1. What does this program print?

```
public class q1{

  public static void main(String[] args) {
    loop(10);
  }

  public static void loop(int n) {
    int i = n;
    while (i > 0) {
      System.out.println(i);
      if (i%2 == 0) {
        i = i/2;
      } else {
        i = i-1;
      }
    }
  }

}
```

10 5 4 2 1

2. A student wrote the following method, called `is_sorted`. He/She thinks the method tests if the input array is sorted (i.e. if its elements are in order from the smallest to the largest). It does not.
   a) Find/describe the mistake
   b) Is the mistake: syntax, logical or run time?
   c) Give example of an array on which the program returns the wrong answer?
   d) Fix the method so it does test if the array is sorted.
   e) Write up to 3 lines of code that does the following: creates an array of integers (with values of your choice) and then invokes/calls the method `is_sorted` with your array as input and then prints the result that `is_sorted` returns.

```
public static boolean is_sorted(int[] A) {
    boolean result=true;
    for(int i=0; i<A.length-1; i++)
    {
      if(A[i]>A[i+1]){
        result=false;
      }
      else{
        result=true;
      }
    }
    return result;
  }
```

(a)

This program only tests if the last two elements in the array are stored.

(b)

logical

(c)

2, 1, 3

(d)

```
public static boolean is_sorted(int[] A) {
    boolean result=true;
    for(int i=0; i<A.length-1; i++)
    {
      if(A[i]>A[i+1]){
        result=false;
      }
    }
     return result;
  }
```

(e)

```
int[] arr ={2, 1, 3};
System.out.println( is_sorted(arr) );
```

3. What does this program print?

```
public class q3{

  public static void main(String[] args) {
    int number = 1;
    int[] numbers = new int[1];
    numbers[0]=1;

    m(number, numbers);
    System.out.println(number);
    System.out.println(numbers[0]);
  }

  public static void m(int x, int[] y) {
    x=3;
    y[0]=3;
  }


}
```

1 3

4. For each of the following methods write one sentence that describes abstractly (in plane English) what the methods does.

```
public static int banana(int[] a) {
  int grape = 0;
  int i = 0;
  while (i < a.length) {
    grape = grape + a[i];
    i++;
  }
  return grape;
}

public static int apple(int[] a, int p) {
  int i = 0;
  int pear = 0;
  while (i < a.length) {
    if (a[i] == p){
      pear++;
    }
    i++;
  }
  return pear;
}

public static int grapefruit(int[] a, int p) {
  for (int i = 0; i<a.length; i++) {
    if (a[i] == p){
      return i;
    }
  }
  return -1;
}

public static void kiwi (int[] mango){
  int[] tmpMango=new int[mango.length];
  int i;
  for(i=0; i<mango.length; i++){
    tmpMango[i]=mango[i];
  }

  for(i=0; i<mango.length; i++){
    mango[i]=tmpMango[mango.length -1 -i];
  }
}
```

Method `banana` does the following:

> Given an array (of integers) this method sums the elements of that array and returns the result.

Method `apple` does the following:

> Given an array (of integers) and another number (again integer), this method counts how many times that given number appears in the given array, and returns that result.

Method `grapefruit` does the following:

> Given an array (of integers) and another number (again integer), this method returns the location of the first occurance of that number in the array if the number is present in the array. Otherwise it returns -1 (to indicated that the number is not present).

Method `kiwi` does the following:

> Given an array (of integers) this method reverses its elements. Eg. if the given array had elements $\{7, 2, 3\}$, at the end of this method it woudl have elements $\{3, 2, 7\}$.

5. For each of the methods in the previous question write up to 3 lines of code that demonstrate how one would use/test such a method. So create input to the method, invoke/call the method and print the result if there are any.

   Using method `banana` :

```
int[] arr = {2, 4, 3, 4};
System.out.println( banana(arr) );
```

   Using method `apple`

```
int[] arr = {2, 4, 3, 4};
System.out.println( apple(arr, 4) );
```

   Using method `grapefruit`

```
int[] arr = {2, 4, 3, 4};
System.out.println( grapefruit(arr, 4) );
```

Using method `kiwi`

```
int[] arr = {2, 4, 3, 4};
kiwi(arr);
```

6. (a) In the space below write a (body of a) method, `duplicates` that takes as input (a reference, A, to) an array of integers and returns true if array has duplicates and otherwise it returns false. Eg. if the array referenced by A has elements {1, 7, 1, 1, 8, 7, 2}, the resulting method would return true. On the other hand, if the array referenced by A is {7}, it would return false. Or if the array referenced by A is {5, 6, 1}, the method would return false.

```
public static boolean duplicates (int[] A){
```

```
for(int i=0; i<A.length-1; i++){
  for(int j=i+1; j<A.length; j++){
    if(A[i]==A[j]){
      return true;
    }
  }
}
return false;
```

```
}
```

(b) Write up to 3 lines of code that demonstrate how one would use/test method `duplicates`. In particular, create input to the method, invoke/call the method and print the result.

```
int[] list={7, 2, 1, 7};
System.out.println(duplicates(list));
```

7. (a) In the space below write a (body of a) method, `max_of_raws` that takes as input a (reference, A, to) a matrix (i.e. 2D array) of integers and returns an array, B, that contains the maximum elements of every raw.

```
public static int[] max_of_raws (int[][] A){
```

```
    int raws=A.length;
    int col=A[0].length;
    int max;

    int[] result = new int[raws];

    for(int i=0; i<raws; i++){
      max=A[i][0];
      for(int j=0; j<col; j++){
        if(A[i][j]>max){
          max=A[i][j];
        }
      }
      result[i]=max;
    }
    return result;
```

```
}
```

(b) Write up to 6 lines of code that demonstrate how one would use/test method `max_of_raws`. In particular, create input to the method, invoke/call the method and print the result.

```
    int[][] list={{7, 2, 1, 7}, {2, 0, 8, 0}, {1, 1, 1, 1}};
    int[] B = max_of_raws(list);

    for(int i=0; i<B.length; i++){
      System.out.println(B[i]);
    }
```

8. What does the following program print?

```
public class q9{

  public static void main(String[] args)
  {
    int x = 5;
    Point blank = new Point(1, 2);

    int t =riddle(x, blank);
    System.out.println(t);
    System.out.println(x);
    System.out.println(blank.x);
    System.out.println(blank.y);
  }

  public static int riddle(int x, Point p)
  {  x = x + 7;
    return x + p.x + p.y;
  }

}
```

Where class `Point` is defined as follows:

```
public class Point{
  int x;
  int y;

  public Point(int x_coor, int y_coor)
  {
    x=x_coor;
    y=y_coor;
  }
}
```

15 5 1 2

9. What does the following program print? Class `Point` is defined as in the question 9.

```
public class q10{

  public static void main(String[] args)
  {
    Point a = new Point(-1, 1);
    Point b = new Point(3, 3);
    a=b;
    a.x = 1;

    System.out.println(a.x);
    System.out.println(a.y);
    System.out.println(b.x);
    System.out.println(b.y);

  }

}
```

1 3 1 3

10. Consider the class `Point` bellow.

(a) Add to this class (in the provided space) an **instance** method, called `my_quadrant` that returns string ‘‘`top right quadrant`" if the point is in top right quadrant, it returns string ’’`top left quadrant`’’ if the point is in the top left quadrant ... and so on. To avoid ambiguity, you may assume that neither x nor y coordinate is zero.

```
public class Point{
  int x;
  int y;

  public Point(int x_coor, int y_coor)
  {
    x=x_coor;
    y=y_coor;
  }
```

```
  public String my_quadrant()
    {
     if (x>=0 && y>=0)
       return "top right quadrant";
     else if  (x>=0 && y<=0)
       return "top left quadrant";
     else if (x<=0 && y<=0)
       return "bottom left quadrant";
     else
       return "bottom right quadrant";
    }
```

```
}
```

(b) In the space below write up to 3 lines of code that use/test method `my_quadrant` that you created. In particular, create a point object, invoke/call the method `my_quadrant` for the object and print the results

```
Point a = new Point(-1, 2);
String s = a.my_quadrant();
System.out.println("Point a is in the "+s);
```

11. Definition: A point p_1 in the plane is said to *dominate* another point p_2 in the plane if the x coordinate of p_1 is at least as big as the x coordinate of p_2 and if the y coordinate of p_1 is at least as big as the y coordinate of p_2.

(a) Add to the Point class below (in the provided space) a **static** method, called `is_dominant` that given two Point objects test if the first point dominates the second point. If yes, it returns true, if not the method returns false.

```
public class Point{
  int x;
  int y;

  public Point(int x_coor, int y_coor)
  {
    x=x_coor;
    y=y_coor;
  }
```

```
 public static boolean is_dominant(Point p1, Point p2)
  {
   if(p1.x >= p2.x && p1.y >= p2.y)
     return true;
   else
     return false;
  }
```

```
}
```

(b) In the space below write up to 5 lines of code that use/test method `is_dominant` that you created. In particular, create two point objects, invoke/call the method `is_dominant` for them and print the results

```
Point a, b;
a = new Point(-1, 2);
b = new Point(3, 3);
boolean answer= Point.is_dominant(a,b);
System.out.println("Point a dominates point b is. That is "+ answer);
```