PAGE No.
Sundaram
DATE 08/07/2025

★ Some programs in C++.

a) Write a C++ program to find the sum of 2 nos.

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

int main () {
    int a, b, sum;
    cout << " Enter 2 numbers: ";
    cin >> a >> b;
    sum = a + b;
    cout << "The sum of the two numbers is " << sum;
    return 0;
}.
```

Output:

Enter 2 numbers: 7 6

The sum of the two numbers is 13.

b) Write a C++ program to find the parity of a number.

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

int main () {
    int w;
    cout << " Enter a number: ";
    cin >> w;
    if (w % 2 == 0) {
    cout << w << " is even"; }
    else {
    cout << w << " is odd"; }.
    return 0;
}.
```

Output:

① Enter a number: 6
6 is even

② Enter a number: 17
17 is odd.

c) ~~Print~~ Write a C++ program to print the ~~to~~ first 10 numbers

① Using for loop

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

int main () {
  int i;
  cout <<"The first ten numbers are:\n";
  for (i=1; i<=10; i++) {
    cout<<" i "<<"\t";
  }
  return 0;
}
```

② Using while loop.

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

int main () {
  int i=1;
cout<<"The first ten numbers are:\n
while (i != 11) {
  cout << i <<"\t";
    i++;
}
return 0;
}.
```

Output:
The first ten numbers are:
1 2 3 4 5 6 7 8 9 10.

d) Write a C++ program to conduct arithmetic operations using switch

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

int main () {
   int a, b;
   cout << "Enter two numbers: ";
   cin ~~<<~~ >> a >> b;
```

~~while (true) {~~ char op;
cout << " Please enter the operation required (+, -, *, /): ";
cin >> op;

switch (op) {
case '+':
cout << "The sum is " << a+b;
break;
case '-':
cout << "The difference is " << a-b;
break;
case '*':
cout << "The product is " << a*b;
break;
case '/':
if (b!=0){
cout << "The quotient is " << a/b; }
else {
cout << "Error: zero division!"; }
break;
default:
cout << "Invalid symbol! Please use +, -, * or /.";
break;
}
return 0;
}.

Output:
Enter two numbers: 13 221
Please enter the operation required (+, -, *, /): +
The sum is 234.

e) Write C++ programs to display 5 rows of the following patterns.

① *
  * *
  * * *

② 1
  1 2
  1 2 3

③ 1
  2 2
  3 3 3

① #include <iostream>
using namespace std;

int main () {
int i, j;
for (i = 0; i <5; i++) {
for (j = 0; j <= i; j++) {
cout << "*";
}
cout << "\n";
}
return 0;
}

Output:
*
* *
* * *
* * * *
* * * * *

② #include <iostream>
using namespace std;

int main () {
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
```

③

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

int main () {
  int i,j;
  for (i=1; i<=5; i++) {
    for (j=1; j<=i; j++)
      cout << i;
    }
    cout << "\n";
  }
  return 0;
}
```

Output:

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
```

— x —

Experiment 1

a) WAP to declare a class 'Student' having data members roll, name. Accept & display data for one object.

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

class Student {
private:
    int roll;
    string name;
public:
    void accept() {
        cout << "Enter name: ";
        getline(cin, name);
        cout << "Enter Roll. no.: ";
        cin >> roll;
    }
    void display() {
        cout << "Name <<", Roll No. ", << roll;
    }
};

int main() {
    Student S1;
    S1.accept();
    S1.display();
    return 0;
}
```

b) WAP to declare a class 'Book' having data members bid, bname, bprice. Accept data for 2 books & display info of book having higher price.

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

class Book {
  private:
    string bname, bid;
  public:
    float bprice;
    void accept () {
      cout << "Enter Name: ";
      cin >> bname;
      cout << "Enter ID: ";
      cin >> bid;
      cout << "Enter price: ";
      cin >> bprice;
    }
    void display () {
      cout << "The book having a greater price is: \n";
      cout << "ID: " << bid << ", " << bname << ", Price: " << bprice;
    }
};

int main () {
  Book b1, b2;
  b1.accept();
  b2.accept();
```

```
    if (b2.bprice > b1.bprice) {
        b2.display();
    }
    else {
        b1.display();
    }
    return 0;
}
```

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

```
#include <stdio.h> <iostream>
using namespace std;

class Time {
  private:
    int hh, mm, ss;
  public:
    void accept () {
      cout << " Enter hours (HH): ";
      cin >> hh;
      cout << " Enter minutes (mm): ";
      cin >> mm;
      cout << " Enter seconds (ss): ";
      cin >> ss;
    }

    void convert () {
      int In cout << " The time in seconds is: " << (hh * 3600) + (mm * 60) + ss
                  << " seconds";
    }
};
```

```
int main () {
    Time T1;
    T1. accept();
    T1. convert();
    return 0;
}
```

——— x ———

Q.

Sundaram PAGE No: DATE: 18/07/2025

Experiment 2

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

```
#include <iostream>
#include <string>
using namespace std;
class City {
  private:
    string name;
  public:
    int popl;
    void getdata() {
      cout << "Enter City Name: ";
      getline(cin, name);
      cout << "Enter " << name << "'s population: ";
      cin >> popl;
    }
    void showdata() {
      cout << "The city having the largest population is " << name;
    }
} C[5];

int main() {
  int max = 0; mpop = 0;
  for (int i = 0; i < 5; i++) {
    C[i].getdata();
    cin.ignore(256, '\n');
    if (max < C[i].popl) {
      max = C[i].popl;
      mpop = i;
    }
}
```

```
c[mpop].showdata();
return 0;
}.
```

b) WAP to declare a class accountant having data members Account no. and balance. Accept this data for 10 accounts and give interest of 10% where balance >, 5000. Display.

```
#include <iostream>
using namespace std;
class Account {
    private:
        double balance;
        long long int acc_no;
    public:
     void getdata() {
        cout<<"Enter account number: ";
        cin>> acc_no;
        cout<<" Enter balance of acc no. "<<acc_no<<": ";
        cin>> balance;
     }
     void interest (){
        if (balance >= 5000) {
            balance += (balance * 10 * 1) /100;
        }
    }
    void showdata () {
      cout<<" Account no. "<< acc_no<<", Balance: "<<balance<<endl;
    }
} acc[10];
```

```
int main () {
    for (int i =0; i<10; i++) {
      acc[i].getdata ();
    } cin.ign
    for (int i=0; i<10; i++) {
      acc[i].interest();
      acc[i].showdata();
    }
  return 0;
}
```

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

```
#include <iostream>
#include<string>
using namespace std;
class staff {
    private:
      string name, post;
    public:
      void getdata () {
        cout<<"Enter Name: ";
        getline (cin, name);
        cout<<" Enter Post: ";
        getline(cin, post);
      }
      void showdata () {
        if (post == "Head of Department" || post=="Head of Dept." || post=="HOD
          || post =="Department Head") {
          cout<<name<<", Head of Dept."<<endl;
        }
    }
};st[5];
```

```
int main () {
  for (int i=0; i<5; i++) {
    st[i].getdata();
  }
  cout<<" Faculty Posts: "<<endl;
  for (int i = 0; i<5; i++) {
    st[i].showdata();
  }
  return 0;
}
```

— X —

25/7/25

Experiment 3

PAGE No:
DATE: 25/07/25

a) WAP to declare a class 'book' with data members book_title, author_name and price. Accept and display information for one object using pointer to object.

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

class Book {
    int price;
    string book_title, author_name;
public:
    void getdata() {
        cout << "Enter book name: ";
getline(cin, this->book_title); //using 'this->' instead of 'obj.'
        cout << "Enter author name: ";
getline(cin, this->author_name);
        cout << "Enter price: ";
        cin >> price;
    }

    void showdata() {
        this->getdata(); // calling the book::getdata() fn within showdata()
        cout << book_title << " by " << author_name << ". Price: " << price;
    }
} b1;

int main() {
    book *bp; //pointer to class book
    bp = &b1 //pointing bp to book b1.
    bp->showdata();
    return 0;
}
```

b) WAP to declare a class 'student' using data members roll_no & percentage. Use 'this' pointer to accept & display data for one object.

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

class student {
    int roll_no; ~~pertg~~
    ~~string name;~~ float pertg;
    public:
     void getdata() {
        cout << "Enter ~~name~~ pertg percentage: ";
        getline (cin, this -> pertg ~~name~~); // using this to point to '& pertg'
        cout << "Enter roll no.: ";
        cin >> roll_no;
     }
     void showdata() {
     this -> getdata(); // calling student:: getdata() via reference.
     cout << ~~name~~ ". Roll no. " << roll_no << ". Percentage: " << pertg;
     }
};

int main() {
   b1.showdata();
   return 0;
}
```

c) WAP to demonstrate the use of nested classes.

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

class Enclosing {
    int n = 13;
    public:
    void showData () {
        cout << " Calling from the enclosing class, n = " << n << endl;
    }

    class Nested {
        int n = 37;
        public :
        void showDataInner () {
            cout << " Calling from the nested class, n = " << n << endl;
        }
    };
};

int main () {
    Enclosing e;
    Enclosing :: Nested n;
    e. showData ();
    n. showDataInner ();
    return 0;
}
```

— x —

2
11/8

Experiment 4

PAGE No.
DATE 08/08/2025

a) WAP to swap two numbers from the same class using object as function argument. Ensure that swap is member.

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

class Num {
    int x, y;
    public:
    void accept() {
        cout << "Enter First Number: ";
        cin >> x;
        cout << "Enter Second Number: ";
        cin >> y;
    }
    void display () {
        cout << "First Number: " << x << endl;
        cout << "Second Number: " << y << endl;
    }
    void swap (Num &n) {
        int temp = n.x;
        n.x = n.y;
        n.y = temp;
    }
};

int main() {
    n.accept();
    n.display();
    cout << "\nSwapping Numbers...\n\n";
    n.swap(n);
    n.display();
    return 0;
}
```

b) WAP to swap 2 numbers of the same class using friend function.

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

class Num {
    int a, b
    public:
    void getdata () {
        cout << "Enter first number: ";
        cin >> a;
        cout << "Enter second number: ";
        cin >> b;
    }
    void showdata () {
        cout << "First number: " << a << endl;
        cout << "Second number: " << b << endl;
    }
    friend void swap (Num &n);
};
void swap (Num &n) {
    int temp = n.a;
    n.a = n.b;
    n.b = temp;
}
int main () {
  n.getdata ();
  n.showdata ();
  cout << "\nSwapping numbers...\n\n";
  swap (n);
  n.showdata ();
    return 0;
}
```

c) WAP to swap 2 nos. of different classes using friend function.

```
#include <iostream>
using namespace std;
class B;
class A {
    int x;
    public:
    void getdata() {
        cout << "Enter first number: ";
        cin >> x;
    }
    void showdata() {
        cout << "First Number: " << x << endl;
    }
    friend void swap(A &a, B &b);
};

class B {
    int x;
    public:
    void getdata() {
        cout << "Enter Second Number: ";
        cin >> x;
    }
    void showdata() {
        cout << "Second Number: " << x << endl;
    }
    friend void swap(A &a, B &b);
};
```

```
void swap(A &a, B &b) {
    int temp= a.x;
    a.x = b.x;
    b.x = temp;
}

int main () {
    n1.getdata();
    n2.getdata();
    n1.showdata();
    n2.showdata();
    cout<<"\nSwapping Numbers...\n\n";
    swap(n1, n2);
    n1.showdata();
    n2.showdata();
    return 0;
}
```

O/P.

```
Enter First Number: 21
Enter Second Number: 71
First Number: 21
Second Number: 71

Swapping Numbers...

First Number: 71
Second Number: 21.
```

* O/P for a), b) and c) is exactly the same.

d) WAP to find the greatest no. from 2 separate classes. Use friend functions.

```
#include <iostream>
using namespace std;
class A;
class B {
  int b;
  public:
  void getdata() {
    cout<<"Enter Second Number: ";
    cin>>b;
  }
  friend string findGreater(A &a, B &b);
};

class A {
  int b;
  public:
  void getdata() {
   cout<<"Enter First Number: ";
   cin>>b;
  }
  friend string findGreater(A &a, B &b);
};

string findGreater(A &a, B &b) {
   if(a.b == b.a) {
     return "Both numbers are equal";
   }
   else if(a.b > b.a) {
     return "The first number is greater";
   }
   else {
     return "The second number is greater";
   }
```

```
int main () {
    a.getdata ();
    b.getdata ();
    ~~findGreater(a,b);~~
    string isGreater = findGreater(a,b);
    cout << isGreater;
    return 0;
}
```

O/P.

① Enter First Number: 6
Enter Second Number: 17
The second number is greater

② Enter First Number: 17
Enter Second Number: 6
The first number is greater

e) WAP to create 2 classes, Result1 and Result2 which stores the marks of a student. Read the value of marks for both classes & compute the average using friend functions.

```
#include <iostream>
using namespace std;
class Result2;
class Result1 {
    ~~int~~ float R1;
  public:
    void getdata() {
        cout << "Enter marks for subject 1: ";
        cin >> R1;
    }
    friend float getAvg(Result1 &a, Result2 &b);
};

class Result2 {
    ~~int R2;~~ float R2;
```

```
public:
void getdata () {
  cout << "Enter marks for subject 2: ";
  cin >> R2;
}
friend float getAvg (Result1 &a, Result2 &b);
};

float getAvg (Result1 &a, Result2 &b) {
    float avg = (a.R1 + b.R2)/2;
    return avg;
}

int main () {
  r1.getdata();
  r2.getdata();
  float avgr = getAvg(r1, r2);
  cout << "Average marks of the student is " << avgr;
  return 0;
}
```

O/P

Enter marks for subject 1: 97.5

Enter marks for subject 2: 98.2

Average marks of the student is 97.85

11/8/25

Experiment 5

PAGE No:
DATE: 03/09/25

a) WAP to find the sum of numbers 1 to n where n is a variable passed to the constructor.

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

class w {
    int n;
public:
    w() {    // Default constructor
        int i, sum=0;
        n= 10;
        for (i=1; i<=n; i++) {
            sum += i;
        }
        cout << "The sum is " << sum << endl;
    }
    w(int num) {    // Parameterized constructor
        int i, sum=0; n=num;
        for (i=1; i<=n; i++) {
            sum += i;
        }
        cout << "The sum is" << sum << endl;
    }
    w(w &obj1) {    // Copy Constructor
        n = obj1.n;
        int i, sum=0;
        for (i=1; i<=n; i++) {
            sum += i;
        }
        cout << "The sum is " << sum << endl;
    }
};
```

```
int main () {
    ω obj 1;    // Default Constructor invocation
    ω obj2(5); // Parameterized Constructor invocation
    ω obj       // Copy Constructor invocation.
    return 0;
}.
```

b) Write a program to declare a class "student" having data members name & percentage. Write constructors to initialize these data members & accept & display for 1 student

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

class student {
    string name;
    float percentage;
    public:
        student () {
            name = "John Doe";
            percentage = 97.4;
            cout << "Name: " << name << ", " << percentage << "%." << endl;
        }
        student (string n, float p) {
            percentage = p;
            name = n;
            cout << "Name: " << name << ", " << percentage << "%." << endl;
        }
        student (student &obj1) {
            percentage = obj1.percentage;
            name = obj1.name;
            cout << "Name: " << name << ", " << percentage << "%." << endl;
        }
};
```

```
int main () {
  student s1;
  student s2("John Joe", 47.9);
  student s3 (student s2);
  return 0;
 }
```

o/p:

Name: John Doe, 97.4%.
Name: John Joe, 47.9%.
Name: John Joe, 47.9%.

c) ~~Def~~ WAP to define a class college with data members rollno, name & course. Write a constructor with default value "Computer Engineering" for course. Accept & display data for objects.

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

class student {
    int rollno;
    string name, course;
    public:
    student () {
      rollno = 13 ;
      name = "Pallavi";
      course: "computer engineering";
    cout<< name<<", roll no. "<< rollno <<", "<<course<<endl;
    }.
    student (int rn, string n, string c: "computer engineering")
    {
      rollno = rn;
      name = n;
      course = c;
```

```
cout << name << ", rollno. " << rollno << ", " << course << endl;
}
student (student & obj1) {
    name = obj1.name;
    rollno = obj1.rollno;
    course = obj1.course;
    cout << name << ", roll no. " << rollno << ", " << course << endl;
}
};

int main () {
    student s1;
    student s2 (13, "Pallavi");
    student s3 (13, "Pallavi", "compsci");
    student s4 (student s3);
    return 0;
}
```

O/P:

Pallavi, roll no. 13, completer engineering
Pallavi, roll no. 13, computer engineering
Pallavi, roll no. 13, compsci
Pallavi, roll no. 13, compsci.

d) WAP to display constructor overloading.

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

class num {
    int n;
public:
    num () {  //Default constructor.
        n = 13;
        cout << "The number is " << n << endl;
    }
```

```
num(int num) { // Parameterized constructor.
    n = num;
    cout << "The number is " << n << endl;
}
num(num &obj1) {
    n = obj1.n;
    cout << "The number is " << n << endl;
}
};

int main() {
    num n1;
    num n2(14);
    num n3(n1);
    return 0;
}
```

O/P:

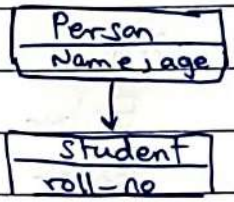
The number is 13.
The number is 14.
The number is 13.

26/9/25

Experiment 6

Sundaram PAGE No: DATE: 10/09/25.

a) WAP to implement the following inheritance & display all details of student.

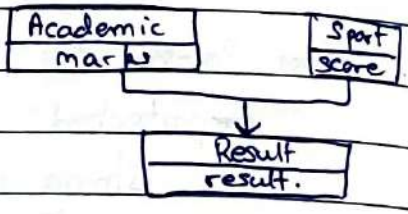



```
#include <iostream>
using namespace std;

class Person {
    protected:
        string name;
        int age;
};

class Student: protected Person {
    private:
        int roll-no;
    public:
        void showdata () {
            cout << name << ", Age " << age << ", Roll no. " << roll-no;
        }
        void getdata () {
            cout << "Enter Name: ";
            cin >> name;
            cout << "Enter Age: ";
            cin >> age;
            cout << "Enter Roll no: ";
            cin >> roll-no;
        }
};

int main () {
    std.getdata();
    Std.showdata();
    return 0;
}
```

O/P:

Enter Name: Pallavi
Enter Age: 18
Enter Roll no: 71
Pallavi, Age 18, Roll no 71.

b) WAP to implement the following inheritance & write a function to calculate score & display data.



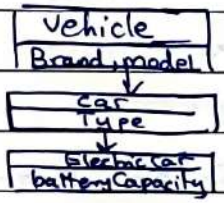

```
#include <iostream>
using namespace std;

class Academic {
    protected:
        int marks;
};
class Sport {
    protected:
        int score;
};
class Result: protected Academic, protected Sport {
    int result;
    public:
    void getdata(){
        cout<<"Enter marks: ";
        cin>>marks;
        cout<<"Enter score (sports). If none, enter 0: ";
        cin>>score;
    }
    void results(Result &R1){
        R1.result = R1.score + R1.marks;
        cout<<" Overall result: "<<R1.result;
    }
};
```

```
int main () {
    o1. getdata ();
    o. results (o);
    return 0;
}
```

o/p:

Enter marks: 37
Enter score (sports). If none, enter 0 : 73
Overall result: 110.

c) WAP to implement the following inheritance & write a function to display data.



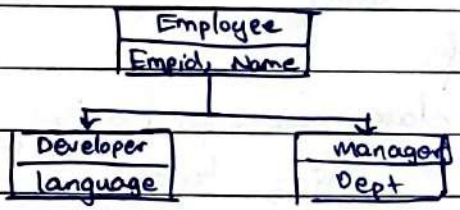

```
#include <iostream>
using namespace std;

class Vehicle {
protected:
    string brand, model;
};
class Car : protected Vehicle {
protected:
    string type;
};
class electricCar : protected Car {
    int batteryCapacity;
    public:
        void getdata () {
            cout << "Enter Brand: ";
            getline (cin, brand);
            cout << "Enter Model: ";
            getline (cin, model);
            cout << "Enter Type: ";
            getline (cin, type);
            cout << "Enter battery capacity in kWH: ";
            cin >> batteryCapacity; }
```

```
void showdata () {
    cout<< brand <<" "<< name <<", Type: "<< type<<", Battery Capacity:
        << batteryCapacity <<" kWh";
    }
};

int main () {
    s.getdata();
    s.showdata();
    return 0;
}
```

O/P:

Enter Brand: ~~Audi~~ Chevrolet
Enter Model: Blazer
Enter Type : & EV.
Enter Battery Capacity in kWh: 66.
Chevrolet Blazer, Type: EV, Battery Capacity: 66 kWh.

d) WAP to implement the following inheritance & display details for both classes.




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

class Employee {
protected:
    int empid;
    string name;
};

class developer : protected employee {
    string language;
```

```
public:
void getdata () {
    cout << "Enter Name: ";
    getline (cin, name);
    cout << "Enter empid: ";
    cin >> empid;
}
```

cout << "Enter programming language: ";
cin >> language;

```
void showdata () {
    cout << Name << ", id: " << empid << ", programming language: " << language << endl;
}.
};d;
class manager : protected Employee {
    sting dept;
    public:
    void getdata () {
        cout << "Enter Name: ";
        getline (cin, name);
        cout << " Enter empid: ";
        cin >> empid;
    }
```

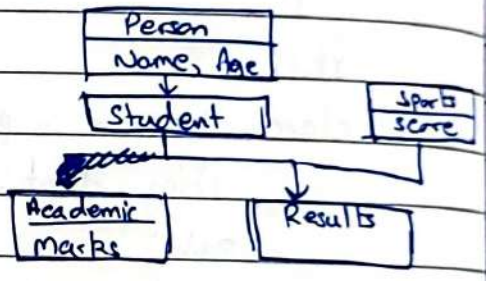
cout << "Enter department: ";
cin >> dept;

```
    void showdata () {
        cout << Name << ", id: " << empid << ", oversees " << dept << " department"
            << endl;
    }.
};m;

int main () {
    d. getdata();
    d. showdata();
    m. getdata();
    m. showdata();
    return 0;
}.
```

O/p:

Enter Name: Pallavi
Enter empid: 123746
Enter programming language: c++.
Pallavi, id: 123746, programming language: c++.
Enter Name: Pallavi2
Enter empid: 647321
Enter department: c++
Pallavi2, id: 647321, oversees c++ department.

c) WAP to implement the following inheritance & display total marks, sports score & student details.



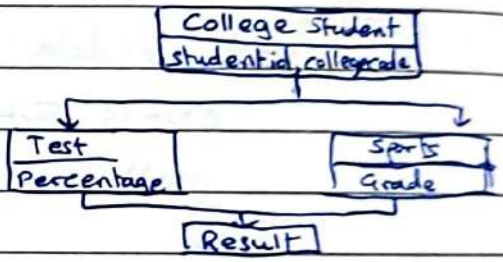

```
#include <iostream>
using namespace std;

class Person {
    protected:
    string name;
    int age;
};
class Student : protected Person {
    protected:
        int sid;
};
class Academic : protected Student {
    public:
        void getdata () {
            cout << "Enter name: ";
            getline(cin, name);
            cout << "Enter age : ";
            cin >> age;
            cout << "Enter marks: ";
            cin >> marks;
        }
```

```
    void showdata () {
        cout<<"Student Details:\n"<<endl;
        cout<<" Name: "<< name<<endl;
        cout<<" Age: "<<age<< endl;
        cout<<" Academic Record: "<<endl;
        cout<<" Marks: "<< marks<<endl;
    }
};a;
class Sport {
    protected:
        int score;
};
class Result : protected Sport, protected Student {
    void getdata () {
        cout<<"Enter sports score: ";
        cin>> score;
    }
    void showdata () {
        cout<<"Sports Record: "<<endl;
        cout<<"Score: "<< score <<endl;
    }
};s;

int main () {
    a.getdata();
    s.getdata();
    a.showdata();
    s.showdata();
    return 0;
}
```

O/p:

Enter Name: Pallavi
Enter Age: 18
Enter Marks: 83
Enter sports score: 137.
Student Details:
Name: Pallavi
Age: 18
Academic Record:
Marks: 83
Sports Record:
Score: 137.

f) WAP to implement the following inheritance & accept/display data for one object.



Note: Use virtual base class.

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

class College-Student {
    protected:
        int studentid, collegecode;
};
class Test: protected virtual College-Student {
    protected:
        int percentage;
};
class Sports: virtual protected College-student {
    protected:
        string grade;
};
```

```
class Result : protected Test, protected Sports {
    public:
     void getdata () { {
        cout << "Enter student id: ";
        cin >> studentid;
        cout << "Enter college code: ";
        cin >> collegecode;
        cout << "Enter test percentage: ";
        cin >> percentage;
        cout << "Enter sports grade: ";
        cin >> grade;
      }
     void showdata () {
        cout << "Student details: " << endl;
        cout << " Id: " << studentid << endl;
        cout << "College: " << collegecode << endl;
        cout << "Percentage: " << percentage << endl;
        cout << "Grade: " << grade << endl;
      }
  };

int main () {
   r. getdata ();
   r. showdata ();
   return 0;
}.
```

o/p:

```
Enter student id: 1274167281
Enter college code: 317439
Enter test percentage: 67
Enter sports grade : A
```

Student details:

Id: 1274167281

College: 317439.

Percentage: 67

Grade: A

26/9/25

Experiment 7

PAGE No:
DATE: 26/09/25

a) WAP using function overloading to calculate the area of a square classroom & rectangular laboratory.

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

class b {
  public:
    void area (int a) {
      cout<<"The area is "<<a*a<<endl;
    }
    void area (int a, int b) {
      cout<<"The area is "<<a*b<<endl;
    }
}b1;

int main() {
  d.area (14);      // Function to calculate area of square
  d.area (10,30);   // Function to calculate area of rect".
  return 0;
}
```

O/P:

The area is 196
The area is 300.

b) WAP using function overloading to calculate the sum of 5 float values & 10 integer values.

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

```
class c {
    public:
        void sum (int arr[], int size) {
            int i; sum=0;
            for (i=0; i< size; i++) {
                sum += arr[i];
            }
            cout << "Sum is " << sum << endl;
        }
        void sum (float arr[], int size) {
            int i;
            float sum=0;
            for (i=0; i<size; i++) {
                sum += arr[i];
            }
            cout << "Sum is " << sum << endl;
        }
};

int main () {
    int arr1[10] = {10, 20, 30, 40, 50, 50, 40, 30, 20, 10};
    float arr2[5] = {13.7, 12.6, 17.9, 19.8, 15.5};
    c1.sum (arr1, 10);
    c1.sum (arr2, 5);
    return 0;
```

O/P:

Sum is 300

Sum is 79.5

c) WAP to implement '-' operator (unary) when used, the numeric data is negated.

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

class d {
    int a;
  public:
    void operator - () {
        a = -a;
    }
    void getdata () {
        cout << " Enter Number: ";
        cin >> a;
    }
    void showdata () {
        cout << "New value: " << a;
    }
};

int main () {
    d1.accept ();
    -d1;
    d1.display ();
}
```

O/P:

Enter Number: 17

New Value: -17.

d) WAP to implement '++' operator (unary) so when used the numeric data is increment.

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

class e {
    int a;
    public:
      void accept () {
          cout<<" Enter Number : ";
          cin>>a;
      }
      void display () {
          cout<<"New value: "<<a;
      }
```

```
    friend void operator ++ (e &a);
    friend void operator ++ (e &a, int);
};

void operator ++(e &a) {
    ++ ~~ob~~ a.a;
}
void operator ++(e &a, int) {
    a.a++;
}

int main () {
    e1. accept ();
    ++ e1;
    e1. display();
    e1++;
    e1. display();
    return 0;
}.
```

O/P:

Enter Number: 137
New value: 138
New value: 139.

26/9/25

Experiment 8

a) WAP to overload the binary + operator so that two strings are concatenated.

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

class w {
    string x;
    public:
        void getdata() {
            cout << "Enter String: ";
            getline (cin, a);
        }
        string operator + (w &w & x) {
            return x+x.x;
        }
};w1, w2;

int main () {
    w1. getdata();
    w2. getdata();
    cout << "Concatenated String = " << w1+w2;
    return 0;
}
```

O/P:

Enter String: Shilpa

Enter String: Shitole

Concatenated String = Shilpa Shitole.

b) WAP to create base class ILogin with datamembers Name & Password. Declare getdata() as a virtual. Derive EmailLogin & MembershipLogin classes from ILogin. Display EmailLogin details & MembershipLogin Details of the employee.

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

class ILogin {
  protected:
    string name, pwd;
    virtual void getdata()=0;
};

class EmailLogin: protected Ilogin {
      string Email;
  public:
    void getdata() {
        cout<<"Name: ";
        getline(cin, Name);
        cout<<"Email: ";
        getline(cin, Email);
        cout<<"Password: ";
        getline(cin, pwd);
    }
    void showdata() {
        cout<<"\n Email LoginInName: "<<name<<"\nEmail: "<<
            Email<<"\nPassword: "<<pwd<<endl;
    }
};x;
```

```
class MembershipLogin : protected ILogin {
    string MID;
    public:
      void getdata () {
          cout<<"Name: ";
          getline (cin, Name);
          cout<<"Membership ID: ";
          getline (cin, MID);
          cout<<"Password: ";
          getline (cin, pwd);
      }
      void showdata() {
        cout<<"\nMembershipLogin\nName: "<<Name<<"\nMembershipID
            <<MID<<"\nPassword: "<<pwd<<endl;
      }
} x1;

int main () {
    cout<<"Enter EmailLogin: "<<endl;
    x.getdata();
    cout<<"Enter MembershipLogin: "<<endl;
    x1.getdata();
    x.showdata();
    x1.showdata();
    return 0;
}
```

O/p:

Enter Email login details:

Name: Shilpa Shitole

Email: shilpa.shitole@hotmail.com.

Password: Abcdef123456

Enter Membership login details:

Name: Shilpa Shitole2.

Email: shilpa Membership ID: a123b234c345dfeg.

Password: Abcdef123456

EmailLogin

Name: Shilpa Shitole

Email: shilpa.shitole@hotmail.com

Password: Abcdef123456

MembershipLogin

Name: Shilpa Shitole2

Membership ID: a123b234c345dfeg.

Password: Abcdef123456

28/9/25

Experiment 9.

a) WAP to copy the contents of one file to another. Open the first file in read & second file in out mode. Copy contents of first file in second file. Assume second file is already created.

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

int main () {
  ifstream one;
  one.open("first.txt");
  ofstream two;
  two.open("second.txt");

  if (!one || !two) {cout << "Error"; return -1;}
  char ch;
  while (!one.eof()) {        while (one.get(ch)) {
    one >> ch;                    two.put(ch);
    two << ch;
  }
  cout << "Text copied successfully!";
  one.close();
  two.close();
  return 0;
}
```

Output: Text copied successfully!

b)/c)/d) Write a WAP to count digits, spaces, words and occurrences of a word in a file.

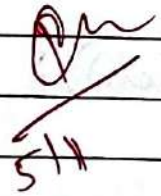
```
#include<iostream>
#include<fstream>          #include<cctype>
using namespace std;

int main () {
  ifstream tonk;
   tonk.open("test.txt");

  if (!tonk) {
     cout<<"Error. File Corrupted or Non-existent";
     return -1;
  }
  char ch;
  int space=0, digit=0, wcount=0, cwcount=0;
  string word; w;
  cout<<"Enter a word which's occurence must be counted"<<
  endl;
  cin>> word;
  while (tonk.get(ch)) {
     if (isdigit(ch)) {
        digit++;
     }
  else if (ispunct(ch)) { };
  else if (isalpha(ch)){
        w += ch;
     }
     if (isspace(ch)) {
        space++;
        if (w != "") {
        wcount++;
           if (w == word) {
              cwcount++;
           }
        }
  }
```

```
    }
    if (w != " ") {
      wcount++;
      if (w == word) {
        cwcount++;
      }
    }

cout << "Spaces: " << space << " Digits: " << digit << " words: "
  << wcount << " Occurences of " << word << ": " << cwcount << endl;
  return 0;
}
```

o/p:

Spaces: 17 Digits: 31 words: 18 Occurrences of banana: 3

—x—



Experiment 10

PAGE No:
DATE 17/10/25

a) WAP to find the sum of an array using a function template.

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

template <class T> T arrSum (T arr[], int length) {
  int i;
  T sum=0;
  for (i=0; i<length; i++) {
    sum += arr[i];
  }
  return 0;
}

int main () {
  int intarr[10] = {14, 16, 17, 43, 31, 14, 16, 38, 58, 68};
  float floatarr[10] = {12.6, 1.8, 47.8, 43.7, 57.2, 41.9, 99.6, 86.1, 103.2, 57};
  double doubarr[10] = {1, 2, 3, 4, 5, 6, 7, 8, 1, 1};

  cout << "Sum of intarr: " << arrSum (intarr, 10) << endl;
  cout << "Sum of floatarr: " << arrSum (floatarr, 10) << endl;
  cout << "Sum of doubarr: " << arrSum (doubarr, 10) << endl;
  return 0;
}
```

O/P:

Sum of intarr: 313
Sum of floatarr: 560.9
Sum of doubarr: 38

b) WAP to find square of int & string using function temp

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

template <class T> T sq(T x) { return x*x; }
template <> string sq<string>(string s) { return (s+s); }

int main () {
  int i = 4;
  cout << i << " => " << sq(i) << endl;
  string w = "uvula";
  cout << w << " => " << sq(w) << endl;
  return 0;
}
```

o/p:

4 ⇒ 16

uvula ⇒ uvulauvula.

c) WAP to build a calculator using a class template

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

template <class T> class c {
    T n1, n2;
    public:
      c(T x, T y) {
        n1 = x;
        n2 = y;
      }
    void calc () {
      cout << "Sum = " << n1+n2 << endl;
```

```
        cout << "Difference = " << n1-n2 << endl;
        cout << "Product = " << n1 * n2 << endl;
        cout << "Qubient = " << n1/n2 << endl;
        cout << "Remainder = " << n1 % n2 << endl;
    }
};

int main() {
    c o(12, 17);
    o.calc();
    return 0;
}
```

o/p:

Sum = 29

Difference = -5

Product = 204

Quotient = 0

Remainder = 12.

d) WAP to implement push & pop operations in stack template

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

template<class T> class stack {
    int top, arr[10];
    public:
    stack() { top = -1; }
    void push(int val) {
        if (top == 9) {
            printf("Error, stack overflow");
            return;
        }
        arr[++top] = val;
```

```
cout << "Pushed " << val << " on stack." << endl;
return;
}

void pop () {
    if (top == -1) {
        printf("Error, Stack Underflow");
        return;
    }
    int val = arr[top--];
    cout << "Popped " << val << " from stack." << endl;
    return;
}
void trav () {
    int i;
    for (i=0; i<=top; i++) {
        cout << arr[i];
        if (i==top<top) {
            cout << " → ";
        }
    }
    cout << endl;
}

int main () {
    stack<int> st;
    st.push (21);
    st.push (31);
    st.push (57);
    st.pop ();
    st.push (67);
```

```
st.push(69);
st.push(74);
st.pop();
st.push(81);
st.push(94);
st.push(98);
st.push(103);
st.pop();
st.traw();
return 0;
}
```

O/P:

Pushed 21 on stack
Pushed 31 on stack
Pushed 57 on stack
Popped 57 from stack
Pushed 67 on stack
Pushed 69 on stack
Pushed 74 on stack
Popped 74 from stack
Pushed 81 on stack
Pushed 94 on stack
Pushed 98 on stack
Pushed 103 on stack
Popped 103 from stack.
21 → 31 → 57 → 69 → 81 → 94 → 98

Experiment 11

a) WAP to create, modify & print vector using stl & multiply its elements by a scalar.

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

void print (vector<int> vec) {
    int i;
    for (i=0; i<vec.size(); i++) {
      if (i==0) { cout << "(" << vec[i] << ";}
      else if (i== vec.size()-1) { cout << ", " << vec[i] << ")" << endl; }
      else { cout << ", " << vec[i]; }
    }
    return;
}

int main () {
    vector <int> v(10);
    int i;
    cout << "Created vector: ";
    print(v);
    for (i=0; i< v.size(); i++) {
      v[i] = (i+1)*10;
    }
    cout << "Added Elements: ";
    print(v);
    for (i=0; i<4; i++) {
      v.push_back((11+1)*10);
    }
    cout << "Modified Vector: ";
    print(v);
```

```
for (i=0; i<v.size(); i++) {
    v[i] *= 21;
}
```

```
for (auto it = v.begin(); it != v.end(); i++) {
    *it *= 21; }
```

```
cout << "Multiplied Elements by Scalar: ";
print(v);
return 0;
}.
```

O/p:

Created vector: (0, 0, 0, 0, 0, 0, 0, 0, 0, 0)

Added Elements: (10, 20, 30, 40, 50, 60, 70, 80, 90, 100)

Modified vector: (10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, 120, 130, 140)

Multiplied Elements by Scalar: (210, 420, 630, 840, 1050, 1260, 1470, 1680, 1890, 2100, 2310, 2520, 2730, 2940);

5/11

Experiment 12

a) ~~Imp~~ WAP to implement stack with STL.

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

int main () {
  stack <int> st;
  st.push (13);
  st.push (27);
  st.push (39);
  st.pop ();
  st.push (43);
  st.push (56);
cout<<(st.top ()<<endl)
  ~~st.pop();~~
  st.pop();
  st.pop();
cout<<(st.top ()<<endl;
  st.push (33);
  ~~return 0;~~ while (!st.empty()) {
        cout << st.top ()<<endl;
        st.pop(); }
  cout <<"Stack Emptied.";
  return 0;
}
```

cout<<"The stack now is: "<<endl;

O/P: 56
27
The stack now is:
33
27
13
Stack emptied.

b) WAP to implement queue using stl

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

int main () {
    queue <int> q;
    q.push (13);
    q.push (27);
    q.push (17);
    q.push (31);
    q.push (47);
    cout <<"Size of queue: " << q.size()<<endl;
    cout <<" Front: " << q.front()<<endl;
    cout <<" Rear: " << q.back()<<endl;
    q.pop ();
    q.pop ();
    q.push (54);
    q.push (63);
    q.push (71);
    q.pop ();
    q.pop();
    cout <<" Size of queue : " <<q.size() <<endl;
    cout <<" Front: " <<q.front() <<endl;
    cout <<" Rear! " <<q.back() <<endl;
    return 0;
}
```

O/P: Size of queue : 5
Front: 13
Rear : 47
Size of queue: 4
Front: 47
Rear: 71.

c) WAP to implement searching & sorting with user defined records, such as Person-Record (Name, dob, phno) or Item_Record (item_name, quantity, cost).

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

struct Item {
    int itemcode;
    int qty;
    string name;
    float cost;
};

int main () {
  vector <Item> it;
  int n,i;
  cout <<" Enter item no: ";
  cin>> n;
  cout<<" Enter item list: "<<endl;
  for (i = 0; i< n; i++) {
    Item a;
    cout <<"Item "<< i+1<<":\n";
    cout <<"Item Code: ";
    cin>> a.itemcode;
    cout<<"Item Name: ";
    cin>> a.name
    cout<<"Item Qty: ";
    cin>>a.qty;
    cout<<"Item Cost: ";
    cin>>a.cost;
    it.push_back(a);
}
```

```
sort(it.begin(), it.end(), [](Item a, Item b) {
    return a.name > b.name; });
cout << "Items sorted by name:\n";
for (i=0; i < it.size(); i++) {
    cout << it[i].itemcode << "\t" << it[i].Name << "\t" << it[i].qty <<
    "\t" << it[i].cost << endl; }

int code;
cout << "\nEnter item code: ";
cin >> code;
auto found = find_if(it.begin(), it.end(), [code](Item i){
        return i.itemcode == code;
});

if (found != it.end()) {
    cout << "\nItem Found: " << found->itemcode << "\t" << found->name
    << "\t" << found->qty << "\t" << found->cost << endl;
}
else { cout << "Item not found.\n" << endl; }
return 0;
```

o/p:

Enter itemlist no: 3

Enter itemlist:

Item 1:

Item Code : 101

Item Name : Toothbrush

Item Qty: 3 ← Item Cost: 130

5/1/1

Item 2:

Item Code: 103

Item Name: Paste

Item qty: 4

Item Cost: 210