CHITKARA INHERITANCE (10 Marks)

Q1. The examination department prints the result of N students by implementing three classes:

1. **Student,**
2. **Marks**
3. **Result**.

Class **‘Student’** contains the **roll\_number**  and two member functions **set\_roll\_number()** and **get\_roll\_number()** to read and display the **Roll number**.

Class **‘Marks’** inherits class **Student** and contains the marks of two subjects as **maths, physics** as its data members and two member functions **set\_marks() and get\_marks()** to get and display the marks for two subjects.

Class **‘Result’** further inherits from **class Marks** and display Roll number, Marks of two subjects and Percentage of the student in the following form:

##### **Output Format**

#### Roll number = 678

#### Marks of Math's = 92

#### Marks of Physics = 98

#### Percentage = 95

**Test case 1**

**Output**

#### Roll number = 678

#### Marks of Math's = 92

#### Marks of Physics = 98

#### Percentage = 95

#### DRIVER CODE:

int main()

{

    Result R;

    R.set\_roll\_number(678);

    R.set\_marks(92,98);

    R.display\_result();

    return 0;

}

**Test case 2**

**Output**

#### Roll number = 987

#### Marks of Math's = 95

#### Marks of Physics = 98

#### Percentage = 96.5

#### DRIVER CODE:

int main()

{

    Result R;

    R.set\_roll\_number(987);

    R.set\_marks(95,98);

    R.display\_result();

    return 0;

}

**Test case 3**

**Output**

#### Roll number = 876

#### Marks of Math's = 92

#### Marks of Physics = 95

#### Percentage = 93.5

#### DRIVER CODE:

int main()

{

    Result R;

    R.set\_roll\_number(876);

    R.set\_marks(92,95);

    R.display\_result();

    return 0;

}

**Complete Code**

#include<iostream>

using namespace std;

class Student

{

    protected:

        int roll\_number;

    public:

        void set\_roll\_number(int);

        void get\_roll\_number(void);

};

void Student :: set\_roll\_number(int r)

{

    roll\_number = r;

}

void Student :: get\_roll\_number()

{

    cout<<"Roll number = "<< roll\_number <<endl;

}

class Marks : public Student

{

  protected:

      float maths;

      float physics;

   public:

      void set\_marks(float,float);

      void get\_marks(void);

};

void Marks :: set\_marks(float m, float p)

{

    maths = m;

    physics = p;

}

void Marks :: get\_marks()

{

     cout<< "Marks of Math's = "<< maths<<endl;

     cout<< "Marks of Physics = "<< physics<<endl;

}

class Result : public Marks

{

     float percentage;

     public:

void display\_result()

{

    get\_roll\_number();

    get\_marks();

    cout<< "Percentage = "<<(maths+physics)/2<<endl;

}

};

int main()

{

    Result R;

    R.set\_roll\_number(876);

    R.set\_marks(92,95);

    R.display\_result();

    return 0;

}

CHITKARA WALL (5 Marks)

2. Write a C++ program to calculate the area of a wall using class object and Parameterized constructor

**Test case 1**

**Output**

Area of Wall 1: 90.3

Area of Wall 2: 53.55

#### DRIVER CODE:

int main()

{

  Wall wall1(10.5, 8.6);

  Wall wall2(8.5, 6.3);

  cout << "Area of Wall 1: " << wall1.calculateArea() << endl;

  cout << "Area of Wall 2: " << wall2.calculateArea();

  return 0;

}

**Test case 2**

**Output**

Area of Wall 1: 143.64

Area of Wall 2: 142.35

#### DRIVER CODE:

int main()

{

  Wall wall1(18.9, 7.6);

  Wall wall2(19.5, 7.3);

  cout << "Area of Wall 1: " << wall1.calculateArea() << endl;

  cout << "Area of Wall 2: " << wall2.calculateArea();

  return 0;

}

**Test case 3**

**Output**

#### Area of Wall 1: 349.6

#### Area of Wall 2: 700.8

#### DRIVER CODE:

int main()

{

  Wall wall1(46, 7.6);

  Wall wall2(96, 7.3);

  cout << "Area of Wall 1: " << wall1.calculateArea() << endl;

  cout << "Area of Wall 2: " << wall2.calculateArea();

  return 0;

}

Complete Code

#include <iostream>

using namespace std;

class Wall

{

  private:

    double length;

    double height;

  public:

    // parameterized constructor to initialize variables

    Wall(double len, double hgt)

    {

      length = len;

      height = hgt;

    }

    double calculateArea()

    {

      return length \* height;

    }

};

int main()

{

  // create object and initialize data members

  Wall wall1(46, 7.6);

  Wall wall2(96, 7.3);

  cout << "Area of Wall 1: " << wall1.calculateArea() << endl;

  cout << "Area of Wall 2: " << wall2.calculateArea();

  return 0;

}

Addition and Subtraction using Pointers (5 Marks)

#### Q3.You have given two integer variables, say a and b and two pointer type variables pa and pb. pa and pb contains the memory address of a and b respectively. Call the function update() which modify the value of a and b in memory. The updated values of a and b are the sum (a+b) and absolute difference |(a-b) | respectively. Complete the update () function in the drive source code.

Function Description :

The function update() accepts the following parameters:

1 int \*a: an integer

2 int \*b: an integer

and is declared with a void return type, so there is no value to return. Modify the values in memory only so that these integer variables ‘a’ and ‘b’ must be updated with sum and absolute difference respectively.

a=a+b

b=|a-b|

#### **Input Format :**

#### Input will contain two integers, and , separated by a newline.

#### **Output Format :**

#### The output will appear in two lines. 1st line contains their sum 2nd line contains the difference

#### **Sample Input**

40 // first integer value

50 // second integer value

#### 

#### **Sample Output :**

90 // Their sum i.e., 40+50=90

10 // Their absolute difference i.e.,|40-50| = 10

**Test case 1**

**Output**

#### 99

#### 31

#### DRIVER CODE:

int main()

{

int a = 34;

int b = 65;

int \*pa = &a, \*pb = &b;

update(pa, pb);

cout<<a<<endl;

cout<<b<<endl;

return 0;

}

**Test case 2**

**Output**

#### 108

#### 22

#### DRIVER CODE:

int main()

{

int a = 43;

int b = 65;

int \*pa = &a, \*pb = &b;

update(pa, pb);

cout<<a<<endl;

cout<<b<<endl;

return 0;

}

**Test case 3**

**Output**

#### 97

#### 33

#### DRIVER CODE:

int main()

{

int a = 65;

int b = 32;

int \*pa = &a, \*pb = &b;

update(pa, pb);

cout<<a<<endl;

cout<<b<<endl;

return 0;

}

COMPLETE CODE

#include <iostream>

using namespace std;

void update(int \*a,int \*b)

{

    int p=\*a;

    \*a=(\*a)+(\*b);

    int k=p-(\*b);

    if(k<0)

    {

     k=k\*(-1);

    }

    \*b=k;

}

int main()

{

int a = 65;

int b = 32;

int \*pa = &a, \*pb = &b;

update(pa, pb);

cout<<a<<endl;

cout<<b<<endl;

return 0;

}