TRIBHUVAN UNIVERSITY

INSTITUTE OF ENGINEERING

PULCHOWK CAMPUS

LAB REPORT ON

OBJECT ORIENTED PROGRAMMING [CT 451]

LAB 7

POLYMORPHISM IN C++

Submitted by:

Rujal Acharya

PUL076BEI029

Submitted to:

Department of Electronics and Computer Engineering, Pulchowk Campus

Institute of Engineering, Tribhuvan University

Lalitpur, Nepal

December, 2020

Task 1

Problem:

WAP in CPP to illustrate the concept of re_interpret cast operator.

Program:

```
#include <iostream>

class Rectangle {
	public:
		void getData ( );

	private:
		int length;
		int breadth;
};

class Parallelogram {
	public:
		void showData ( );

	private:
		int length;
		int height;
};

void Rectangle::getData ( ) {
	std::cout << "Enter the length and breadth of rectangle: " << std::endl;
	std::cin >> this->length >> this->breadth;
}

void Parallelogram::showData ( ) {
	std::cout << "Length of parallelogram: " << this->length << std::endl
			<< "Height of parallelogram: " << this->height << std::endl;
}

int main ( ) {
	Rectangle *rect;
	rect->getData ( );
	Parallelogram *par = reinterpret_cast <Parallelogram *> (rect);
	par->showData ( );
	return EXIT_SUCCESS;
}
```

Task 2

Problem:

WAP in CPP to illustrate the concept of dynamic cast operator.

Program:

```
#include <iostream>

class Parent {
	public:
		virtual void temp ( ) { }
		Parent ( );
};

class Child : public Parent {
	public:
		Child ( );
};

Parent::Parent ( ) {
	std::cout << "Inside parent class constructor..." << std::endl;
}

Child::Child ( ) {
	std::cout << "Inside child class constructor..." << std::endl;
}

int main ( ) {
	Parent *p = new Child ( );
	Child *c = dynamic_cast <Child *> (p);
	if (c != NULL) {
		std::cout << "Successfully downcasted..." << std::endl;
		return EXIT_SUCCESS;
	} else {
		std::cerr << "Error occured while casting..." << std::endl;
		return EXIT_FAILURE;
	}
}
```

Task 3

Problem:

WAP in CPP to illustrate the concept of typeid operator.

Program:

```
#include <iostream>

class Base {
	public:
		virtual void temp ( ) { }
		Base ( );
};

class Complex : public Base {
	public:
		Complex ( );

	private:
		int real, imz;
};

class Rectangle {
	public:
		Rectangle ( );

	private:
		float length, breadth;
};

Base::Base ( ) {
	std::cout << "Inside base class..." << std::endl;
}

Complex::Complex ( ) {
	std::cout << "Inside complex class..." << std::endl;
}

Rectangle::Rectangle ( ) {
	std::cout << "Inside rectangle class..." << std::endl;
}

int main ( ) {
	int i, *iptr;
	float f, *fptr;
	char ch, *chptr;
	Base b, *bptr;
	Complex cmp, *cmptr;
	Rectangle r, *rptr;
```

```
    std::cout << "Lvalue of: " << std::endl
                    << "Integer: " << typeid(i).name( ) << std::endl
                    << "Integer pointer: " << typeid(iptr).name( ) << std::endl
                    << "Float: " << typeid(f).name( ) << std::endl
                    << "Float pointer: " << typeid(fptr).name( ) << std::endl
                    << "Character: " << typeid(ch).name( ) << std::endl
                    << "Character pointer: " << typeid(chptr).name( ) << std::endl
                    << "Base class: " << typeid(b).name( ) << std::endl
                    << "Base class pointer: " << typeid(bptr).name( ) << std::endl
                    << "Complex class: " << typeid(cmp).name( ) << std::endl
                    << "Complex class pointer: " << typeid(cmptr).name( ) << std::endl
                    << "Rectangle class: " << typeid(r).name( ) << std::endl
                    << "Rectangle class pointer: " << typeid(rptr).name( ) << std::endl;

    return EXIT_SUCCESS;
}
```

Task 4

Problem:

WAP in CPP to illustrate the concept of virtual functions.

Program:

```
#include <iostream>

class College {
	public:
		virtual void getData ( ) = 0;
		virtual void showData ( ) = 0;
		virtual ~ College ( ) { }

	protected:
		std::string name, address;
};

class Student : public College {
	public:
		void getData ( );
		void showData ( );

	private:
		int roll;
};

class Teacher : public College {
	public:
		void getData ( );
		void showData ( );

	private:
		int id;
};

void Student::getData ( ) {
	std::cout << "Enter student name: ";
	std::cin >> this->name;
	std::cout << "Enter student address: ";
	std::cin >> this->address;
	std::cout << "Enter student roll no: ";
	std::cin >> this->roll;
}

void Student::showData ( ) {
	std::cout << "Student name: " << this->name << std::endl
			<< "Student address: " << this->address << std::endl
			<< "Student roll no: " << this->roll << std::endl;
}
```

```
void Teacher::getData ( ) {
	std::cout << "Enter teacher name: ";
	std::cin >> this->name;
	std::cout << "Enter teacher address: ";
	std::cin >> this->address;
	std::cout << "Enter teacher id no: ";
	std::cin >> this->id;
}

void Teacher::showData ( ) {
	std::cout << "Teacher name: " << this->name << std::endl
			<< "Teacher address: " << this->address << std::endl
			<< "Teacher id no: " << this->id << std::endl;
}

int main ( ) {
	College *college;
	college = new Student ( );
	college->getData ( );
	college->showData ( );
	delete college;
	college = new Teacher ( );
	college->getData ( );
	college->showData ( );
	delete college;
	return EXIT_SUCCESS;
}
```