**Question Number 1:**

**PROGRAM**

#include <iostream>

using namespace std;

class Arthematic

{

public:

Arthematic(int a=0,int b=0)

{

num1 = a, num2 = b;

}

virtual void setdata()

{

cout << "(From Base) Enter the Value of 1st Number : ";

cin >> num1;

cout << "(From Base) Enter the value of 2nd Number : ";

cin >> num2;

}

float num1, num2;

};

class Add :public Arthematic

{

public:

Add() :Arthematic(num1, num2){}

void op()

{

cout << "Addition of both Number is : " << num1 + num2 << endl;

}

void setdata()

{

cout << "(From Derived) Enter the Value of 1st Number : ";

cin >> num1;

cout << "(From Derived) Enter the value of 2nd Number : ";

cin >> num2;

}

};

class Subtract :public Arthematic

{

public:

Subtract() :Arthematic(num1, num2) {}

void op()

{

cout << "Subtraction of both Number is : " << num1 - num2 << endl;

}

void setdata()

{

cout << "(From Derived) Enter the Value of 1st Number : ";

cin >> num1;

cout << "(From Derived) Enter the value of 2nd Number : ";

cin >> num2;

}

};

int main()

{

Arthematic \*A, \*S;

Add add;

Subtract subtract;

A = &add;

S = &subtract;

cout << "Enter Numbers for Addition !!" << endl;

A->setdata();

cout << endl;

cout << "Enter Numbers for Subtraction !!" << endl;

S->setdata();

cout << endl;

add.op();

subtract.op();

cout << endl << endl;

system("pause");

}

**A screenshot of a computer screen

Description automatically generated**

**Question Number 2:**

**PROGRAM**

#include <iostream>

using namespace std;

class Shape

{

public:

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

{

length = a, width = b;

}

virtual void getdata()

{

cout << "(BASE) Enter the Value of Length : ";

cin >> length;

cout << "(BASE) Enter the Value of Width : ";

cin >> width;

}

virtual void area() = 0;

float length, width;

};

class Square :public Shape

{

public:

Square():Shape(length,width){}

void getdata()

{

cout << "Enter the Value of Length : ";

cin >> length;

}

void area()

{

cout << "Area of Square is : " << length\*length << endl;

}

};

class Rectangle :public Shape

{

public:

Rectangle() :Shape(length, width) {}

void getdata()

{

cout << "Enter the Value of Length : ";

cin >> length;

cout << "Enter the Value of Width : ";

cin >> width;

}

void area()

{

cout << "Area of Rectangle is : " << length \* width << endl;

}

};

class Circle :public Shape

{

public:

Circle() :Shape(length, width) {}

void getdata()

{

cout << "Enter the Value of Radius : ";

cin >> width;

}

void area()

{

cout << "Area of Circle is : " << 3.14 \* (width \* width) << endl;

}

};

int main()

{

Shape \*S, \*R, \*C;

Square square;

Rectangle rectangle;

Circle circle;

S = &square;

R = &rectangle;

C = &circle;

cout << "Enter Values For Square !!" << endl;

S->getdata();

cout << endl << "Enter Values For Rectangle !!" << endl;

R->getdata();

cout << endl << "Enter Values For Circle !!" << endl;

C->getdata();

cout << endl;

S->area();

R->area();

C->area();

cout << endl << endl;

system("pause");

}

**A screenshot of a computer screen

Description automatically generated**

**Question Number 3:**

**PROGRAM**

#include <iostream>

using namespace std;

class Employee

{

public:

virtual void Calculate\_Pay()

{

cout << "Employee Class Dosen't have Any Method To Calculate Pay " << endl;

}

};

class Manager :public Employee

{

public:

void Calculate\_Pay()

{

cout << "The Calculated Pay of The Manager is : $$$$" << endl;

}

};

class Hourly :public Employee

{

public:

void Calculate\_Pay()

{

cout << "Hourly Pay is : $50 " << endl;

}

};

int main()

{

Employee \*PTR,E;

Manager M;

Hourly H;

PTR = &E;

PTR->Calculate\_Pay(); // For Employee

PTR = &M;

PTR->Calculate\_Pay(); // For Manager

PTR = &H;

PTR->Calculate\_Pay(); // For Hourly

cout << endl << endl;

system("pause");

}

**A screenshot of a computer screen

Description automatically generated**

**Question Number 4:**

**PROGRAM (By Making Object of Derived Class)**

**(IT IS NOT POSSIBLE)**

#include <iostream>

using namespace std;

class B\_Over\_Riding

{

public:

virtual void M\_A(int a=0)

{

cout << "BASE: " << a << endl;

}

virtual void M\_A(int a = 0, int b = 0)

{

cout << "BASE: " << a <<" " << b <<endl;

}

virtual void M\_A(int a = 0, int b = 0, int c = 0)

{

cout << "BASE: " << a <<" "<< b <<" " << c << endl;

}

};

class D\_Over\_Riding :public B\_Over\_Riding

{

public:

void M\_A(int a = 0, int b = 0, int c = 0)

{

cout << "Derived: " << a << " " << b << " " << c << endl;

}

};

int main()

{

D\_Over\_Riding Der;

Der.M\_A(1);

Der.M\_A(1, 2); // Calling Base Function From Derived

Der.M\_A(1,2,3); // It is not possible as shown in output

cout << endl << endl;

system("pause");

}

**A screenshot of a computer screen

Description automatically generated**

**PROGRAM (By Giving Address to Base Function)**

**(IT IS POSSIBLE)**

#include <iostream>

using namespace std;

class B\_Over\_Riding

{

public:

virtual void M\_A(int a = 0)

{

cout << "BASE: " << a << endl;

}

virtual void M\_A(float a = 0, int b = 0)

{

cout << "BASE: " << a <<" , " << b <<endl;

}

virtual void M\_A(char a = 'a', float b = 0, int c = 0)

{

cout << "BASE: " << a << " , " << b <<" , " << c << endl;

}

};

class D\_Over\_Riding :public B\_Over\_Riding

{

public:

void M\_A(char a = 'a', float b = 0, int c = 0)

{

cout << "Derived: " << a << " , " << b << " , " << c << endl;

}

};

int main()

{

B\_Over\_Riding \*Base;

D\_Over\_Riding Der;

Base = &Der;

Base->M\_A(1);

Base->M\_A(1.1, 2); // Calling Function of Base Class by Derived Class Address

Base->M\_A('A', 2.2, 1); // It is possible as shown in output

cout << endl << endl;

system("pause");

}

**A screenshot of a computer screen

Description automatically generated**

**PROGRAM (Without Over-Riding Functions)**

**(IT IS POSSIBLE)**

#include <iostream>

using namespace std;

class B\_Over\_Riding

{

public:

virtual void M\_A(int a = 0)

{

cout << "BASE: " << a << endl;

}

virtual void M\_A(float a = 0, int b = 0)

{

cout << "BASE: " << a <<" , " << b <<endl;

}

virtual void M\_A(char a = 'a', float b = 0, int c = 0)

{

cout << "BASE: " << a << " , " << b <<" , " << c << endl;

}

};

class D\_Over\_Riding :public B\_Over\_Riding

{

public:

};

int main()

{

D\_Over\_Riding Der;

Der.M\_A(1);

Der.M\_A(1.1, 2); // Calling Function of Base Class Without over-riding in derived class

Der.M\_A('A', 2.2, 1); // It is possible as shown in output

cout << endl << endl;

system("pause");

}

**A screenshot of a computer screen

Description automatically generated**

**Question Number 5:**

**PROGRAM**

#include <iostream>

using namespace std;

class Account

{

public:

virtual void Calculate() = 0;

virtual void Show() = 0;

};

class Current :public Account

{

public:

void Calculate()

{

cout << "Total Salary is : 10,000 " << endl;

cout << "Zakaat Deduction of 2.5% on Salary is = " << ((2.5 \* 10000) / 100) << endl;

}

void Show()

{

cout << "Salary After Zakaat Deduction is : " << 10000 - ((2.5 \* 10000) / 100) << endl;

}

};

class Saving :public Account

{

public:

void Calculate()

{

cout << "Total Salary is : 10,000 " << endl;

cout << "Annual Increment of 5% on Salary is = " << (5 \* 10000) / 100 << endl;

cout << "Zakaat Deduction of 2.5% on Salary is = " << ((2.5 \* 10000) / 100) << endl;

}

void Show()

{

cout << endl << "Salary After Annual Increment is : " << ((5 \* 10000) / 100) + 10000 << endl;

cout << "Salary After Zakaat Deduction is : " << 10000 - ((2.5 \* 10000) / 100) << endl;

cout << "Over-All Salary Becomes : " << (((5 \* 10000) / 100) + 10000) - ((2.5 \* 10000) / 100);

}

};

int main()

{

Account \*A;

Saving S;

Current C;

cout << "Data For Saving Account " << endl << endl;

A = &S;

A->Calculate();

A->Show();

cout << endl << endl << "Data For Current Account " << endl << endl;

A = &C;

A->Calculate();

A->Show();

cout << endl << endl;

system("pause");

}

**A screenshot of a computer screen

Description automatically generated**