1. pascal triangle:

```python
def generate_pascals_triangle(rows):
    triangle = []

    for i in range(rows):
        row = [1] * (i + 1)

        # Update the values inside the triangle
        for j in range(1, i):
            row[j] = triangle[i-1][j-1] + triangle[i-1][j]

        # Append the current row to the triangle
        triangle.append(row)

    return triangle

# Default number of rows for Pascal's Triangle
rows = 5  # You can change this to any number of rows you'd like to generate

triangle = generate_pascals_triangle(rows)

# Print Pascal's Triangle
for row in triangle:
    print(row)
```

Expected Output:

```
[1]
[1, 1]
[1, 2, 1]
[1, 3, 3, 1]
```

```
[1, 4, 6, 4, 1]
```

2. Sum of Natural Numbers using Recursion

```python
def sum_of_natural_numbers(n):
    # Base case: if n is 0, return 0
    if n == 0:
        return 0
    else:
        # Recursive case: sum = n + sum of (n-1)
        return n + sum_of_natural_numbers(n-1)

# Default value for n
n = 10  # You can change this to any number you'd like

result = sum_of_natural_numbers(n)
print(f"The sum of natural numbers up to {n} is {result}")
```

Expected Output:

```
The sum of natural numbers up to 10 is 55
```

3. Find the Frequency of a Substring in a String

```java
public class SubstringFrequency {

    public static int countSubstringFrequency(String str, String substring) {
        int count = 0;
        int index = 0;

        // Loop through the string to find all occurrences of the substring
        while ((index = str.indexOf(substring, index)) != -1) {
            count++;
            index += substring.length(); // Move the index forward by the length of the substring
        }

        return count;
    }

    public static void main(String[] args) {
        String str = "ababcabcabc";
        String substring = "abc";

        int frequency = countSubstringFrequency(str, substring);

        System.out.println("The frequency of the substring \"" + substring + "\" is: " + frequency);
    }
}
```

Expected Output:

```
The frequency of the substring "abc" is: 3
```

4. Delete All Repeated Words in String

```
import java.util.*;

public class RemoveRepeatedWords {

    public static String removeDuplicates(String input) {
        // Split the input string into words
        String[] words = input.split("\\s+");

        // Set to keep track of unique words
        Set<String> seen = new HashSet<>();

        // StringBuilder to store the result
        StringBuilder result = new StringBuilder();

        for (String word : words) {
            // If the word is not already in the set, add it to the result
            if (!seen.contains(word)) {
                seen.add(word);
                result.append(word).append(" ");
            }
        }

        // Remove the trailing space and return the result
        return result.toString().trim();
    }

    public static void main(String[] args) {
        String input = "This is a test test string with with repeated repeated words";

        String output = removeDuplicates(input);

        System.out.println("Original String: " + input);
        System.out.println("String after removing repeated words: " + output);
    }
}
```

Expected Output:

```
Original String: This is a test test string with with repeated repeated words
String after removing repeated words: This is a test string with repeated words
```

5. Find Missing Numbers in Array

```c
#include <stdio.h>

void main() {
    int n = 6; // Default size of the array (you can change this value)
    int array[] = {1, 2, 3, 4, 6}; // Default array (you can modify this array)

    int i, b, c;

    // Calculate the XOR of all elements in the array
    b = array[0];
    for (i = 1; i < n - 1; i++) {
        b = b ^ array[i];
    }

    // Calculate the XOR of all numbers from 1 to n
    for (i = 2, c = 1; i <= n; i++) {
        c = c ^ i;
    }

    // The missing number will be the XOR of the two results
    c = c ^ b;

    printf("Missing element is: %d\n", c);
}
```

Expected Output:

```
Missing element is: 5
```

8. Compare Two Strings

```
#include <stdio.h>
#include <string.h>

int main() {
   int count1 = 0, count2 = 0, i;
   char string1[30] = "Hello"; // Default first string
   char string2[30] = "World"; // Default second string

   printf("First string: %s\n", string1);
   printf("Second string: %s\n", string2);

   // Find the lengths of both strings
   while (string1[count1] != '\0')
      count1++;
   while (string2[count2] != '\0')
      count2++;

   // Compare the strings lexicographically
   i = 0;
   while (string1[i] == string2[i] && string1[i] != '\0') {
      i++;
   }

   if (string1[i] > string2[i])
      printf("First string is greater than second string\n");
   else if (string1[i] < string2[i])
      printf("Second string is greater than first string\n");
   else
      printf("Both strings are EQUAL\n");
```

```
    return 0;
}
```

Expected Output:

```
First string: Hello
Second string: World
Second string is greater than first string
```