Answers in exam

**Q1:**

**Q2:**

**Q2.1 – Answer:**

```
public class Main {
public static void main(String[] args) {
   String str1 = “I love Jimi Hendrix”;
   String str2 = "“I love Jimi Hendrix";

   // comparing str1 with str2
   boolean result = str1.equals(str2);
System.out.println("The result of comparing str1 and str2 is: " + result); }}
```

**Q2.2 – Answer:**

```
public class Main {
  public static void main(String[] args) {

   //here a while loop for Metallica
   int i = 0;
   while (i < 3) {
     System.out.println("Pearl Jam Live in Madison Square Garden, NY!");
     i++;
   }

   //here a for loop for Bob Dylan
   for (i=0; i<=2; i++)
   {
     System.out.println("Bob Dylan Live in Armadillo, Glasgow!");
   }

   //here a do-while for Radiohead
```

```
   int j=1;
   do
   {
         System.out.println("Radiohead Live in O2 Arena, London!");
       j++;

   }while(j<3);
  }
}
```

**Q2.3- Answer:**

```
class Main {
  public static void main(String[] args) {

   // create an array
   int[] nums = {2, 9, 5, 10};

   // for each loop
   for (int number: nums) {
    System.out.println(number*10);
   }
  }
}
```

Q2.4 – Answer:

```
class MultidimensionalArray {
   public static void main(String[] args) {

      int[][] my2D = {
         {0, -1, 9},
         {-10, -5, 2, 9},
         {10, -2, 15},
      };

      for (int i = 0; i < my2D.length; ++i) {
         for(int j = 0; j < my2D[i].length; ++j) {
            System.out.println(my2D[i][j]);
         }
      }
   }
}
```

Q3.2

```
int i, n = 10;
        for (i = 1; i <= n; i=i*2) {
            System.out.printf("Liverpool FC!\n");
        }
```

Q3.3

```
int binarySearch(int arr[], int left, int right, int x) {
    if (right >= left) {
      int mid = left + (right - left) / 2;

      // If the element is present at the middle itself
      if (arr[mid] == x)
        return mid;

      // If element is smaller than mid, then it can only be present in left subarray
      if (arr[mid] > x)
         return binarySearch(arr, left, mid - 1, x);

      // Else the element can only be present in right subarray
      return binarySearch(arr, mid + 1, right, x);
    }
   // We reach here when element is not in the array
   return -1;
  }
```

Q3.4
Answer is quicksort.
Although both Quicksort and Mergesort have an average time complexity of O(n log n), Quicksort is the preferred algorithm, as it has an O(log(n)) space complexity. Mergesort, on the other hand, requires O(n) extra storage, which makes it quite expensive for arrays.

Q4.3

```
interface Polygon {
  void getArea(int length, int width);
}

// implement the Polygon interface
class Rectangle implements Polygon {

  // implementation of abstract method
  public void getArea(int length, int width) {
    System.out.println("The area of the rectangle is " + (length * width));
  }
}
```

```
class Main {
  public static void main(String[] args) {
    Rectangle r1 = new Rectangle();
    r1.getArea(5, 5);
  }
}
```

Q.4.4
Hybrid, output is D

Q5.4:
Solution:

```
class Mult {

    // Method with 2 integer parameters
    static int multiply(int a, int b)
    {
        // Returns product of integer numbers
        return a * b;
    }

    // Method 2
    // With same name but with 2 double parameters
    static double multiply(double a, double b)
    {
        // Returns product of double numbers
        return a * b;
    }

    // // Multiplication of 3 int numbers
    static int multiply(int a, int b, int c)
    {

        // Return product
        return a * b * c;
    }

}

// Class 2
// Main class
class Main {
    // Main driver method
    public static void main(String[] args)
    {
        // Calling method by passing
        // input as in arguments
        System.out.println("Mutiplication with 2 integers:" +
Mult.multiply(2, 4));
        System.out.println("Mutiplication with 3 integers:" +
Mult.multiply(2, 4, 2));
```

```
        System.out.println("Mutiplication with 2 doubles:" +
Mult.multiply(5.5, 3.0));
    }
}
```