1. Given a string, that contains special character together with alphabets (‘a’ to ‘z’ and ‘A’ to ‘Z’), reverse the string in a way that special characters are not affected.
Examples:
Input: str = "a,b$c"
Output: str = "c,b$a"
Note that $ and , are not moved anywhere.
Only subsequence "abc" is reversed
Input: str = "Ab,c,de!$"
Output: str = "ed,c,bA!$

Code

```
class ReverseString
{
   public static void reverse(char str[])
   {
            int r = str.length - 1, l = 0;     // Initialize left and right pointers

             // Traverse string from both ends until 'l' and 'r'
                      while (l < r)
                     {
                              if (!Character.isAlphabetic(str[l]))        // Ignore special characters
                                l++;
                                else if(!Character.isAlphabetic(str[r]))
                                 r--;
                                 else       // Both str[l] and str[r] are not spacial
                                 {
                                          char tmp = str[l];
                                          str[l] = str[r];
                                          str[r] = tmp;
                                           l++;
                                           r--;
                                  }
                     }
   }

   public static void main(String[] args)
   {
      String str = "a!!!b.c.d,e'f,ghi";
      char[] charArray = str.toCharArray();
      System.out.println("Input string: " + str);
                        reverse(charArray);
      String revStr = new String(charArray);
      System.out.println("Output string: " + revStr);
   }
}
```

2. Given an array of distinct integers and a sum value. Find count of triplets with sum smaller than given sum value. Expected Time Complexity is O(n2).
Examples:
Input : arr[] = {-2, 0, 1, 3}
sum = 2.
Output : 2
Explanation : Below are triplets with sum less than 2
(-2, 0, 1) and (-2, 0, 3)
Input : arr[] = {5, 1, 3, 4, 7}
sum = 12.
Output : 4
Explanation : Below are triplets with sum less than 4
(1, 3, 4), (1, 3, 5), (1, 3, 7) and (1, 4, 5)

Code

```
class FindTriplet {

   // returns true if there is triplet with sum equal to 'sum' present in A[]. Also, prints the triplet

   boolean find3Numbers(int A[], int arr_size, int sum)
   {
      int l, r;
      quickSort(A, 0, arr_size - 1);       /* Sort the elements */

  /* Now fix the first element one by one and find the
        other two elements */
      for (int i = 0; i < arr_size - 2; i++) {

         /* find the other two elements, start two index variables  from two corners of the array and     move
              them toward each other */

         l = i + 1;      // index of the first element in the remaining elements
         r = arr_size - 1;      // index of the last element

         while (l < r) {
           if (A[i] + A[l] + A[r] == sum) {
               System.out.print("Triplet is " + A[i] +
                        ", " + A[l] + ", " + A[r]);
               return true;
            }
            else if (A[i] + A[l] + A[r] < sum)
               l++;
    else // A[i] + A[l] + A[r] > sum
               r--;
         }
      }
   // If we reach here, then no triplet was found
      return false;
   }
   int partition(int A[], int si, int ei)
   {
      int x = A[ei];
      int i = (si - 1);
      int j;
      for (j = si; j <= ei - 1; j++) {
```

```
            if (A[j] <= x) {
                i++;
                int temp = A[i];
                A[i] = A[j];
                A[j] = temp;
    }
        }
        int temp = A[i + 1];
        A[i + 1] = A[ei];
        A[ei] = temp;
        return (i + 1);
    }
    /* Implementation of Quick Sort
    A[] --> Array to be sorted
    si  --> Starting index
    ei  --> Ending index
     */
    void quickSort(int A[], int si, int ei)
    {
        int pi;
        /* Partitioning index */
        if (si < ei) {
            pi = partition(A, si, ei);
            quickSort(A, si, pi - 1);
            quickSort(A, pi + 1, ei);
        }
    }
    public static void main(String[] args)
    {
        FindTriplet triplet = new FindTriplet();
        int A[] = { 1, 4, 45, 6, 10, 8 };
        int sum = 22;
        int arr_size = A.length;
        triplet.find3Numbers(A, arr_size, sum);
    }
}
```

3. Given an array of integers, write a function that returns true if there is a triplet (a, b, c) that satisfies a2 + b2 = c2
Example:
Input: arr[] = {3, 1, 4, 6, 5}
Output: True
There is a Pythagorean triplet (3, 4, 5).
Input: arr[] = {10, 4, 6, 12, 5}
Output: False
There is no Pythagorean triplet

Code

```
import java.io.*;
class PythagoreanTriplet {
   // Returns true if there is Pythagorean triplet in ar[0..n-1]
   static boolean isTriplet(int ar[], int n)
   {
      for (int i=0; i<n; i++)
      {
         for (int j=i+1; j<n; j++)
         {
            for (int k=j+1; k<n; k++)
            {
               // Calculate square of array elements
               int x = ar[i]*ar[i], y = ar[j]*ar[j], z = ar[k]*ar[k];

               if (x == y + z || y == x + z || z == x + y)
                  return true;
            }
         }
      }
      // If we reach here, no triplet found
      return false;
   }

  public static void main(String[] args)
   {
      int ar[] = {3, 1, 4, 6, 5};
      int ar_size = ar.length;
      if(isTriplet(ar,ar_size)==true)
        System.out.println("Yes");
      else
        System.out.println("No");
   }
}
```

6. Given an index k, return the kth row of the Pascal's triangle
For example, when k = 3, the row is [1,3,3,1]

Code

```
import java.io.*;

class PascalTriangle {

   // Function to print first n lines of Pascal's Triangle

   static void printPascal(int n)
   {

     // Iterate through every line and print entries in it

     for (int line = 0; line < n; line++)
     {
       // Every line has number of integers equal to line number

       for (int i = 0; i <= line; i++)
         System.out.print(binomialCoeff(line, i)+" ");

       System.out.println();
     }
   }

   static int binomialCoeff(int n, int k)
   {
      int res = 1;

      if (k > n - k)
         k = n - k;

      for (int i = 0; i < k; ++i)
      {
         res *= (n - i);
         res /= (i + 1);
      }
      return res;
   }

      public static void main(String args[])
   {
     int n = 7;
     printPascal(n);
   }
}
   return res;
}
```

Part B

1. Expose one or more of the solutions in part A above as a web service.

Time Complexity of the above solution of Pythagorean Triplet is $O(n^3)$.

Using Sorting Time Complexity is $O(n^2)$.

```
import java.io.*;
import java.util.*;

class PythagoreanTriplet
{
   // Returns true if there is a triplet with following property A[i]*A[i] = A[j]*A[j] + A[k]*[k]
   // Note that this function modifies given array

   static boolean isTriplet(int arr[], int n)
   {
      // Square array elements
      for (int i=0; i<n; i++)
         arr[i] = arr[i]*arr[i];

      // Sort array elements
      Arrays.sort(arr);

      // Now fix one element one by one and find the other two elements

      for (int i = n-1; i >= 2; i--)
      {
        /*To find the other two elements, start two index variables from two corners of the array
  and move them toward each other */


         int l = 0;   // index of the first element in arr[0..i-1]
         int r = i-1; // index of the last element in arr[0..i-1]
         while (l < r)
         {
            // A triplet found
            if (arr[l] + arr[r] == arr[i])
               return true;

            // Else either move 'l' or 'r'
            if (arr[l] + arr[r] < arr[i])
              l++;
            else
              r--;
         }
      }

      // If we reach here, then no triplet found
      return false;
   }

   public static void main(String[] args)
   {
      int arr[] = {3, 1, 4, 6, 5};
```

```
        int arr_size = arr.length;
        if (isTriplet(arr,arr_size)==true)
          System.out.println("Yes");
        else
          System.out.println("No");
    }
}
```

2. Create a client (web / mobile (or both)) that you shall use to feed input data to your service and response should be the output of your solution in A.

```
<html>
<head>
  <title>Pascal Triangle in PHP</title>
</head>
<body>
<?php $level = $_POST['line']; ?>
<div>
  <form action="" method="post">
  Input Line Number <input type="text" name="line" value="<?php echo $level; ?>"> <input type="submit"
value="Submit">
  </form>
</div>
<div>
<table>
<?php
/* step 4 - finish */
for ($y = 1; $y <= $level; $y ++){
  echo "<tr>";
  for ($x = 1; $x <= $y; $x ++){
    if($x == 1){
      $number[$y][$x] = 1; // start with 1

      //show tab from left
      if($level != $y){
        echo "<td colspan='".($level-$y)."'></td>";
      }

      echo "<td>".$number[$y][$x]."</td>";
      echo "<td></td>"; //show tab
    }elseif($x == $y){
      $number[$y][$x] = 1; // end with 1
      echo "<td>".$number[$y][$x]."</td>";
    }else{
      $number[$y][$x] = $number[$y-1][$x-1] + $number[$y-1][$x];
      echo "<td>".$number[$y][$x]."</td>";
      echo "<td></td>"; //show tab
    }
  }
  echo "</tr>";
}
?>
</table>
```

```
</div>
</body>
</html>
```