Name: ID:

Question 1

```
public class Quiz3A{
  public static int temp = 3;
  public int sum;
  public int y;
  public Quiz3A(){
    y = temp - 1;
    sum = temp + 1;
    temp+=2;
  }
  public Quiz3A(int k){
    temp = temp++;
    sum = ++temp + k;
    y = sum - 3;
  }
  public int methodB(int m, int n){
    int x = 1;
    y = y + m + (temp++);
    x = x + 2 +  n;
    sum = sum + x + y;
    System.out.println(x + " " + y+ " " + sum);
    return sum;
  }
}
```

Consider the following code:

```
Quiz3A a1 = new Quiz3A();
a1.methodB(2,4);
Quiz3A a2 = new Quiz3A(2);
Quiz3A.temp += 2;
a2.methodB(1,2);
a1.methodB(1,1);
a2.methodB(2,1);
```

x	y	sum

Question 2

Assume that the following classes have been defined:

```
class Foo {
  public void method1() {
    System.out.println("foo method 1");
  }

  public void method2() {
    System.out.println("foo method 2");
  }
}

class Bar extends Foo {
  public void method2() {
    System.out.println("bar method 2");
  }

  public void method3() {
    System.out.println("bar method 3");
  }
}

class Baz extends Foo {
  public void method1() {
    System.out.println("baz method 1");
  }

  public void method2() {
    System.out.println("baz method 2");
    method1();
  }
}

class Mumble extends Baz {
  public void method1() {
    super.method1();
    System.out.println("mumble method 1");
  }

  public void method3() {
    System.out.println("mumble method 3");
  }
}
```

And assume that the following variables have been defined:

```
Foo foo1 = new Bar();
Foo foo2 = new Baz();
Bar bar1 = new Bar();
Baz baz1 = new Baz();
Baz baz2 = new Mumble();
Object obj1 = new Bar();
```

In the table below, indicate in the right-hand column the output produced by the statement in the left-hand column. If the statement produces more than one line of output, indicate the line breaks with slashes as in "a/b/c" which indicates three lines of output with "a" followed by "b" followed by "c". If the statement causes an error, fill in the right-hand column with either "CT" for "compile time error" or RE for "runtime error" to indicate when the error would be detected.

Statement	Output
foo1.method1();	
foo2.method1();	
bar1.method1();	
baz1.method1();	
baz2.method1();	
obj1.method1();	
foo1.method2();	
foo2.method2();	
bar1.method2();	
baz1.method2();	
baz2.method2();	
((Baz)obj1).method2();	
((Object)bar1).method3();	
((Foo)baz2).method3();	
((Bar)foo1).method3();	
((Mumble)baz1).method3();	
((Mumble)baz2).method3();	
((Baz)foo2).method3();	
((Bar)foo2).method2();	
((Foo)obj1).method2();	

Question 3

Write the output of the following code:

```
public class ParentException extends Exception{
  protected String msg;
  public ParentException(Object o){
    msg = o.toString();
  }
  public String toString() {
    return "Parent: "+ msg;
  }
}
```

```
public class ChildException extends ParentException{
  public ChildException(Object o){
    super(o);
  }
  public String toString() {
    return "Child: "+ msg;
  }
}
```

```
public class BadThing{
  private String msg = null;
  private int num;
  public BadThing (int i){
    num = i;
  }
  public void badMethod() throws Exception{
    if (num == 7) {
      throw new NullPointerException ();
    }
    if (num < 11){
      throw new ArrayIndexOutOfBoundsException ();
    }
    if (num%2 == 0){
      throw new ParentException(new BadThing(num));
    }else{
      throw new ChildException(new BadThing(num));
    }
  }
  public String toString(){
    if (num%2 == 0){
      return "You are an Even Steven "+ num;
    }else{
      return "You are an Odd One "+ num;
    }
  }
}
```

```
public class GoodThing{
  public static void goodMethod(Object o) throws Exception{
    try{
      ((BadThing)o).badMethod();
    }catch(ChildException c){
      System.out.println("goodThing: "+c);
    }catch(Exception e){
      throw(e);
    }finally {
      System.out.println("finally I am at goodMethod. ;)");
    }
  }
  public static void main(String [] args){
    int i =  0;
    for (i =7; i < 17; i+=3){
      try{
        goodMethod(new BadThing(i));
      }catch(ParentException p){
        System.out.println("main: "+p);
      }catch(RuntimeException e){
        System.out.println("Boo Hoo! I could stop it.");
      }catch(Exception e){
        System.out.println("Boo Hoo! I could not stop it.");
      }
    }
  }
}
```

Output

Question 4

```
class A{
  public static int temp = 2;
  public int sum;
  public int y;
  public A(int x){
    y = temp - 2 + x;
    sum = temp + 2;
    temp-=2;
  }
  public void methodB(int m, int n){
    int x = 1;
    y = y + m + (temp++);
    x = x + 2 +  n;
    sum = sum + x + y;
    System.out.println(x + " " + y+ " " + sum);
  }
}
class B extends A {
  public int x;
  public int sum;
  public B(int p){
    super(p);
    y = temp + p ;
    temp-=1;
  }
  public B(B b){
    super(b.sum);
    sum = b.sum;
    x = b.x;
  }
  public void methodB(int m, int n){
    int  y =0;
    y = y + this.y;
    x = this.y + 2 + temp;
    super.methodB(x, y);
    sum = x + y + super.sum;
    System.out.println(x + " " + y+ " " + sum);
  }
}
```

What is the output of the following code generation?

```
A a1 = new A(2);
B b1 = new B(3);
B b2 = new B(b1);
a1.methodB(1, 1);
b2.methodB(1, 2);
b2.methodB(2, 2);
```

x	y	sum

Question 5

Write a function called **findPrimeFactors** that will print how many times each prime number appears in a given number. If we factorize the number 24, we get 2 x 2 x 2 x 3. That is we get the number 2, three times and the number 3 only one time.

The function will look like the following:

```
public class A {
        public static void findPrimeFactors ( int n) {
                // your code will go here …
```

```
        }
}
```

Example 1) If given 24, the output is like following:

2 appears 3 time (s)
3 appears 1 time (s)

Example 2) If given 21, the output is like following:

3 appears 1 time (s)
7 appears 1 time (s)

You program should work for any number between 2 and 100. We will use a program similar to the following program to test your code:

```
public class Driver {
        public static void main (String [] args) {
                A.findPrimeFactors (21);
                A.findPrimeFactors (24);
                A.findPrimeFactors (100);
        }
}
```

Question 6

Study the following Code and Output:

Code	Output

`public class StudentTest{`
`  public static void main(String [] args){`
`  Student s1 = new Student();`
`  System.out.println(s1.getName());`
 	
`  Student s2 = new Student("Matin");`
`  System.out.println(s2.getName());`
 	
`  Student s3 = new Student("Saad");`
`  System.out.println(s3.getName());`
 	
`  System.out.println(Student.numberOfStudents);`
`  }`
`}`
	`default name`
	`Matin`
	`Saad`
	`3`

Now write **Student** class in such a way that the above output is created.