Name:- Gaurav Tewari

Section:- F
Roll no:- 30
Uni. Roll no:- 2014659

Q1:

Answer:-

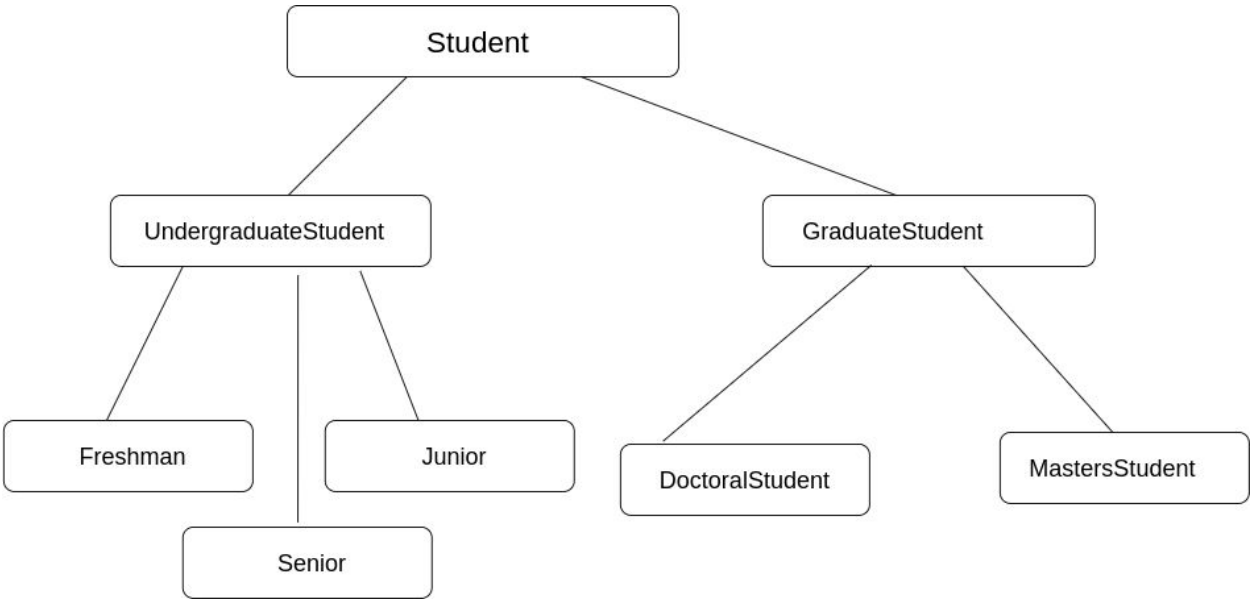
Inheritance supports the concept of code reusability in the program
If we want to create a Class new class, but suppose there is already a class which does the same work then we can Inherit the same class and it saves us a lot of time and promotes code reusability of the code.

For example here Dog class will be Inherited what to do from pets... And we don't have to write it again .. this will save our time and promote code reusability.

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

class Pets {
public:
	void Whattodo() {
		cout << "Eat sleep Repeat" << "\n";
	}
};
class Dogs : public Pets {
public:
	void Woff() {
		cout << "Woof! Let's go for a walk Human!!" << "\n";
	}
```

```
};
int main() {


	Dogs Edgar;
	Edgar.Woff();
	Edgar.Whattodo();
}
```

Q2:



Hierarch

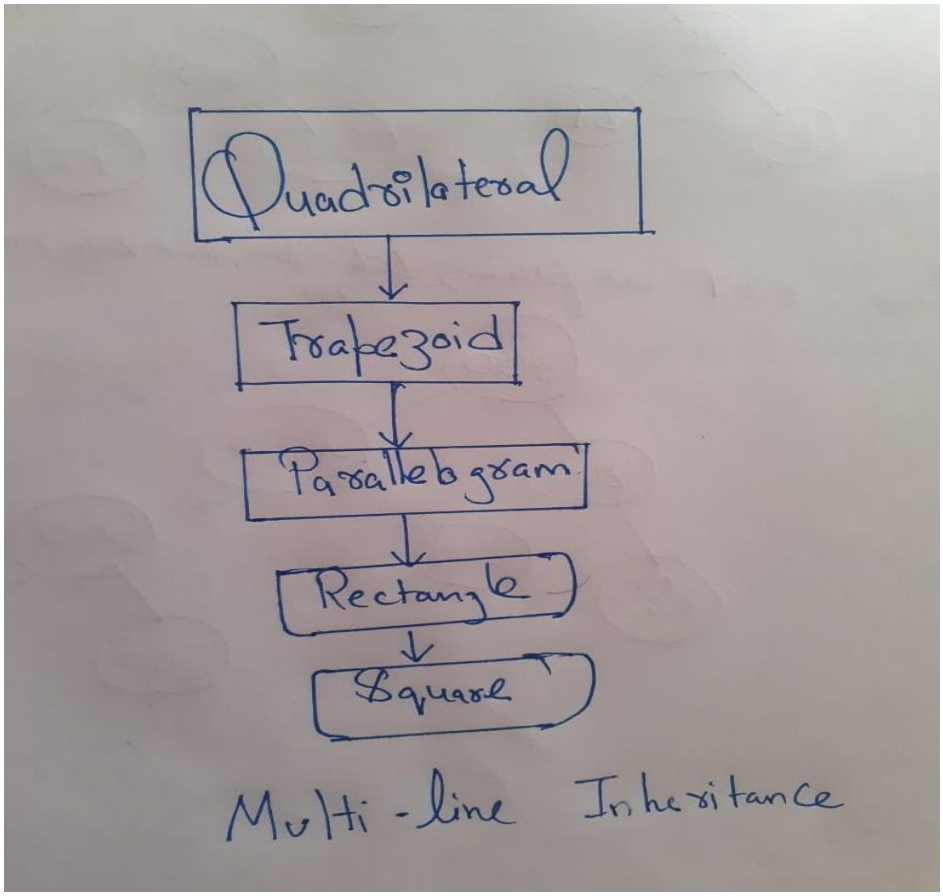
Here , Senior, junior, and freshman have inherited from the class Undergraduate student and the undergraduate student have inherited from the class student.
Similarly Doctoral Student class and Masters student have inherited from Graduate student and Graduate student have also inherited from the Student class.

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

```
class Student {
public:
	void Study() {
		cout << "I will study after 2 hours" << "\n";
	}
};
class UndergraduateStudent: public Student {
public:
	void Btech() {
		cout << "I am doing Btech from GEU" << "\n";
	}

};
class Freshman: public UndergraduateStudent {
public:
	void Motivation() {
		cout << "i will score 10CGPA this sem" << "\n";
	}
};
class Junior: public UndergraduateStudent {
public:
	void coding() {
		cout <<  "My program works.. but i don;t know why?" << "\n";
	}
};
class Senior: public UndergraduateStudent {
public:
	void coding() {
		cout <<  "What am i doing here?" << "\n";
	}
};
class GraduateStudent: public Student {
public:
	void Knowledge()
	{
		cout << "I want to get more Knowledge" << "\n";
	}

};
class MastersStudent : public GraduateStudent {
public:
	void Master() {
		cout << "I will master this subject" << "\n";
```

```
	}
};
class DoctoralStudent : public GraduateStudent {
public:
	void Doctoral() {
		cout << "I will research new things in this subject" << "\n";
	}
};

int main() {

	Junior Gaurav;
	Gaurav.Btech();
	Gaurav.coding();
	Gaurav.Study();

	DoctoralStudent Tewari;
	Tewari.Doctoral();
	Tewari.Study();

}
```

Q3:




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



class Quadrilateral {
public:
	void propertyofQuard() {
		cout << "A quadrilateral should be closed shape with 4 sides. All the internal angles of a quadrilateral sum up to 360°" << "\n";
	}
};
class Trapezoid: public Quadrilateral {
public:
	void propertyoTrapezoid() {
		cout << "A Trapezoid have parallel sides" << "\n";
	}
```

```
	//sum of parallel side... divided by 2 multipiled by height
	int areaTrapezoid(int a, int b, int h) {
		return (a + b) * h / 2;
	}
	int perimeterTrapezoid(int s1, int s2, int s3, int s4)
	{
		return s1 + s2 + s3 + s4;
	}
};
class Paralleologram : public Trapezoid {
public:
	void propertyofParalleogram() {
		cout << "A paralleologram have pair of parallel and equal sides" << "\n";
	}
	int areaPalleogram(int base, int height)
	{
		return base * height;
	}
	int perimeterPalleogram(int a, int b)
	{
		return 2 * (a + b);
	}
};
class Rectangle : public Paralleologram {
public:
	void propertyofRectangle() {
		cout << "It have a pair of parallel and equal sides and all angles are of 90deg."
<< "\n";
	}
	int areaRectangle(int a, int b)
	{
		return a * b;
	}
	int perimeterRectangle(int a, int b)
	{
		return 2 * (a + b);
	}
};
class Square : public Rectangle {

public:
	void propertyofSquare() {
		cout << "All sides of Square are equal and all angles of 90 deg" << "\n";
	}
```

```
	int areaSquare(int s) {
		return s * s;
	}
	int perimeterSquare(int s) {
		return 4 * s;
	}
};
int main() {


	Square cool;
	cool.propertyofQuard();
	cout << "area using Square with side 5 " << cool.areaSquare(5) << "\n";
	cout << "perimeter " << cool.perimeterSquare(5) << "\n";
	cout << "finding area of same Square using Rectangle's formula " <<
cool.areaRectangle(5, 5) << "\n";
	cout << "finding area of same Square using Paralleologram 's formula " <<
cool.areaPalleogram(5, 5) << "\n";
	cout << "finding area of same Square using Trapezoid 's formula " <<
cool.areaTrapezoid(5, 5, 5) << "\n";
	cout << "thus the hierarchy is proved";


}
```

Q4:

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

class shape {
public:
	void propertiesOfshape() {
		cout << "A shape is the form of an object or its external boundary, outline, or
external surface, as opposed to other properties such as color, texture or material" << "\n";
	}
};
class TwoDshape : public shape {
public:
	void propertiesof2d() {
		cout << "Two D shape just have Length and Breath" << "\n";
	}
```

```
};
class Square : public TwoDshape {
public:
	int calculateArea(int s) {
		return s * s;
	}
};
class Rectangle : public TwoDshape {
public:
	int calculateArea(int a, int b) {
		return a * b;
	}
};
class ThreeDshape: public shape {
public:
	void propertiesof3d() {
		cout << "a 3-d shape has Length ,Breath and height" << "\n";
	}
};
class Sphere : public ThreeDshape {
public:
	int  areaofSpehere(int radius) {
		return 4 / 3 * 3.17 * radius * radius * radius;
	}
};


int main() {
	Sphere Ball;
	cout << Ball.areaofSpehere(5) << "\n";
	Square Something;
	cout << Something.calculateArea(5) << "\n";


}
```

Q5:

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

class base
{
```

```
public:
  int x;
};
class num1 : public base
{
public:
  num1()      //constructor to initialize x in base class num1
  {
    x = 10;
  }
};
class num2
{
public:
  int y;
  num2()   //constructor to initialize num2
  {
    y = 4;
  }
};
class num3 : public num1, public num2   //num3 is derived from class num1 and class num2
{
public:
  void sum()
  {
    cout << "Sum= " << x + y;
  }
};

int main()
{



  num3 obj1;          //object of derived class num3 which is sum
  obj1.sum();
  return 0;
}
```