

Experiment 1

Create a class namely Employee array with members Employee No, Employee Name, Basic, and two classes Gross and Deduct. The Gross has members DA, HRA, TA and Deduct has members PF and IT. The calculations for DA (50% of Basic), HRA (30% of Basic), TA (RS. 800 fixed), PF (5% of Basic), IT (30% of gross if total income per annum exceeds 1 lakh otherwise nil). Write a C++ Program to calculate the net salary of the Employee and display all the details.

```
#include <iostream>
#include <cstdio>
#include <string>
using namespace std;
class Employee {
    int employee_number;
    string employee_name;
    float employee_basic;
public :
    Employee(int number, string name, float basic) {
        this -> employee_number = number;
        this -> employee_name = name;
        this -> employee_basic = basic;
    }
    string getName() {
        return this -> employee_name;
    }
    int getNumber() {
        return this -> employee_number;
    }
    float getBasic() {
        return this -> employee_basic;
    }
    class Gross {
    public :
        float calculateDA(float basic) {
```

```

        return (0.5 * basic);
    }
    float calculateHRA(float basic) {
        return (0.3 * basic);
    }
    float getTA() {
        return 800.0;
    }
};

class Product {
    public :
        float calculatePF(float basic) {
            return (0.05 * basic);
        }
        float calculateIT(float gross) {
            return (0.3 * gross);
        }
};

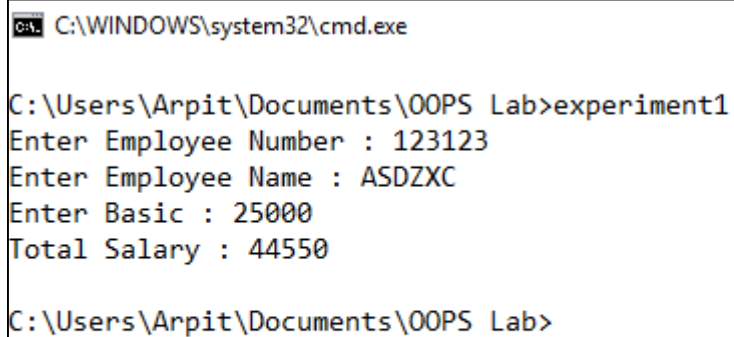
void calculateSalary() {
    Gross gross;
    Product product;
    float basic = this -> getBasic();
    float total = basic;
    total += gross.calculateDA(basic) +
gross.calculateHRA(basic) + gross.getTA();
    total -= product.calculatePF(basic);
    if(total >= 100000)
        total -= product.calculateIT(total);
    cout << "Total Salary : " << total << "\n";
}

};

int main() {
    ios_base::sync_with_stdio(0);
    string name;

```

```
int number;  
float basic;  
cout << "Enter Employee Number : " ;  
cin >> number;  
cout << "Enter Employee Name : " ;  
cin >> name;  
cout << "Enter Basic : " ;  
cin >> basic;  
Employee employee(number, name, basic);  
employee.calculateSalary();  
}
```



The screenshot shows a Windows command prompt window titled "C:\WINDOWS\system32\cmd.exe". The user is in the directory "C:\Users\Arpit\Documents\OOPS Lab" and has run the command "experiment1". The program prompts for three inputs: "Enter Employee Number : 123123", "Enter Employee Name : ASDZXC", and "Enter Basic : 25000". After processing, it displays the output "Total Salary : 44550". The prompt "C:\Users\Arpit\Documents\OOPS Lab>" is visible at the bottom.

```
C:\WINDOWS\system32\cmd.exe  
  
C:\Users\Arpit\Documents\OOPS Lab>experiment1  
Enter Employee Number : 123123  
Enter Employee Name : ASDZXC  
Enter Basic : 25000  
Total Salary : 44550  
  
C:\Users\Arpit\Documents\OOPS Lab>
```


Experiment 2

Write a C++ program to overload + operator to add two complex numbers.

```
#include <iostream>
using namespace std;
class complexNumber {
    float real, imaginary;
public :
    complexNumber(float real, float imaginary) {
        this -> real = real;
        this -> imaginary = imaginary;
    }
    complexNumber operator+(const complexNumber b) {
        float real = this -> real + b.real;
        float imaginary = this -> imaginary + b.imaginary;
        return complexNumber(real, imaginary);
    }
    float getReal() {
        return this -> real;
    }
    float getImaginary() {
        return this -> imaginary;
    }
};

int main () {
    ios_base::sync_with_stdio(0);
    float realPart, imaginaryPart;
    cout << "Enter real part of A : ";
    cin >> realPart;
    cout << "Enter imaginary part of A : ";
    cin >> imaginaryPart;
    complexNumber A(realPart, imaginaryPart);
```

```
    cout << "Enter real part of B : ";
    cin >> realPart;
    cout << "Enter imaginary part of B : ";
    cin >> imaginaryPart;
    complexNumber B(realPart, imaginaryPart);
    complexNumber C = A + B;
    cout << "Real : " << C.getReal() << "\n" ;
    cout << "Imaginary : " << C.getImaginary() << "\n" ;
}
```

 C:\WINDOWS\system32\cmd.exe

```
C:\Users\Arpit\Documents\OOPS Lab>g++ experiment2.cpp -o experiment2 && experiment2
Enter real part of A : 10
Enter imaginary part of A : 20
Enter real part of B : 20
Enter imaginary part of B : 30
Real : 30
Imaginary : 50
C:\Users\Arpit\Documents\OOPS Lab>
```

Experiment 3

Create a class to represent a vector (of float values) by including the member functions:

- a) To create a vector**
- b) To modify the vector**
- c) To display the vector**
- d) To add to two vectors and display the result**

```
#include <iostream>
#include <vector>
using namespace std;
class Vector {
    int components;
public:
    vector<float> _vector;
    Vector(int components) {
        this->components = components;
        this->_vector.resize(components);
    }
    void Display() {
        cout << "(";
        for (int i = 0; i < components; i++) {
            cout << _vector[i];
            if (i != components - 1)
                cout << ",";
        }
        cout << ")\n";
    }
    void Modify() {
        cout << "Enter new vector\n";
        for (int i = 0; i < components; i++)
            cin >> this -> _vector[i];
    }
}
```

```

Vector operator + (const Vector &v) {
    if (this->components != v.components) {
        cout << "Cannot add due to difference in dimension\n";
        return *new Vector(0);
    }
    Vector* added = new Vector(this->components);
    for (int i = 0; i < this->components; i++)
        added->_vector[i] = (this->_vector[i] + v._vector[i]);
    return *added;
}

};

int main() {
    ios_base::sync_with_stdio(0);
    int components;
    cout << "Enter number of components in vector A : ";
    cin >> components;
    Vector* a = new Vector(components);
    cout << "Enter the components\n";
    for (int i = 0; i < components; i++)
        cin >> a->_vector[i];
    cout << "Enter number of components in vector B : ";
    cin >> components;
    Vector* b = new Vector(components);
    cout << "Enter the components\n";
    for (int i = 0; i < components; i++)
        cin >> b->_vector[i];
    char ch = 'y';
    while(ch == 'y') {
        cout << "1. Modify a vector\n" << "2. Display a vector\n" << "3.
Add the vectors\n";
        cin >> ch;
        switch(ch) {
            case '1':
                cout << "Which Vector A/B ?";

```

```

        char c1;
        cin >> c1;
        if(c1 == 'A' || c1 == 'a')
            a -> Modify();
        else
            b -> Modify();
        break;
    case '2':
        cout << "Which Vector A/B ?";
        cin >> c1;
        if(c1 == 'A' || c1 == 'a')
            a -> Display();
        else
            b -> Display();
        break;
    case '3':
        Vector c = *a + *b;
        if(c._vector.size() != 0)
            cout << "Addition Vector : ", c.Display();
        break;
    }
    cout << "Continue (y/n)?\n";
    cin >> ch;
}
return 0;
}

```


C:\WINDOWS\system32\cmd.exe

C:\Users\Arpit\Documents\OOPS Lab>g++ experiment3.cpp -o experiment3 && experiment3

Enter number of components in vector A : 3

Enter the components

10

20

30

Enter number of components in vector B : 3

Enter the components

20

30

40

1. Modify a vector.

2. Display a vector

3. Add the vectors

2

Which Vector A/B ?A

(10,20,30)

Continue (y/n)?

y

1. Modify a vector.

2. Display a vector

3. Add the vectors

2

Which Vector A/B ?B

(20,30,40)

Continue (y/n)?

y

1. Modify a vector.

2. Display a vector

3. Add the vectors

3

Addition Vector : (30,50,70)

Continue (y/n)?

n

C:\Users\Arpit\Documents\OOPS Lab>

Experiment 4

What are dynamic constructors? Explain with an example and write a C++ program to illustrate.

Dynamic constructor is used to allocate the memory to the objects at the run time. Memory is allocated at run time with the help of 'new' operator.

```
#include <iostream>
```

```
using namespace std;
```

```
class Coordinate {
```

```
    int x, y;
```

```
public:
```

```
    Coordinate() {
```

```
        cout << "Default Constructor\n";
```

```
        x = 0;
```

```
        y = 0;
```

```
    }
```

```
    Coordinate(int x, int y) {
```

```
        this -> x = x;
```

```
        this -> y = y;
```

```
        cout << "Initialised Coordinate object with (" << x << ", " << y  
<< ")\n";
```

```
    }
```

```
};
```

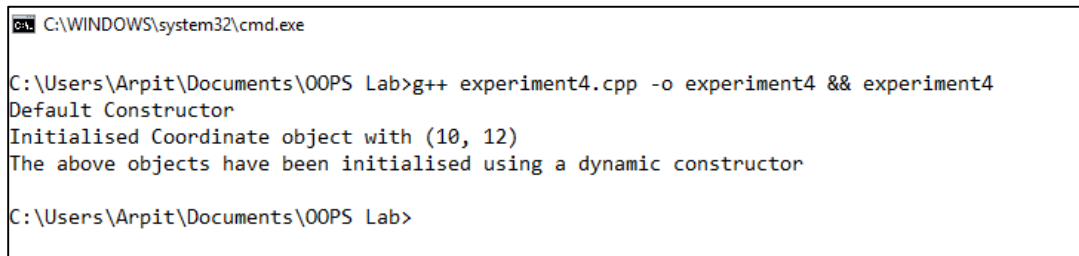
```
int main() {
```

```
    Coordinate c1, *c2 = new Coordinate(10, 12);
```

```
    cout << "The above objects have been initialised using a dynamic  
constructor\n";
```

```
    return 0;
```

```
}
```



```
C:\WINDOWS\system32\cmd.exe  
  
C:\Users\Arpit\Documents\OOPS Lab>g++ experiment4.cpp -o experiment4 && experiment4  
Default Constructor  
Initialised Coordinate object with (10, 12)  
The above objects have been initialised using a dynamic constructor  
  
C:\Users\Arpit\Documents\OOPS Lab>
```

Experiment 5

A Bank maintains two types of accounts namely Savings and Current account. The savings account provides compound interest and withdrawal facilities but no cheque book. The current account provides cheque book facility without interest. All the account holders should maintain a minimum balance RS. 500. The current account holder balance falls below the minimum balance a service charge is imposed and overdraft limit is RS. 1000. The Account has members (AC No, name, type: S for savings, C for current). Two classes derived from base class Account such as curr_account has member functions (cheque, overdraft limit) and sav_account (compound_interest, withdrawal). Include necessary member functions in order to achieve

- 1) Accept deposit and update balance
- 2) Compute and deposit interest
- 3) Permit Withdrawal
- 4) Check for minimum balance, impose penalty if necessary and update balance.

```
#include <iostream>
#include <string>
#include <cstdio>
#include <cmath>
using namespace std;
class Account {
protected:
    string account_no, name;
    char type;
    float balance;
public:
    void updateBalance(float amount) {
        this -> balance += amount;
        printBalance();
    }
    void printBalance() {
        cout << "Transaction Complete ! Your account balance is : Rs. " << this
        -> balance << "\n";
        checkMinimum();
    }
}
```

```

void computeInterest() {
    if(this -> type == 'C') {
        cout << "Current Account does not have Interest facility !\n";
        return;
    }
    int years;
    cout << "Enter time in years for interest compounded annually at 10%:
";
    cin >> years;
    float p = this -> balance;
    float temp = (1 + 10.0 / 100.0);
    float interest = pow(temp, years);
    interest *= p;
    cout << "Interest : Rs. " << interest << "\n";
    updateBalance(interest);
}

void checkMinimum() {
    if(this -> balance < 500.0) {
        cout << "Your account balance is below the minimum balance, a
penalty of Rs. 100 will be imposed\n";
        updateBalance(-100.0);
    } else
        cout << "Your account balance is above the minimum balance\n";
}

void details() {
    cout << "Name : " << this -> name << "\n";
    cout << "Type : " << this -> type << "\n";
    cout << "Balance : " << this -> balance << "\n";
}
};

```

```

class SavingsAccount : public Account {
public:
    SavingsAccount(string account_no, string name, float balance) {

```

```

        this -> account_no = account_no;
        this -> name = name;
        this -> type = 'S';
        this -> balance = balance;
    }
};

class CurrentAccount : public Account {
public:
    CurrentAccount(string account_no, string name, float balance) {
        this -> account_no = account_no;
        this -> name = name;
        this -> type = 'C';
        this -> balance = balance;
    }
};

int main() {
    ios_base::sync_with_stdio(0);
    char type;
    string name, account_no;
    float amount, balance;
    cout << "Enter type of account Savings / Current (S / C) : ";
    cin >> type;
    cout << "Enter your name : ";
    cin >> name;
    cout << "Enter account no. : ";
    cin >> account_no;
    cout << "Enter Initial Balance : ";
    cin >> balance;
    while (balance < 500.0) {
        cout << "Minimum Balance is Rs. 500.0" << "\n";
        cout << "Please enter Initial balance again : ";
        cin >> balance;
    }
    if(type == 'S') {

```

```

SavingsAccount sAccount(account_no, name, balance);
char option = 'c';
while (option != '0') {
    cout << "1. Deposit\n";
    cout << "2. Withdraw\n";
    cout << "3. Compute Interest\n";
    cout << "4. Check for Minimum Balance\n";
    cout << "5. Details\n";
    cout << "0. Exit\n";
    cin >> option;
    switch(option) {
        case '1' :
            cout << "Enter amount to deposit : ";
            cin >> amount;
            sAccount.updateBalance(amount);
            break;
        case '2' :
            cout << "Enter amount to withdraw : ";
            cin >> amount;
            sAccount.updateBalance(-amount);
            break;
        case '3' :
            cout << "Computing Interest...\n";
            sAccount.computeInterest();
            break;
        case '4' :
            sAccount.checkMinimum();
            break;
        case '5' :
            sAccount.details();
            break;
        case '0' :
            break;
    }
}

```

```

    }
}
else {
    CurrentAccount cAccount(account_no, name, balance);
    char option = 'c';
    while (option != '0') {
        cout << "1. Deposit\n";
        cout << "2. Withdraw\n";
        cout << "3. Compute Interest\n";
        cout << "4. Check for Minimum Balance\n";
        cout << "5. Details\n";
        cout << "0. Exit\n";
        cin >> option;
        switch(option) {
            case '1' :
                cout << "Enter amount to deposit : ";
                cin >> amount;
                cAccount.updateBalance(amount);
                break;
            case '2' :
                cout << "Enter amount to withdraw : ";
                cin >> amount;
                cAccount.updateBalance(-amount);
                break;
            case '3' :
                cout << "Computing Interest...\n";
                cAccount.computeInterest();
                break;
            case '4' :
                cAccount.checkMinimum();
                break;
            case '5' :
                cAccount.details();
                break;

```

```

        case '0' :
            break;

    }

}

}

}

```

```

CA% experiment5

C:\Users\Arpit\Documents\OOPS Lab>g++ experiment5.cpp -o experiment5 && experiment5
Enter type of account Savings / Current (S / C) : S
Enter your name : ASD
Enter account no. : 1242314
Enter Initial Balance : 10000
1. Deposit
2. Withdraw
3. Compute Interest
4. Check for Minimum Balance
5. Details
0. Exit
1
Enter amount to deposit : 1200
Transaction Complete ! Your account balance is : Rs. 11200
Your account balance is above the minimum balance
1. Deposit
2. Withdraw
3. Compute Interest
4. Check for Minimum Balance
5. Details
0. Exit
3
Computing Interest...
Enter time in years for interest compounded annually at 10%: 2
Interest : Rs. 13552
Transaction Complete ! Your account balance is : Rs. 24752
Your account balance is above the minimum balance
1. Deposit
2. Withdraw
3. Compute Interest
4. Check for Minimum Balance
5. Details
0. Exit
2
Enter amount to withdraw : 10000
Transaction Complete ! Your account balance is : Rs. 14752
Your account balance is above the minimum balance
1. Deposit
2. Withdraw
3. Compute Interest
4. Check for Minimum Balance
5. Details
0. Exit
5
Name : ASD
Type : S
Balance : 14752

```


Experiment 6

What are virtual functions? Write a C++ program to illustrate virtual functions to find the product of any two given matrices.

Virtual Function is a function in base class, which is overridden in the derived class, and which tells the compiler to perform Late Binding on this function.

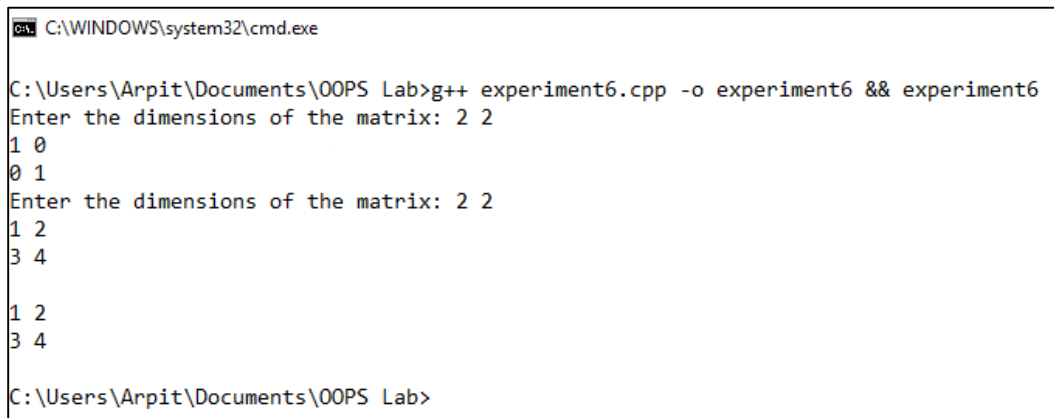
```
#include <iostream>
#include <vector>
using namespace std;
class mat{
    public: virtual void product(){}
};
class pMat : public mat {
    public:
        vector<vector<int> > arr;
        int m, n;
        void read(){
            cout << "Enter the dimensions of the matrix: ";
            cin >> m >> n;
            arr.resize(m, vector<int>(n));
            for(int i = 0; i < m; i++)
                for(int j = 0; j < n; j++)
                    cin >> arr[i][j];
        }
        void display(){
            for(int i = 0; i < m; i++){
                for(int j = 0; j < n; j++)
                    cout << arr[i][j] << " ";
                cout << endl;
            }
        }
        void product(){
            pMat mat1, mat2;
            mat1.read();
```

```

        mat2.read();
        int m1, n1, m2, n2;
        m1 = mat1.m;
        n1 = mat1.n;
        m2 = mat2.m;
        n2 = mat2.n;
        vector<vector<int> > finMat(m1, vector<int>(n2));
        for(int i = 0; i < m1; i++)
            for(int j = 0; j < n2; j++)
                for(int i1 = 0; i1 < n1; i1++)
                    finMat[i][j] += mat1.arr[i][i1] * mat2.arr[i1][j];
        cout << endl;
        for(int i = 0; i < m1; i++){
            for(int j = 0; j < n2; j++)
                cout << finMat[i][j] << " ";
            cout << endl;
        }
    }
};

int main(){
    mat *ptr;
    pMat obj;
    ptr = &obj;
    ptr->product();
    return 0;
}

```



```

C:\WINDOWS\system32\cmd.exe

C:\Users\Arpit\Documents\OOPS Lab>g++ experiment6.cpp -o experiment6 && experiment6
Enter the dimensions of the matrix: 2 2
1 0
0 1
Enter the dimensions of the matrix: 2 2
1 2
3 4

1 2
3 4

C:\Users\Arpit\Documents\OOPS Lab>

```

Experiment 7

Explain the difference between static and dynamic polymorphism. Write a C++ program to demonstrate both these polymorphisms.

Polymorphism is of two types, static or dynamic. In dynamic polymorphism the response to message is decided on run-time while in static polymorphism it is decided on compile-time.

The assignment of data types in dynamic polymorphism is known as late or dynamic binding. In dynamic binding method call occur based on the object (instance) type at Run time. Eg: method overriding

If the assignment of data types is in compile time it is known as early or static binding. In static binding method call occur based on the reference type at compile time. Eg: method overloading

Method Overloading - This means creating a new method with the same name and different signature. It uses early binding.

Method Overriding - This is the process of giving a new definition for an existing method in its child class. All object created at run time on the heap therefore actual binding is done at the runtime only.

Method overloading would be an example of static polymorphism, whereas overriding would be an example of dynamic polymorphism.

```
/*
```

```
    In this program, static and dynamic polymorphism have been demonstrated.
```

```
    The area() function is inherited in both Rectangle and Triangle classes however, the compiler decides beforehand that the
```

```
    area() function of the base class will be called. This is not the case with the perimeter() function which uses the virtual
```

```
    keyword that tells the compiler to decide which function is called at runtime. This is an example of dynamic polymorphism.
```

```
*/
```

```
#include <iostream>
```

```
#include <cmath>
```

```
using namespace std;
```

```
class Shape {
```

```
protected:
```

```
    int width, height;
```

```
public:
```

```
    Shape(int a = 0 , int b = 0) {
```

```

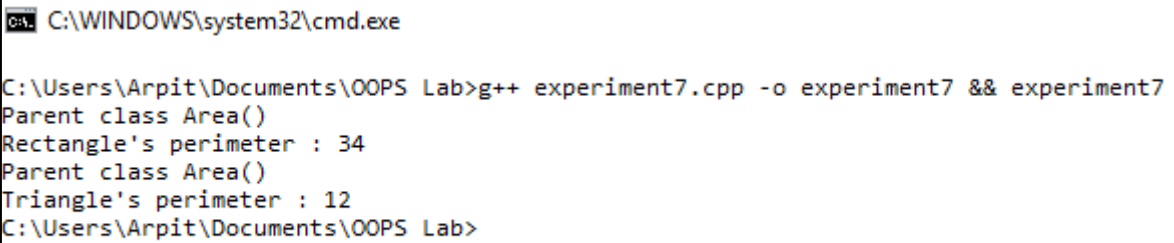
        width = a;
        height = b;
    }
    void area() {
        cout << "Parent class Area()\n";
    }
    virtual void perimeter() {
        cout << "Parent class Perimeter()\n";
    }
};

class Rectangle : public Shape {
public:
    Rectangle(int a = 0, int b = 0) : Shape(a, b) {}
    void area() {
        cout << "Rectangle's area : " << (width * height) << "\n";
    }
    void perimeter() {
        cout << "Rectangle's perimeter : " << (2 * (width + height))
<< "\n";
    }
};

class Triangle : public Shape {
public:
    Triangle(int a = 0, int b = 0) : Shape(a, b) {}
    void area() {
        cout << "Triangle's area : " << (width * height / 2) <<
"\n";
    }
    void perimeter() {
        cout << "Triangle's perimeter : " << (width + height +
(int)sqrt(pow(width, 2) + pow(height, 2)))
    }
};

```

```
int main() {  
    Shape *shape;  
    Rectangle r(10, 7);  
    Triangle t(3, 4);  
    shape = &r;  
    shape -> area();  
    shape -> perimeter();  
    shape = &t;  
    shape -> area();  
    shape -> perimeter();  
    return 0;  
}
```



C:\WINDOWS\system32\cmd.exe

```
C:\Users\Arpit\Documents\OOPS Lab>g++ experiment7.cpp -o experiment7 && experiment7  
Parent class Area()  
Rectangle's perimeter : 34  
Parent class Area()  
Triangle's perimeter : 12  
C:\Users\Arpit\Documents\OOPS Lab>
```

Experiment 8

Write a C++ exception handler if the user tries to refer the index more than the specific size of the array.

```
#include <iostream>
#include <exception>
using namespace std;
struct ArrayIndexOutOfBoundsException : public exception {
    const char* what () const throw () {
        return "Invalid Array Index";
    }
};
int main() {
    int n;
    cout << "Enter the size of the array to create : \n";
    cin >> n;
    int array[n];
    for(int i = 0 ; i < n ; i++)
        array[i] = (i + 1);
    cout << "Array initialised with 1 - " << n << "\n";
    try {
        cout << "Enter an index : ";
        int index;
        cin >> index;
        if(index < 0 || index >= n)
            throw ArrayIndexOutOfBoundsException();
        cout << "Element at " << index << " : " << array[index] << "\n";
    } catch (ArrayIndexOutOfBoundsException& e) {
        cout << e.what() << "\n";
    }
}
```

C:\WINDOWS\system32\cmd.exe

C:\Users\Arpit\Documents\OOPS Lab>g++ experiment8.cpp -o experiment8 && experiment8

Enter the size of the array to create :

4

Array initialised with 1 - 4

Enter an index : 10

Invalid Array Index

C:\Users\Arpit\Documents\OOPS Lab>

Experiment 9

Create a file called "Student.dat" which contains the details of the students such as Name, Semester, and Marks in six subjects, total marks. If the candidate got more than 40 marks in each subject then the result is pass else the result is fail. The passed student's details should be stored in a file "Passed.dat" and failed students in "Failed.dat".

```
#include <iostream>
#include <algorithm>
#include <fstream>
#include <string>
using namespace std;
class Student {
    string name;
    int semester;
    int marks[6], total;
    void write_student(ofstream &fs) {
        fs << "Name : " << name << "\n";
        fs << "Semester : " << semester << "\n";
        fs << "Marks : (";
        for (int i = 0; i < 6; i++) {
            fs << marks[i];
            if (i == 5)
                fs << ")";
            else
                fs << ",";
        }
        fs << "\n";
        fs << "Total : " << total << "\n";
    }
public:
    void read() {
        cout << "Enter Name, semester and Marks in 6 subjects\n";
        cin >> name;
        cin >> semester;
```



```

        total = 0;
        for(int i = 0 ; i < 6 ; i++)
            cin >> marks[i], total += marks[i];
    }
    void write() {
        ofstream fs;
        fs.open("Student.dat", ios::binary | ofstream::app);
        write_student(fs);
        fs.close();
        int min_marks = *min_element(marks, marks + 6);
        if (min_marks >= 40) {
            ofstream pass_fs;
            pass_fs.open("Passed.dat", ios::binary | ofstream::app);
            write_student(pass_fs);
            pass_fs.close();
        }
        else {
            ofstream fail_fs;
            fail_fs.open("Failed.dat", ios::binary | ofstream::app);
            write_student(fail_fs);
            fail_fs.close();
        }
    }
};

int main() {
    int number;
    cout << "Enter the number of students to enter : ";
    cin >> number;
    while(number-->0) {
        Student student;
        student.read();
        student.write();
    }
    return 0;
}

```

}

```
C:\WINDOWS\system32\cmd.exe

C:\Users\Arpit\Documents\OOPS Lab>g++ experiment9.cpp -o experiment9 && experiment9
Enter the number of students to enter : 3
Enter Name, semester and Marks in 6 subjects
ASD
4
39 41 41 41 41 41
Enter Name, semester and Marks in 6 subjects
BEF
3
77 66 88 99 100 55
Enter Name, semester and Marks in 6 subjects
CGH
2
10 23 45 65 76 88

C:\Users\Arpit\Documents\OOPS Lab>
```

Student.dat	Failed.dat	Passed.dat
1 Name : ASD	1 Name : ASD	1 Name : BEF
2 Semester : 4	2 Semester : 4	2 Semester : 3
3 Marks : (39,41,41,41,41,41)	3 Marks : (39,41,41,41,41,41)	3 Marks : (77,66,88,99,100,55)
4 Total : 244	4 Total : 244	4 Total : 485
5 Name : BEF	5 Name : CGH	
6 Semester : 3	6 Semester : 2	
7 Marks : (77,66,88,99,100,55)	7 Marks : (10,23,45,65,76,88)	
8 Total : 485	8 Total : 307	
9 Name : CGH		
10 Semester : 2		
11 Marks : (10,23,45,65,76,88)		
12 Total : 307		

Experiment 10

What is virtual base class? Explain. Illustrate it with a C++ program.

When two or more objects are derived from a common base class, we can prevent multiple copies of the base class being present in an object derived from those objects by declaring the base class as virtual when it is being inherited. Such a base class is known as virtual base class.

```
#include <iostream>

using namespace std;

class A {
public:
    int i;
};

class B : virtual public A {
public:
    int j;
};

class C: virtual public A {
public:
    int k;
};

class D: public B, public C {
public:
    int sum;
};

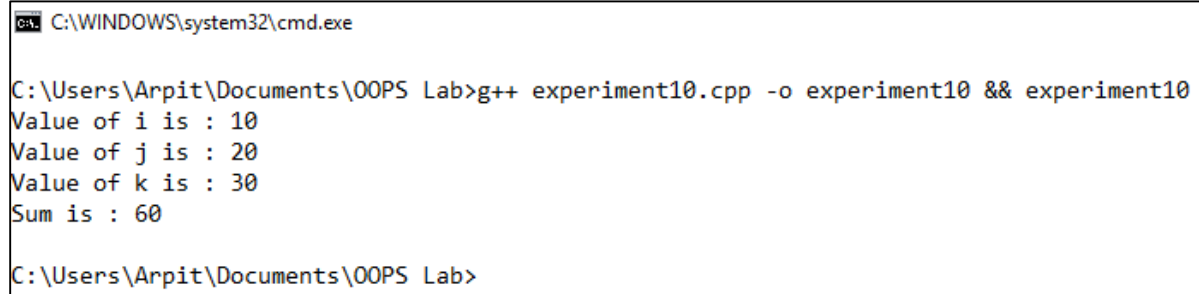
int main() {
    D ob;

    ob.i = 10; //unambiguous since only one copy of i is inherited.
    ob.j = 20;
    ob.k = 30;

    ob.sum = ob.i + ob.j + ob.k;

    cout << "Value of i is : " << ob.i << "\n";
    cout << "Value of j is : " << ob.j << "\n";
    cout << "Value of k is : " << ob.k << "\n";
    cout << "Sum is : " << ob.sum << "\n";
}
```

```
    return 0;  
}
```



A screenshot of a Windows command prompt window. The title bar reads "C:\WINDOWS\system32\cmd.exe". The command prompt shows the following sequence of text: a prompt "C:\Users\Arpit\Documents\OOPS Lab>" followed by the command "g++ experiment10.cpp -o experiment10 && experiment10", and then the output of the program: "Value of i is : 10", "Value of j is : 20", "Value of k is : 30", and "Sum is : 60". Finally, the prompt "C:\Users\Arpit\Documents\OOPS Lab>" is shown again.

```
C:\WINDOWS\system32\cmd.exe  
  
C:\Users\Arpit\Documents\OOPS Lab>g++ experiment10.cpp -o experiment10 && experiment10  
Value of i is : 10  
Value of j is : 20  
Value of k is : 30  
Sum is : 60  
  
C:\Users\Arpit\Documents\OOPS Lab>
```