# Question 3 - Polymorphism

In this file we have multilevel inheritances which have been used in order to implement Polymorphic behaviour. We have the following lines:

* a \*one = new a();

When the above line is executed, the pointer **\*one** is assigned to an object of class **a**. The constructor of class **a** is called and since we have

class a {

public:

a(){ print();}

virtual void print() {

printf("print\_a\n");

}

};

The output would be “print\_a”. In this line, no polymorphic behaviour is observed as the pointer **\*one** is of type **a** and is pointing to an object of type **a**.

* a \*two = new b();

This line exhibits polymorphic behaviour as the pointer **\*two** is of type **a** but is pointing to an object of type **b** (which is a class derived from **a**). When the **new** operator creates an instance of **b**, first the constructor of **a** is called, then constructor of **b** is called because **b** is derived from **a**. Hence, the output will be:

“print\_a”

“print\_b”

* a \*three = new c();

This is also polymorphic as **\*three** is pointing to an object of type **c**, which is a descendant of **a**. Class **c** is derived from **b**, which in turns is derived from **a**, hence, constructor of **a** will be called first, then **b**’s constructor and finally **c**’s constructor. Thus, when this line will be executed, the output will be:

“print\_a”

“print\_b”

“print\_c”

* c \*four = dynamic\_cast<c\*>(three);

In this line the pointer **\*three**, which is of type **a**,has been downcasted to a pointer of type **c**. This is allowed because **c** is a descendant of **a**. Since this is just a usual pointer assignment, no constructor or function has been called, so this line will not print anything.

* one->print();

Pointer **\*one** is of type **a** and points to an object of type **a**, hence, the compiler will call **a**’s print() function. The output is therefore:

“print\_a”

* two->print();

Pointer **\*two** is of type **a** and points to an object of type **b**. Since in **a** the print() function has been declared virtual, **b**’s version of print() will be called, hence the output is:

“print\_b”

* three->print();

Pointer **\*three** is of type **a** but its pointing to an object of type **c**. Because both **a**’s and **b**’s print() functions are virtual, this line will call **c**’s version of print(). Hence, the output is:

“print\_c”

* four->print\_loop(3);

The pointer **\*Three** (type **a**) has been downcasted to a pointer of type **c** using a dynamic cast and stored in the pointer **\*four**. The comments explain which line prints what:

void print\_loop(int count){

for(int i=0; i < count; i++){

print(); //this line will always call c's print()

class a\* item = new a(); //this line will always call a's print()

item->print();

}

}

Hence, since count = 3, the output would be

“print\_c”

“print\_a”

Repeated for 3 times