POLYMORPHISM – VIRTUAL FUNCTION, PURE VIRTUAL FUNCTION AND ABSTRACT CLASS

1. Virtual Function

```
#include <iostream>
using namespace std;
// Declaring a Base class
class NBase {
public:
	// virtual function
	virtual void display()
	{
		cout << "Called virtual Base Class function"
			<< "\n\n";
	}
	void print()
	{
		cout << "Called NBase print function"
			<< "\n\n";
	}
};
// Declaring a Child Class
class NChild : public NBase {
public:
	void display()
	{
		cout << "Called NChild Display Function"
			<< "\n\n";
	}
	void print()
	{
```

```
                cout << "Called NChild print Function"
                        << "\n\n";
        }
};
int main()
{
        // Create a reference of class NBase
        NBase* base;
        NChild child;
        base = &child;
        // This will call the virtual function
        base->NBase::display();
        // this will call the non-virtual function
        base->print();
}
```

Output:

Called virtual Base Class function

Called NBase print function

2. Pure Virtual Function and Abstract Class

```
#include <iostream>
using namespace std;
class Base {
        // private member variable
        int x;
public:
        // pure virtual function
        virtual void fun() = 0;
        // getter function to access x
```

```
        int getX() { return x; }
};
// This class inherits from Base and implements
fun() class Derived : public Base {
        // private member variable
        int y;
public:
        // implementation of the pure virtual
        function void fun() { cout << "fun() called";
        }
};
int main(void)
{
        // creating an object of Derived class
        Derived d;
        // calling the fun() function of Derived
        class d.fun();
        return 0;
}
```

Output:

```
fun() called
```