CSE 464: Software Quality Assurance and Testing
Arizona State University
Midterm II – Practice Questions
(Solutions will be discussed during the review session on Wednesday April 2nd)

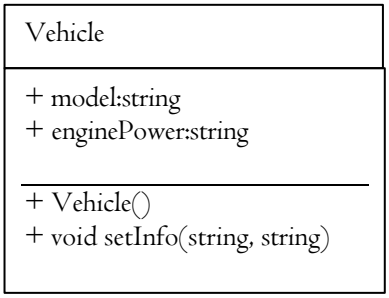
Multiple Choice

a) According to C-K matrix , a good OO design has more depth in inheritance hierarchy as it encapsulates functionality at different levels ..... (T/F)

b) Code inspection is more formal than walkthrough as the inspection is carried out by software professionals outside of the software development team…. .... (T/F).

c) In general, module design produce better software quality. Which of the following is true about modular deign
   A) High coupling and low cohesion are desirable in modular design
   B) High coupling and high cohesion are desirable in modular design
   C) Low coupling and low cohesion is desirable in modular design
   D) Low coupling and high cohesion is desirable in modular design

d) What is main issue with the following object design?



   a) High Coupling b) low cohesion c) No Information hiding d) None of the above

2. **Coupling and Cohesion** Consider the following code that you need to evaluate it for cohesiveness. Explain how we can make the following object more cohesive. **Rewrite** the code based on your answer.

```
public class A
{
  private int x, y, p, q;
  public A()
  {
     x=y=p=q=0;
  }

  public void setXY(int a , int b)
  {
     x=2*a;
     y=b;
  }
   public void setPQ(int a , int b)
    {
       p=3*a;
       q=b/2;
    }
  public int diff()
  {
      return (x*x - y);
  }

  public int sub()
  {
    return y*x - x;
  }
  public int mul()
  {
    return p*q;
  }

 public int div()
 {
    return (p*p)/q;
 }
}
```

Explanation:

Re-Written Code:

3. The following class set has **many symptoms** of bad coupling. Identify type of coupling in the following code. You can mark the code section and that has coupling and name the type of coupling exists.

```
class TempMonitor
{
        public  double  temp;
        public  void setTemp(double Temp)
        {
                temp=Temp;
        }

};
```

```
class Wind
{
  public int speed;
  public Wind(int sp)
  {
        speed=sp;

  }
};
```

```
class Weather
{
        public String weatherCondition;


        public Weather(TempMonitor t, Wind w)
        {
                if(t.temp > 75 && w.speed < 25)
                  weatherCondition="Best";
                 else if(t.temp < 75 && w.speed < 25)
                      weatherCondition="Good";
                    else
                    weatherCondition="Bad";
        }
};
```

```
class suggestSport
{
        public static void main(String[] args)
        {
     String sport;
    TempMonitor t=new TempMonitor();
    t.setTemp(78.5);
    Wind w= new Wind(45);
    Weather W=new Weather(t,w);

        if(W.weatherCondition.equals("Best"))
            sport="Swimming";
            else if( W.weatherCondition.equals("Good"))
            sport="Tennis";
             else
              {
                w.speed=20;
                t.temp=65;
         }
  }
 }
```

4. Data Path Testing : Draw the control flow graph for the following SequentialSearch method. Then design test cases to test the SequentialSearch algorithm implementation.

```
int  SequentialSearch(int[] a, int searchItem)
{
int itemIndex = -1; Boolean found= false;
for(location = 0 ; location < a.length ; location++)
    {
       if(a[location] = = searchItem)
        {
          found = true;
          itemIndex = location;
          break;
        }
    }
     return itemIndex;
}
```

a) Draw Dataflow graph and then identify D-U pairs and D-U paths for following variables in the above code segment.

itemIndex
found

b) Design test cases based on D-U paths identified.