MULTIPLE CHOICE QUESTIONS 1-10. (4 points each)
For questions 1-2 below, use the following partial class definitions:

```
public class C1
{
     public int a;
     private int b;
     protected int c;
...
}

public class C2 extends C1
{
     protected int x;
     public int y;
...
}

public class C3 extends C2
{
     private int z;
...
}
```

1. Which of the following is true with respect to C1, C2 and C3?

   a. C3 is a superclass of C2 and C2 is a superclass of C3
   b. C1 and C2 are both subclasses of C3
   c. C2 and C3 are both subclasses of C1
   d. C3 is a subclass of C2 and C2 is a subclass of C1
   e. None of the above

2. Which of the following lists of instance data are accessible in class C2?

   a. x, y, z, a, b
   b. x, y, a
   c. x, y, a, c
   d. a, y
   e. None of the above

3. In the following array, what is the value of `stats.length`?

```
double[][] stats={{3.00, 3.50},{6.35,3.4},{2.7, 7.35, 8.35, 9.00}};
```

   a. 1
   b. 2
   c. 4
   d. 8
   e. None of the above

4. Given the classes below, what is the output of the following program execution:

```
public class PosNumberException extends
Exception
{

//-----------------------------------------------------------------
//Sets up exception object with a particular
//message.
public PosNumberException (String message)

{

super (message);
}

}
```

```
public class Tester
{

static public void main (String[] args)
{

String[] numbers={"57", "fifty", "-50", "39"};
PosNumberException problem=new PosNumberException("positive ");

for (int i=0;i<numbers.length;i++)
{
  try{
       if (Integer.parseInt(numbers[i])>0) throw problem;
       System.out.print("negative ");

      }

  catch (PosNumberException e){
       System.out.print(e.getMessage());

       }
  catch (NumberFormatException e){
       System.out.print("none ");

       }
}
}
}
```

a. none negative
b. positive none negative positive
c. negative none positive negative
d. positive positive
e. None of the above

5. Show the output of the following code:

```
public class Test5 {
public static void main(String[] args) {
      int[] x = {1, 2, 3};
      int[] y = {3, 2, 3};

      modArrays(x[0],y);
      System.out.println(x[0] + " " + y[0]);
      }

public static void modArrays(int a, int[] b) {
      for (int i=0;i<b.length;i++)
            b[i]+=a;
      a=b[2];
      }
}
```

a. 1 3
b. 4 4
c. 3 4
d. 2 4
e. None of the above

6. If the method is invoked as `rose(4)`, what is returned?

```
public static int rose(int n)
{
  if ( n <= 0)
      return 1;
  else
     return ( rose (n-1) * n);
}
```

a. 64
b. 12
c. 256
d. 24
e. None of the above

7. Choose the correct output of the following code fragment.

```
int[] arr={1,0,1} ;
    try {
         for ( int i= 0 ; i <= arr.length ; i++ )
            System.out.print( 1/arr[i] + " " );
         }
    catch(ArithmeticException e){
            System.out.print("0");
         }
    catch (ArrayIndexOutOfBoundsException e){
            System.out.print("out");
        }
```

a. 1 0
b. 1 0 1 out
c. 1 0 out
d. 1 0 1
e. None of the above

8. Abstract methods are used when defining

a. derived classes
b. classes that have no constructors
c. arrays
d. interfaces
e. None of the above

9. Given the following code, what is the output?

```
public class Animal
   {
     public String getColor( )
         {
            return "color";
         }
      }
```

```
public class Cat extends Animal
 {
     public String getColor( )
     {
        return "black";
      }
 }
```

```
public class Dog extends Animal
{
     public String getColor( )
     {
        return "white";
      }
 }
```

```
public class Inherit
   {
     public static void tryme(Animal a)
      {
         System.out.print(a.getColor( )+" ");
      }

     public static void main(String[ ] args)
      {
         Animal a = new Dog( );
         tryme(a);
         a=new Cat();
         tryme(a);

      }
```

a. color black
b. color white
c. color black
d. white black
e. None of the above

10. The following method purports to be a recursive method. What is the problem with it?

```
public int recurse( int n )
{
   int result = 1, rec=2;
   if (n<2)
       result= (n-1)*rec;
   return result ;
}
```

a. Method parameter must be a double
b. There is no base case
c. The recursive call does not move the parameter closer to the base case
d. Method does not call itself
e. None of the above

11. **(20 points)** Write a **recursive** method called **`recProd`** that takes two parameters:
- `array`, a non-empty array of integers
- `count`, a positive integer less than or equal to the size of `array`

The method returns the product of `count` array entries. For example, the code fragment below would display `24`.

```
…
int [] a={1, 2, 3, 4, 5};
System.out.println(recProd(a,4));
…
```

12. **(20 points)** Given the **Student class**:

```
public class Student
{

   private int  finalScore;
   public Student(int finalScore)
   {
     this.finalScore=finalScore;
   }
   public void setFinalScore(int f){
       finalScore = f;
    }
   public int getFinalScore(){
       return finalScore;
    }
}
```

Write **ONLY** the constructor for a class **Course** that has two attributes:

- **stdList**, an array of **Student objects** enrolled in the course
- **stats**, an array of integers containing the frequencies of each "A" score from the array of students. An "A" score is a score greater than 90.

The constructor takes an integer parameter named **size** representing the array's size. It initializes all the elements of the array to **student objects with finalScore value a random number between 1 and 100** and populates the **stats** array accordingly.

13. **(20 points)** Write the **definition** of a child class of class **Figure** named **Line**. The Line class is defined by 2 point (vertices) attributes defined by class **Point** below. The class should also have these members:

- a constructor that takes two Point objects as parameters to initialize the attributes.
- getters and setters for the class attributes
- a method equals() that returns true of two lines have same length and false otherwise.

HINT: Remember to define/redefine any other methods needed in the child class.

```
public abstract class Figure{
  protected String name;

  public Figure(String s){
    name=s;
  }
  public abstract double perimeter();
 //returns the total length between the vertices in figure

  public String toString(){
     return "figure";

    }
}



public class Point
{
   private int x, y;

   public Point(int x, int y)
   {
      this.x=x;
      this.y=y;
    }

    public String toString()
    {
      return "x:"+x+","+"y:"+y;
    }

    public double distance(Point other)
    {
      return Math.sqrt(Math.pow(x-other.x,2)+Math.pow(y-other.y,2));
    }
}
```

Question 11 – Solution

Question 12 – Solution

Question 13 – Solution

Quick Reference

Scanner Class

Scanner(InputStream source)
Scanner(File source)
Scanner(String source)
String next()
String nextLine()
int nextInt()
double nextDouble()
float nextFloat()

String Class

String(String str)
int length()
int compareTo(String anotherString)
char charAt(int index)
boolean equals(String anotherString)
String substring(int beginIndex, int endIndex)
String substring(int beginIndex)

Random Class

Random()
float nextFloat()
int nextInt(int num)
int nextInt()

Math Class

static double random()
static final double PI