17-355/17-665/17-819 Program Analysis
Program Synthesis Exercises

April 16, 2019

Name: _____

Andrew ID: _____

These exercises provide an introduction to **SyPet**, a program synthesizer for Java libraries that automatically constructs programs by composing APIs.

1 From Java libraries to Petri nets

SyPet uses a Petri net representation of Java libraries. Suppose you have the following two Java classes:

```
class Point {
  Point();
  int getX();
  int getY();
  void setX(int);
  void setY(int);
}
```

```
class MyPoint {
  MyPoint(int x, int y);
  int getX();
  int getY();
}
```

Suppose you want to synthesize a method `convert` that converts the object `MyPoint` to `Point`:

```
Point convert(Mypoint pt){

    // to be synthesized

}
```

Andrew ID: _____

► Build the Petri net representation of the `Point` and `MyPoint` classes.

► What are the places and transitions of the Petri net?

► What is the initial marking in the Petri net to synthesize the `convert` method?

► What is the final marking in the Petri net to synthesize the `convert` method?

Andrew ID: _____

2 Petri net reachability

A reachable path in a Petri net is a *sequence of transitions* that starts from the *initial marking* and ends in the *final marking*.

► Find a reachable path that is **not** the solution to the `convert` method.

► Find a reachable path that corresponds to the solution of the `convert` method.

Andrew ID: _____

3 Sketch completion

A reachable path in the Petri corresponds to a program sketch since it does not specify arguments for each method call.

► Write the code for the `convert` method with holes that correspond to the following sequence of API calls:

```
int getX(MyPoint)
int getY(MyPoint)
Point Point()
void setX(Point, int)
void setY(Point, int)
```

► Assume that each variable must be used at least once. How many concrete programs can be created from this sketch?