**Midterm Project**

**For each of the problems described below, explain which paradigm (Object-Oriented or Logical) would best be suited to create the solution. Be sure to explain your reasoning.**

**1. You require a program that is fast, and simple to model.**

Object-Oriented program would be the best choice in this scenario because it improves the software maintainability, productivity, and produces high quality software. Object-Oriented programs have several libraries of objects that makes coding fast as well as simple to model.

**2. You are creating a complex model of real-world phenomena.**

Although not in every aspect of reality but realistically a programmer can use object-oriented programs to capture some aspects of real-world. Take virtual reality for example, most AR/VR development is done using c#, c/c++. These programming languages encompasses strong functional, declarative, imperative, as well as object-oriented disciplines.

**3. You are dealing with complex data structures.**

Logical programming languages are not suitable to handle complex data structures. Take prolog for example, although it’ll be sufficient to represent any other data structure it is not efficient. Modifying complex data structures in logical programs can be very expensive. In other words Object-Oriented would be best suitable to handle complex data structures.

**4. You are working with a large database of facts.**

Logical programming languages would be the best choice because programs created using logic languages can be used as true/false statements rather than objects.

**5. You can infer a solution through provable relations.**

In the logic programming model, the programmer is responsible for specifying the basic logical relationship, thus not specifying the manner in which the interference rules are applied. An example of relations in logic programs: Male(X), Siblingof(X,Y), and Parentof(Y,Z). We can infer the obvious relations therefore can define the predicate uncleof(X,Z). The implementation are as follows:

uncleof(X,Z)←-male(X), sibling(X,Y), parentof(Y,Z).

**What is the difference between Declarative Programming and Imperative Programming? What are their strengths & weaknesses? Why use one over the other? Be sure to explain your reasoning.**

Imperative programming is writing a program to execute HOW you want to do something. An example can be calculating a factorial using recursion. In the function below we see that if you read the code line by line you tend to understand exactly how the code will work.

int factorial(int number)

{

if(number > 1)

return number \* factorial(number-1)

else

return 1;

}

With a few line of code imperative programming is easy to follow, however it can be non scalable and very convoluted when it is a large code.

Declarative programming on the other hand is a program that tells you WHAT you want to accomplish. Declarative programming is simpler and safe to use. It is used in most cases as a means to interact with code. An example could be clicking a button:

class Button extends React.Component{

this.state = { color: 'red' }

handleChange = () => {

const color = this.state.color === 'red' ? 'blue' : 'red';

this.setState({ color });

}

render() {

return (<div>

<button

className=`btn ${this.state.color}`

onClick={this.handleChange}>

</button>

</div>);

}

}

When a programmer thinks declaratively he/she always know what the outcome would be before implementing a code, which sets up for a good control flow which can be predicted.