Create a polymorphic class Vehicle and create other derived classes Bus, Car and Bike from Vehicle. With this program illustrate RTTI by the use of dynamic_cast and typeid operators.

```
#include <iostream>
#include <cstring>
#include <typeinfo>
using namespace std;
class vehicle
{
 private:
 protected:
  string registration;
  int noofwheels;
 public:
  vehicle(string r, int n)
  {
    registration = r;
    noofwheels = n;
  }
  string getregistration()
  {
    cout << "Vehicle getRegistratin called" << endl;
    return registration;
  }
};
class bus : public vehicle
{
 private:
 public:
  bus(string r):vehicle(r,4){};
  string getregistration()
  {
    cout << "Bus getRegistratin called" << endl;
    return registration;
  }
};
class car : public vehicle
{
 private:
```

```
 public:
  car(string r):vehicle(r,4){};
  string getregistration()
  {
    cout << "Car getRegistratin called" << endl;
    return registration;
  }
};
class bike : public vehicle
{
 private:
 public:
  bike(string r):vehicle(r,2){};
  string getregistration()
  {
    cout << "Bike getRegistratin called" << endl;
    return registration;
  }
};

int main()
{
  vehicle *vlist[3];
  bus *bs = new bus("1");
  car *c = new car("1");
  bike *b = new bike("1");
  vlist[0] = dynamic_cast<vehicle *>(bs);
  vlist[1] = dynamic_cast<vehicle *>(c);
  vlist[2] = dynamic_cast<vehicle *>(b);
  for(int i = 0; i < 3 ; i++)
  {
    cout << typeid(*vlist[i]).name() << endl;
    cout << vlist[i]->getregistration() << endl;
  }
  cout << typeid(*bs).name() << endl;
  cout << typeid(*c).name() << endl;
  cout << typeid(*b).name() << endl;
  return 0;
}
```

```
#include <iostream>//or
#include <cstring>
#include <typeinfo>
using namespace std;
class Vehicle
{
protected :
   string vec_name;
   int no_of_wheels;
public:
   Vehicle(string name, int wheels)
   {
      vec_name=name;
      no_of_wheels=wheels;
   }
   void display()
   {
      cout<<"From Vehicle:"<<endl;
      cout<<"name ="<<vec_name<<endl<<"no of
wheels="<<no_of_wheels<<endl<<endl;
   }
};
class Bus:public Vehicle
{
   int bus_id;
public:
   Bus (string name, int wheels,int id):Vehicle(name,wheels)
   {
      bus_id=id;
   }
   void display()
   {
      cout<<"From Bus:"<<endl;
      cout<<"name ="<<vec_name<<endl<<"no of
wheels="<<no_of_wheels<<endl<<"bus id ="<<bus_id<<endl<<endl;
   }
};
class Car:public Vehicle
{
   int car_id;
```

```
public:
   Car (string name, int wheels,int id):Vehicle(name,wheels)
   {
      car_id=id;
   }
   void display()
   {
      cout<<"From Car:"<<endl;
      cout<<"name ="<<vec_name<<endl<<"no of
wheels="<<no_of_wheels<<endl<<"car id ="<<car_id<<endl<<endl;
   }
};
class Bike:public Vehicle
{
   int bike_id;
public:
   Bike (string name, int wheels,int id):Vehicle(name,wheels)
   {
      bike_id=id;
   }
   void display()
   {
      cout<<"From Bike:"<<endl;
      cout<<"name ="<<vec_name<<endl<<"no of
wheels="<<no_of_wheels<<endl<<"bike id ="<<bike_id<<endl<<endl;
   }
};
int main()
{
   Vehicle *v[3];
   Bus *bus =new Bus("Tata",6,123);
   Car *car = new Car("Tesla",4,342);
   Bike *bike = new Bike("ninja",2,898);
   cout<<"Before type casting:"<<endl;
   bus->display();
   car->display();
   bike->display();
   v[0]=dynamic_cast<Vehicle *>(bus);
   v[1]=dynamic_cast<Vehicle *>(car);
   v[2]=dynamic_cast<Vehicle *>(bike);
```

```
    cout<<"After type casting:"<<endl;
    for(int i=0;i<3;i++)
    {
       v[i]->display();
       cout<<endl;
    }
    cout<<"Type ID:"<<endl;
    cout<<typeid(*bus).name()<<endl;
    cout<<typeid(*car).name()<<endl;
    cout<<typeid(*bike).name()<<endl;
}
```