→ output :

Enter a number : 3
The sum of numbers upto 3 is 6.

Q2)

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

class student {
        string name;
        float per;
        public:
            student (string n, float p) {
                name = n;
                per p;
            }

            void display () {
                cout << " Name : " << name << end
                l;
                cout << " Percentage : " << per <<
                    endl; }
            };

int main () {

        student s ( " Amish ", 93.2);
        s.display ();

        return 0;
    }
```

Output :
Name : Amish
Percentage 93.2

Q3)

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

class college {
    int roll;
    string name;
    string course;
    public:
    college (int r = 00, string n = "Unknown",
    string
        c = " Computer Engineering ") {
        roll = r;
        name = n;
        course = c;
    }

void display () {
    cout << " Roll Number : " << roll
    << endl;
    cout << "Name : " << name <<
    endl
    cout " Course : " << course endl;
}
};

int main () {
    College S1 (37, " Toyash ");
    College S2 (18, " Amish ");

    S1. display ();
    S2. display ();

    return ();
}
```

Output :

```
Roll number : 37
Name        : Tayash
Course      : Computer ~~engeeneering~~ engineering
Roll number : 18
Name : Gaurav
Course : Computer engineering
```

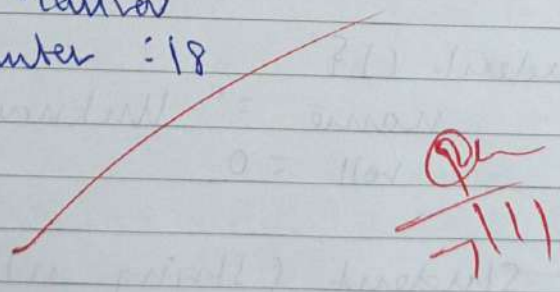
Q4)

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

class student {
        int roll
        string name;
        public :

        student () {
                name :" Unknown";
                roll = 0;
        }
                student (string n) {
                        name :n;
                        roll : 0;
        }
        student (string, int r) {
                name = n;
                roll = r;

        void display () {
                cout " Name :" << name <<
                endl;
                cout << " Roll number : " << roll << endl
                ;
```

```
    }
};

int main(){
    Student S1;          Anish
    Student S2("Anushree");
    Student S3("Gaurav", 18);

    S1.display();
    S2.display();
    S3.display();
    return 0;
}
```

Output :

Name : Unknown
Roll number : 0
Name : Gaurav
Roll number : 18



Experiment 6

Write a program to implement multilevel inheritance. Assume suitable data

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

class department {
protected :
    string name ;
};

class student : protected department { protected :
        string sname;
        int roll;
};

class marks : protected student {
        int m1 , m2 , percentage;
public :
        void accept () {
        cout << " Enter department : "
            << endl;
        cin >> dname;
        cout << " Enter name : " << endl;
        cin >> ;
        cout << " Enter marks1; " <<
            endl; cin >> m1;
        cout << Enter marks 2 : " <<
            endl; cin >> m2;
        }
```

```
void calculate (){
    int per = (m1 + m2) / 2
    cout << " Department : " << dname <<
        end
    cout << "Name : " << sname <<
        << endl ;
    cout << " Percentage : " << per
        << end
    }
};

int main (){
    marks ;
    m. accept ();
    m. calculate ();

    return 0;
}
```

Output :

```
Enter department
Amish
Enter marks 1:
88
Enter marks 2:
91
Department : CSE
Name : Amish
Percentage : 89
```

(2)

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

class department {
protected :
        string dname ;
};

class student {
protected :
        string sname ;
        int roll ;
};

class marks : protected department , protect
              -ed student {

        int m1 , m2 , percentage ;

public :
        void accept () {

                cout << " Enter department :
                " << endl, cin >> dname ;

                cout << " Enter name :" <<
                        endl ; cin >> sname ;
                cout << " Enter marks 1 : " <<
                  endl ; cin >> m1 ;
                cout << " Enter marks 2 :"
                  << endl ; cin >> m2 ;
```

```
    }
        void calculate (){
            int per =(m1 +m2) / 2;
            cout << " Department :" <<
                dname << endl;
            cout << "Name :" << sname <<
                endl;
            cout << " Percentage :" << per
                << endl;
        }
    };
int main (){
    marks m;
    m. accept ();
    m. calculate ();
}
```

Q3)
```
# include <iostream>
using namespace std;

class Person {
    protected :
    string name;
    int age;
    public :
    void accept Per () {
        cout <<" Entername :" << endl;
        cin >> name;
        cout << " Enter age :" << endl;
        cin >> age
    }
```

```
};

Class student : public Person {
        int roll;
        float per;
        public:
        void accept stud () {
            cout << " Enter roll number
            endl;
            cin >> roll;
            cout << " Enter Percentage:"
                                << endl;
            cin >> per;
        }

    void display stud () {

            cout << " Name : "<<name<<endl;
            cout << " Age :"<< age <<endl;
            cout << " Roll No :"<<roll<<endl;
            cout << " Percentage :"<<per<<
              endl;
        }
    };

Class staff : public person {
        int emp-id;
        string subject;
        public:
        void accept staff () {
            cout <<" Enter employee ID:
              << endl;
            cin >> emp-id;
```

```
        cout << " Enter subject : " endl ;
             cin >> subject ;
    }

    void display staff () {
            cout << " Name : " name << endl ;
            cout << " Age : " age << endl ;
            cout << " Emp ID : << emp_id
                 << endl
             cout << " Subject taught : <<
                  subject << endl ;
        }
    };

int main () {
    Student s ;
    staff ;

    s. accept Per () ;
    s. accept () ;
    t. accept Per () ;
    t. accept staff () ;

    s. display Stud () ;
    t. display staff () ;

    return 0 ;
}
```

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

class College {
        protected :
            string name;
    };

class Employee : Protected College {
                protected :
                string emp_name;
            int id;

    };

class staff : public Employee {
            string name;
            int dept ID;
            public :
            void accept Emp() {
                cout << " Enter clg. name
                        : " <<endl;
                cin >> name;
                cout << " Enter emp_name
                        : " <<endl
                cin >> emp_name;
                cout << Enter t]' : " <<endl;
                cin >> id;
                cout << Enter staff name
                        : " <<endl;
                cin >> name;
```

```
cout << " Enter deptid :- " << endl;
    }

void displayEmp(){
    cout << " College :" <<
    (name << endl;
    cout << " Emp name :" << emp
    .name endl
    cout << "ID :" id << endl;
    cout << " Staff Name :" <<
        name << endl
    cout << " Department ID:"
    << deptid << endl
    }
};

Class Student : protected College {
    string stu-name;
    int roll;
    public:
    void accept stud() {
    cout << " Enter College name
        :" end <<
    cin >> cname;
    cout << " Enter name:"
        << endl;
    cin >> stu_name;
    cout << " Enter roll number
        :" << endl;
    cin >> roll;
}

void displayStud(){
```

```
        cout << " College : " << cname << endl;
        cout << " Student name : << Stu_name
                << endl;
        cout << " Roll Number : " roll << endl;


int main() {
        Staff staff;
        Staff1.acceptEmp();
        Staff1.displayEmp();

        Student Stud;
        Stud1.acceptStud();
        Stud1.displayStud();

        return 0;
}
```

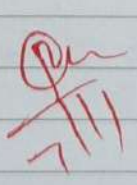
c)

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

class College Student {
    protected :
        int Student_id;
        String Ccode;
    public :
        void accept() {
            cout << " Enter Student ID : ";
            cin >> Student_id;
            cout << " Enter College
                    Code : ";
            cin >> Code;
        }
```

```
void display (){
        cout << " Student ID : " <<
            Student id << endl
        cout << " College Code : "
            Code << endl
    }
};

class Test : Virtual public College Student {
    protected :
        float percentage;
    public :
        void accept (){
            College Student : accept ();
            cout << " Enter Test percentag
            -e : " ;
            cin >> percentage ;
        }

        void display (){
            College Student :: display ();
            cout << " Test percentage : "
                << percentage << "
                % " << endl;
        }
};

class Sports : virtual public College Student
    {
    protected :
        char grade ;
    public :
        void accept (){
```

```
cout << " Enter Sports Grade :";
cin >> grade;

public :
    void accept () {
        cout << "Enter Sports Grade :";
        cin >> grade;
    }

    void display () {
        cout " Sports Grade :" << grade
        << endl;
    }
};

class Result : public Test, public Sports {
    & float tot_marks;

public :
    void accept () {
        College Student : accept ();
        cout << " Enter Test
            Percentage :";
        cin >> Percentage ; Sport
        cout << " Enter Test Percent
            -age :";          Grade :";
        cin >> grade;
        cout << " Enter Total Marks:
        cin >> tot_marks;
    }
```

```
void display (){
        College_student :: display
            ();
        cout << " Test percentage
            " << percentage << "%" <<
                endl
        cout << " Sports Grade :" << grade <<
                endl endl
        cout << " Total marks :" << tot -
                marks endl
            }
    };

int main (){
        Result r;
        cout << " --- Enter Student Details
                --- " << endl;
        r.accept ();
        cout << "\n--- Student Result ---
                " << endl;
        r.display ();
        return 0;
    }
```



Experiment -7

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

Class Area {
public :
    float Calculate (float lenght , float
                     breadth) {
        return lenght * breadth ~~}~~ ;
    }

    float calculate (float side) {
        return side * side ;
    }
};

int main () {
    Area a;
    Cout << " Area of lateratay : "
        << a . calculate (1.0) << endl;
    cout << " Area of square : " << a .
    Calculate (5) << endl;
```

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

Class Sum {
public :
    int total (int a [] , int n ) {
        int s=0;
        for (int i s0 ; i < n , i++)
            + = a [i];
```

```
        return s;
    }
};

int main(){
    Sum s;
        int marks[10] = {45, 56, 67,
78, 89, 90, 76, 88, 92, 85};
        float grades[5] = {9.2, 8.7, 9.5
                , 8.9, 9.0}

    cout << " Sum of 10 student marks
                :" <<
            s.total(marks, 10) << endl;
    cout " Sum of 5 Student grade
                points :"
                << s.total(grades, 5)
                << endl;
        return 0;
    }
```

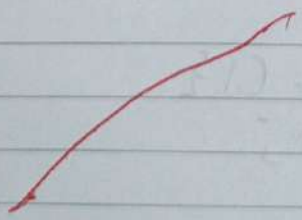
Q3)

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

class Teacher {
public :    int experience;

        Teacher(int e){
            experience = e;
        }
        void display(){
            cout <<" Experience
    << " "year" <<endl;
};
```

```
        void operator-() {
            experience = -experience;
        }
    };

int main() {
    teacher t1(10);
    t1.display();
    -t1;
    cout << "After negation:";
    t1.display();
    return 0;
}
```

4)
```
#include <iostream>
using namespace std;

class Student {
    int count;
public:
    Student(int c=0) {
        count = c;
    }

    void operator++() {
        ++count;
    }

    void operator++(int) {
        count++;
    }
```

```
    }
    void display {
        cout << " Student count : " << cout
                endl;
            }
    };

int main (){
        Student S1( 50);
        cout << " Betae increment : " <<
                endl;
            S1. display;

    ++ S1;
        Cout << After pre-increment : " <<
                endl;
            S1. display();

        S1 ++;
        cout << " After post increment
                : " << endl
            S1. display ();
        return 0;
}
```



PAGE No.
DATE / /

Experiment - 8

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

class combine {
    string str;
public:
    combine (string s : " ")
        str = s;
    }

    combine operator (combine & obj) {
        return combine (str obj.str
        );
    }

    void display () {
        cout << str << endl;
    }
};

int main () {
    combine s1 (" xyz"), s2 (" pqr")
    , s3; s3 = s1 + s2
    cout << "Concatenated string:";
    s3.display ();
```

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

class I login {
    protected:
        string name, passwd;
    public:
        virtual void accept () {
            cout << " Enter name : ";
            cin >> name;
            cout << " Enter passwd : ";
            cin >> passwd;
        }

        virtual void display () {
            cout << " Name : " << Name <<
            name << endl;
            cout << "Passwd : " << passwd
            << endl;
        }
    };

class Email login : public I login {
    string email;
    public:
        void accept () override {
            cout << " Enter Email ID : ";
            cin >> email;
            I login :: accept ();
```

```
}

void display () override {
        cout << "\n --- Email Login
    Details --- " << endl;
    Cout " Email ID : " << email <<
        endl;
    I login :: display();
}
};

class Membership login : public I login
{ string member I);

public:
    void accept() override {
        cout << " Enter Member ship
            ID : ";
        Cin >> member ID;
        Ilogin :: accept ();
    }

    void display () override {
        cout << "\n --- Membership
        login Details --- " << endl;

    cout << " Membership ID : " << member
        ID endl;
        I login :: display ();
}
};
```

7/11

```
int main() {
    Ilogin *login;

    Email login e;
    Membership login m;

    login = &e;
    login -> accept();
    login -> display();

    login = &m;
    login -> accept();
    login -> display();

    return 0;
}
```

Experiment 9

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

int main () {
    ifstream inFile ("First.txt");
    ofstream outFile ("Second.txt");

    if (!inF) {
        cout " Error opening First.txt"
        << endl
        return 1;
    }

    char ch;
    while (inFile.get (ch)) {
        outfile.put (ch);
    }

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

    infile.close();
    outfile.close();
    return 0;
}
```

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

int main() {
    ifstream file("First.txt");
    if (!file) {
        cout << "Error opening file"
             << endl;
        return 1;
    }
    char ch;
    int digit = 0; spaces = 0;

    while (file.get(ch)) {
        if (isdigit(ch))
            digits++
        else if (isspace(ch))
            spaces++;
    }
    cout << "Digits : " << digits <<
        endl;
    cout << " Spaces : " << spaces
        << endl;
    file.close();
    return 0;
}
```

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

int main() {
    ifstream file ("first.txt");
    if (!file) {
        cout << " Error " << endl;
        return 0;
    }
    string word;
    int count = 0

    while ( files >> word)
        count ++;

cout << " Total words : " << count << endl;

    file.close();
    return 0;
}
```

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

int main () {
        ifstream file ("First.txt");
        if (!file) {
            cout << "Error" << endl;
            return 1;
        }

        string word, target;
        int count = 0;

        cout << "Enter word to count: " << endl;
        cin >> target;

        while (file >> word) {
            if (word == target)
                count++;
        }
        cout << "Occurence of " << target
             << " : " << count << endl;
        file.close();
}
```

Experiment 10

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

template (class T>
T sum (T arr, int n) {
T sum = 0
for (int i = 0 ; i<n ; i++)
        sum += arr (i),
            return sum;
            }
int main (){
int arr : [5] = { 1, 2, 3, 4, 5};
cout << " Sum of Array :" << arr sum
( arr , 5);
}
```

O/P

sum of array : 15

```
#include <iostream>
#include <string>
using namespace std;
template <class T>
    T sq ( In) {
    T sq (Tn){
        return n n;
            }
template ()
String sq (string > (string s)
    return s+s
        }
```

```
int main(); {
    int i=5;
string str="ABC";
cout<<" square of integer :"<<sq(i)<<
    endl;
cout<<" square of string :"<<sq(str)<<
    endl;
}
```

O/P -

Square of Integer : 25
Square of String : ABCABC

Code for calculator

```
#include<class T>
class calc { T num1, num2;
Public:
    calc(Ta, Tb){
        num1=a;
        num2=b;
    }
void disp(){
    cout<<" Addition :"<<num1+num2;
    cout<<" Substraction :"<<num1-num2;
    cout<<" Multiplication :"<<num1*
        num2;
    cout<<" Division :"<<num1/num2;
    cout<<" Modulous :"<<num1%num2;
    cout<<" Square of num1 :"<<num1*
    num1;
    cout<<" Square of num2 :"<<num2
        * num2;
```

```
cout << " Cube of num2 : " << num2 * num2 * num2;
cout << " Cube of num2 " << " Average : " << (num1 + num2)/2;
cout << " Maximum : " << ((num1 > num2) ? num1 : num2);
cout << minimum : 2 << ((num1 > num2) ? num1 : num2;
    }
    }
int main(){
    int a, b
    cout << " Enter two integers : ";
        cin >> a >> b;
        calc Cint (a(ab);
        cout << " calculate : " << endl;
            calc . disp();
            }
```

O/P

Enter two integers : 10 5
Calculate Addition : 15
Substraction : 5
Multiplication : 50
Division : 2
Modulus : 0
Square of num 1 : 100
Square of num 2 : 25
cube of num 1 : 1000
Cube of num 2 : 125
Average : 7
Maximum : 10
Minimum : 5

```
#include <iostream>
using namespace std;
template <class T>
class Stack { T a[5]; int top;
public:
Stack () { top = -1; }
void push (T val) { if (top == 4) cout ("
        Stack overflow");
    else
    a[++top] = val;
    }
void pop () { if (top == -1) cout << "Stack
    underflow";
    else
cout << "popped " << a[top--] << endl;
    void disp () { cout "Stack elements :";
        for (int i = 0; i <= top; i++)
            cout << a[i] << " ";
            cout << endl;
                }
                };
int main () {
Stack <int> s;
s.push (20);
s.push (30);
s.display ();
s.pop ();
s.display ();
}
```

Experiment - 11

PAGE No.
DATE / /

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

int main() {
  vector<int> v = {1, 2, 3, 4, 5, 6, 7,
                   8, 9, 10};
  cout << "Initial vector :" << endl;
  for (int i = 0; i < 10; i++) {
  cout << v[i] << " " << endl;
    }
  cout << "Multiply by 10" << endl;
        for (int i = 0; i < 10; i++) {
            v[i] = v[i] * 10;
            }
        cout << " New vector " << endl;
            for << v[i] << " " << endl
            }

        return 0;
        }
```

O/P

Initial vector :

1 2 3 4 5 6 7 8 9 10

Multiply by 10

New vector :

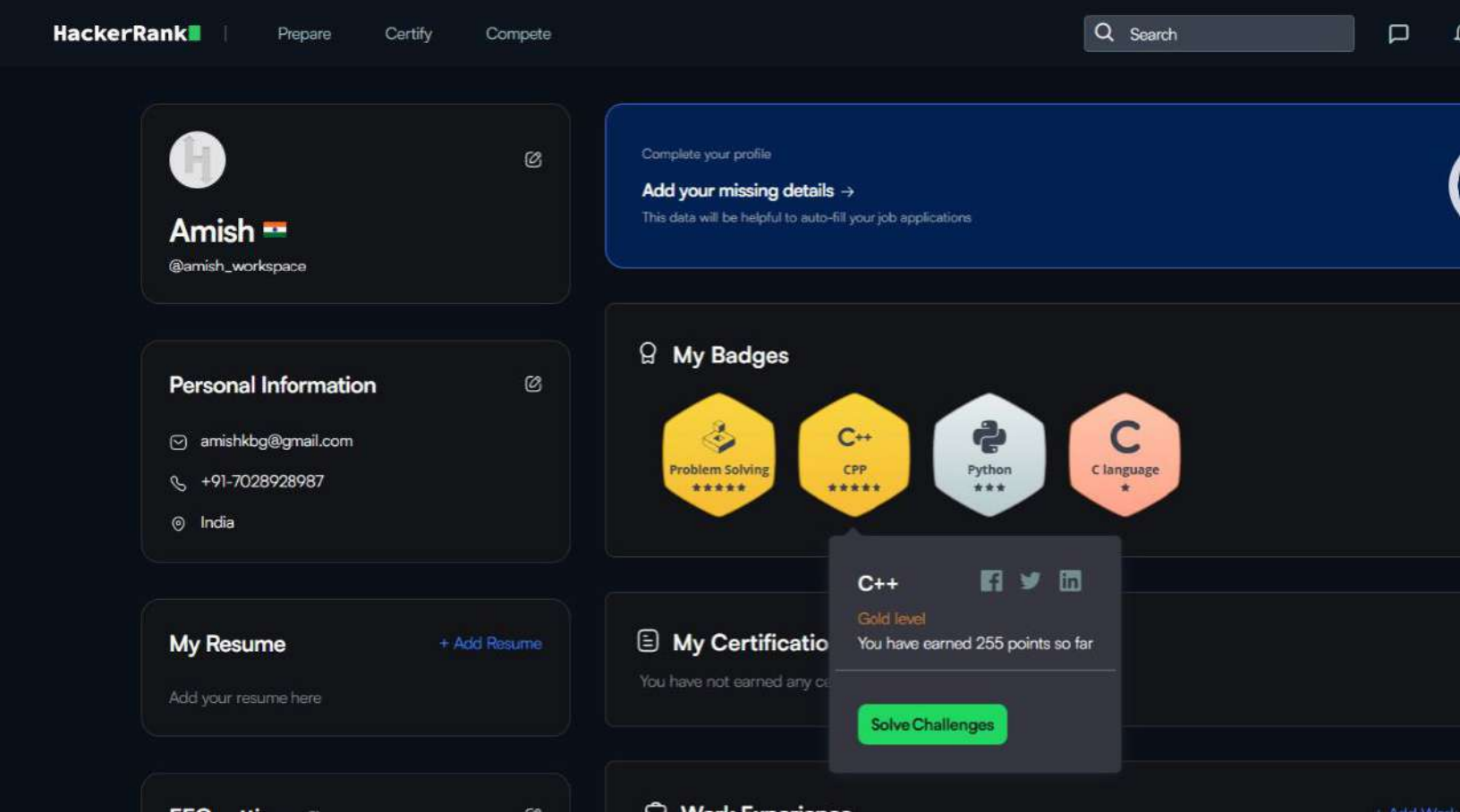
10 20 30 40 50 60

70 80 90 100

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

int main() {
    vector <int> v = {1, 2, 3, 4, 5, 6, 7
                      , 8, 9, 10};

    cout << " Initial vector : " << endl;
    for (auto it = v.begin(); it != v.
            end(); ++i) {
        cout << *it << " ";
    }
    cout << endl
    cout << " Multiplying each element
        by 10 ... " << endl;
    for (auto it = v.begin(); it != v.end
            (); ++i) {
        *it = (*it) * 10;
    }
    cout << " New vector after multipli
        -cation : " << endl;
        cout << *it << " ";
    }
    cout << endl;
    return 0;
}
```

Experiment - 12

```
#include <iostream>
#include <stack>
using namespace std;
stack (int) stack1;
void disp (){
stack (int) temp = stack1
while (!temp.empty()){
void disp(){
cout << temp.top() << " ";
temp.pop();
}
cout << endl;
}
int main(){
    int num;
    cout << "Enter a no:" << endl;
    cin >> num;
    for (int i = 0 ; i < num ; i++){
    int temp;
cout << Enter element at position" << i+1 <<
    endl;
    (in)) temp;
  Stack1. push(temp);
    }
cout << endl;
cout << " top most element" << stack
    top() << endl;
        cin >> temp;
```


HackerRank
Prepare
Certify
Compete
Search
Amish
@amish_workspace
Complete your profile
Add your missing details →
This data will be helpful to auto-fill your job applications
Personal Information
amishkbg@gmail.com
+91-7028928987
India
My Badges
Problem Solving
CPP
Python
C language
C++
Gold level
You have earned 255 points so far
Solve Challenges
My Resume
+ Add Resume
Add your resume here
My Certificatio
You have not earned any