```
➢ def index_sort_count(A):
    count = 0
    for K in range(1, len(A) + 1):
        if A[K - 1] != K:
            count += 1
    return count
# Example usage:
array = [3, 1, 2]
result = index_sort_count(array)
print(result)
```

```
>def max_reachable_squares(H, W):
    return min(H, W)
# Example usage:
rows = 5
columns = 6
result = max_reachable_squares(rows, columns)
print(result)
```

```
>public class MaxBishopSquares {
    public static int maxBishopSquares(int H, int W) {
return Math.min(H, W);  }
public static void main(String[] args) {
int H = 5;
        int W = 7;
        int result = maxBishopSquares(H, W);
        System.out.println(result); }}
```

Birthday party :

```
>import java.util.Scanner;
public class BirthdayParty {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the number of pieces of the first cake (N): ");
        int N = scanner.nextInt();
        System.out.print("Enter the number of pieces of the second cake (M): ");
        int M = scanner.nextInt();
        int result = maxPeopleWithEqualPieces(N, M);
        System.out.println("Maximum number of people at the party: " + result);
    }
    private static int maxPeopleWithEqualPieces(int N, int M) {
        int gcd = findGCD(N, M);
            return gcd;
    }
    private static int findGCD(int a, int b) {
        while (b != 0) {
            int temp = b;
            b = a % b;
            a = temp;
        }
        return a;  } }
```

```
>def min_steps_to_magic_string(S):
    # Count occurrences of each character
    char_count = {}
    for char in S:
        if char in char_count:
            char_count[char] += 1
        else:
            char_count[char] = 1
    # Find the maximum occurrence of a character
    max_occurrence = max(char_count.values()) if char_count else 0
    # Calculate the minimum steps required
    return len(S) - max_occurrence if max_occurrence > 0 else 0
# Example usage:
input_string = "al.com"
result = min_steps_to_magic_string(input_string)
print(result)
```

```
>def unique_marbles(N, A, B):
    unique_set = set()
    def explore(marbles):
        if marbles > 0 and marbles not in unique_set:
            unique_set.add(marbles)
            explore(marbles - A)
            explore(marbles - B)

    explore(N)
    return len(unique_set)

# Example usage:
initial_marbles = 10
operation_A = 3
operation_B = 5

result = unique_marbles(initial_marble
operation_A, operation_B)
print(result)_____
```

```
➢ def special_series(N):
if N == 1 or N == 2:
        return 1
    prev_1 = 1
    prev_2 = 1
    current = 0
    for i in range(3, N+1):
        current = (prev_1 * (i-1) + prev_2 * (i-2)) % 47
        prev_2 = prev_1
        prev_1 = current
    return current
# Example usage:
N_value = 10
result = special_series(N_value)
print(result)
```

```
>def maximize_grade(N, P, K):
    # Convert the string to a list for easy swapping
    grades = list(N)
    # Find the lexicographically smallest character
    smallest_char = min(grades)
    # Swap the smallest character with the character at position P
    grades[P - 1] = smallest_char
    # Perform additional swaps if allowed
    swaps_left = K - 1
    idx = 0
    while swaps_left > 0 and idx < len(grades):
```

```
>import java.util.Scanner;
public class MinASCIIDistance {
    public static int minASCIIDistance(String s, String a) {
        int distance = 0;
        for (int i = 0; i < s.length(); i++) {
            char charS = s.charAt(i);
            char charA = a.charAt(i);

            int asciiS = (int) charS;
            int asciiA = (int) charA;

            int diff = Math.abs(asciiS - asciiA);
            distance += diff;
```

```
>def ant_original_position(N, A):
    net_displacement = 0
    for move in A:
        if move == 1:
            net_displacement += 1
        else:
            net_displacement -= 1
    # If the net displacement is zero, the ant returns to the original position
    return abs(net_displacement // N)
# Example usage:
moves = 7
ant_moves = [1, 1, -1, -1, 1, -1, -1]
result = ant_original_position(moves, ant_moves)
```

```
    if grades[idx] != smallest_char:
        grades[idx] = smallest_char
        swaps_left -= 1
    idx += 1
  # Convert the list back to a string and
return
  return ''.join(grades)
# Example usage:
grades = "Jeepk"
position = 3
max_swaps = 2
result = maximize_grade(grades, position,
max_swaps)
print(result)
```

```
    }
    return distance;
  }
  public static void main(String[] args)
{
Scanner n=new Scanner(System.in);
    String stringS =n.next();
    String stringA = n.next();
    int result =
minASCIIDistance(stringS, stringA);
    System.out.println("Minimum total
ASCII distance: " + result);
}
}
```

```
print(result)
```

```
>public class FellisFunction {
public static void main(String[] args) {
    int N = 2;
    int result = fellisFunction(N);
    System.out.println(result);    }
public static int fellisFunction(int N) {
    if (N == 0) {
        return 1;
    } else if (N == 1) {
        return 1;
    } else {
        return (fellisFunction(N - 1) +
* fellisFunction(N - 2) + (N / 4)) %
1000000007; } } }
```

- def string_swap(S, operations):

```
  for op in operations:
    if op == 1:
      # Swap the first and last characters
      S = S[-1] + S[1:-1] + S[0]
    elif op == 2:
      # Swap the first and last N/2
characters
      mid = len(S) // 2
      S = S[mid:] + S[:mid]
  return S
# Example usage:
initial_string = "abcdefgh"
sequence_of_operations = [1, 2, 1]
result = string_swap(initial_string,
sequence_of_operations)
print(result)
```

- def special_series(N):

```
  if N == 1 or N == 2:
    return 1
  prev_1 = 1
  prev_2 = 1
  current = 0
  for i in range(3, N+1):
    current = (prev_1 * (i-1) + prev_2 *
(i-2)) % 47
    prev_2 = prev_1
    prev_1 = current
  return current
# Example usage:
N_value = 10
result = special_series(N_value)
print(result)
```

- def string_swap(S, A):

```
  for op in A:
    if op == 1:
      # Swap the first and last
characters
      S = S[-1] + S[1:-1] + S[0]
    elif op == 2:
      # Swap the first N/2 and last N
characters
      mid = len(S) // 2
      S = S[mid:] + S[:mid]
  return S
# Example usage:
initial_string = "abcdefgh"
operations_sequence = [1, 2, 1]
result = string_swap(initial_string,
operations_sequence)
print(result)
```

- public class AntOnRail {

```
  public static void main(String[] args) {
      int[] moves = {1, -1, 1, -1, 1};
    int numberOfMoves = moves.length;
    int result =
countBackToStartPosition(numberOfMoves,
moves);
    System.out.println("Number of times
the ant reaches back to its original position:
" + result);
  }
  public static int
countBackToStartPosition(int N, int[] A) {
    int currentPosition = 0; // Initialize the
current position
    int count = 0; // Initialize the count of
reaching back to the original position
    for (int i = 0; i < N; i++) {
      currentPosition += A[i];
      if (currentPosition == 0) {
        count++;
      }
    }
```

- public class TossAndScore {

```
  public static int
calculateFinalScore(String s) {
    int score = 0;
    int consecutiveHeads = 0;
    for (char result : s.toCharArray()) {
      if (result == 'H') {
        consecutiveHeads++;
        score += 2;
      } else {
        consecutiveHeads = 0;
        score -= 1;
      }
      if (consecutiveHeads == 3) {
        break; // Game ends when 3
consecutive heads are obtained
      }
    }
    return score;
  }
  public static void main(String[] args)
{
    String tossResults = "HTHHTT"; //
```

- import java.util.Scanner;

```
public class SpecialFibonacci {
  public static void main(String[] args
{
    Scanner scanner = new
Scanner(System.in);
System.out.print("Enter the value of N
");
    int N = scanner.nextInt();
    int result =
calculateSpecialFibonacci(N);
    System.out.println("Result: " +
result);
    scanner.close();
  }
  private static int
calculateSpecialFibonacci(int N) {
    if (N == 0 || N == 1) {
      return N;
    }

    int[] fib = new int[N + 1];
    fib[0] = 1;
```

```
        return count;
    }
}
```

```
Replace with your actual toss results
        int finalScore =
calculateFinalScore(tossResults);
        System.out.println("Final Score: " +
finalScore);
    }
}
```

```
        fib[1] = 1;

        for (int i = 2; i <= N; i++) {
            fib[i] = (fib[i - 1] * fib[i - 1] +
fib[i - 2]*fib[i-2]);
        }

        return fib[N];
    }
}
```

```
import java.util.Arrays;
public class MaxProductPair {
    public static int[]
maxProductPairWithSum18(int[] arr) {
        int maxProduct =
Integer.MIN_VALUE;
        int[] resultPair = new int[2];
        for (int i = 0; i < arr.length; i++) {
            for (int j = i + 1; j < arr.length; j++) {
                if (arr[i] + arr[j] == 18 && arr[i] >
arr[j]) {
                    int product = arr[i] * arr[j];
                    if (product > maxProduct) {
                        maxProduct = product;
                        resultPair[0] = arr[i];
                        resultPair[1] = arr[j];
                    } } }   }
        return resultPair; }
    public static void main(String[] args) {
        int[] integerArray = {4, 8, 2, 10, 6};
        int[] result =
maxProductPairWithSum18(integerArray);
        System.out.println("Pair with
maximum product and sum 18: " +
Arrays.toString(result));
    }
}
```

```
public class FellisFunction {

    public static void main(String[] args)
{
        int N = 2;
        int result = fellisFunction(N);
        System.out.println(result);
    }

    public static int fellisFunction(int N) {
        if (N == 0) {
            return 1;
        } else if (N == 1) {
            return 1;
        } else {
            return (fellisFunction(N - 1) + 7
* fellisFunction(N - 2) + (N / 4)) %
1000000007;
        }
    }
}
```

```
import java.util.Arrays;
public class PSBalance {
    public static void main(String[] args
{
        int[] a={1,2,3,4,5};
        int[] res=Balance(a);
System.out.println(Arrays.toString(res
    }
    public static int[] Balance(int[] a){
        int n=a.length;
        int[] r=new int[n];
        int sum=0;
        for(int i=0;i<n;i++){
            sum+=a[i];
            r[i]=Math.abs(sum-
sumR(i+1,a));
        }
        return r;  }
    public static int sumR(int j,int[] a){
        int n=a.length;
        int val=0;
        for(int i=j;i<n;i++){
            val+=a[i];
        }
        if(j==n){
            return 0; }
        return val;  } }
```

Generated numbers :

```
import java.util.HashSet;
import java.util.Scanner;
import java.util.Set;
public class UniqueMarbles {
    public static void main(String[] args) {
        Scanner scanner = new
Scanner(System.in);
        System.out.print("Enter the initial
number of marbles (N): ");
        int N = scanner.nextInt();
        System.out.print("Enter the value of A:
");
        int A = scanner.nextInt();
        System.out.print("Enter the value of B:
");
        int B = scanner.nextInt();
        int result = countUniqueMarbles(N, A,
B);
```

Pyramid sum :

```
import java.util.Scanner;
public class PyramidSum {
    public static void main(String[] args)
{
        Scanner scanner = new
Scanner(System.in);
        System.out.print("Enter the height
of the pyramid (N): ");
        int N = scanner.nextInt();
        int result = pyramidSum(N);
        System.out.println("Sum of the
pyramid: " + result);
    }
    private static int pyramidSum(int N) {
        int sum = 0;
        for (int i = 1; i <= N; i++) {
            sum += rowSum(i);
        }
```

- ```
  import java.util.Scanner;
  public class MarbleJarOperations
  {
      public static int
  performOperations(int N, int A, int B)
      {
          while (N > 0)
          {
              if (N - A >= 0)
              {
                  N -= A;
              }
              else if (N - B >= 0)
              {
                  N -= B;
              }
              else
              {
                  break;
  ```
  ```

```
        System.out.println("Total number of
unique marbles that can be left behind: " +
result);
    }
    private static int countUniqueMarbles(int
N, int A, int B) {
        Set<Integer> uniqueMarbles = new
HashSet<>();
        uniqueMarbles.add(N);
        for (int i = 0; i <= N; i += A) {
            for (int j = 0; j <= N; j += B) {
                uniqueMarbles.add(N - i - j);
            }
        }
        return uniqueMarbles.size();
    }
}
```

```
        return sum;
    }
    private static int rowSum(int row) {
        int sum = 0;
        for (int i = 1; i <= row; i++) {
            sum += i;
        }
        return sum;
    }
}
```

```
            } }
        return N;  }
 public static void main(String[] args)
    {
        Scanner scanner = new
Scanner(System.in);
        System.out.print("Enter the initia
number of marbles (N): ");
        int initialMarbles =
scanner.nextInt();
        System.out.print("Enter the numb
of marbles to take out (A): ");
        int takeOutA = scanner.nextInt();
        System.out.print("Enter the numb
of marbles to take out (B): ");
        int takeOutB = scanner.nextInt();
        int remainingMarbles =
performOperations(initialMarbles,
takeOutA, takeOutB);
        System.out.println("Remaining
marbles in the jar: " +
remainingMarbles);
        scanner.close(); } }
```

```
def find_original_number(z1, z2, z3):
"""
Finds the original integer X given its
transformed values z1, z2, and z3.

Args:
z1: Integer resulting from the first bit flip.
z2: Integer resulting from the second bit flip.
z3: Integer resulting from the third bit flip.

Returns:
The original integer X.
# Find the differences between the
transformed values.
diff12 = z2 ^ z1
diff23 = z3 ^ z2

# Identify the bit positions of the
differences.
bit12 = diff12.bit_length() - 1
bit23 = diff23.bit_length() - 1

# Determine the original bit values based on
the differences.
bit1 = (z1 >> bit12) & 1
bit2 = (z2 >> bit23) & 1

# Calculate the original value of X.
x = z1 ^ (1 << bit12) ^ (1 << bit23)

# Check if x is valid and satisfies the
conditions.
if x.bit_length() > 32 or (2 * (x ^ (1 <<
```

```
public class ParallelUniverseChessBoard
{
    public static int
maxSquaresBishopCanReach(int rows,
int columns) {
        int[][] chessboard = new
int[rows][columns];.
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                chessboard[i][j] = 0;
            } }
        chessboard[0][0] = 1;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                if (chessboard[i][j] == 1) {
                    int k = i - 1;
                    int l = j - 1;
                    while (k >= 0 && l >= 0) {
                        chessboard[k][l] = 1;
                        k--;
                        l--;
                    }
                    k = i - 1;
                    l = j + 1;
                    while (k >= 0 && l <
columns) {
                        chessboard[k][l] = 1;
                        k--;
                        l++;
                    }
                    k = i + 1;
                    l = j - 1;
                    while (k < rows && l >= 0)
```

```
public class StringSwap {

    public static String
performOperations(String s, int[]
operations) {
        for (int op : operations) {
            if (op == 1) {
                s = swapFirstAndLast(s);
            } else if (op == 2) {
                s = swapFirstAndLastHalf(s
            }
        }
        return s;
    }

    private static String
swapFirstAndLast(String s) {
        char[] chars = s.toCharArray();
        char temp = chars[0];
        chars[0] = chars[chars.length - 1]
        chars[chars.length - 1] = temp;
        return new String(chars);
    }

    private static String
swapFirstAndLastHalf(String s) {
        int length = s.length();
        int half = length / 2;
        char[] chars = s.toCharArray();

        for (int i = 0; i < half; i++) {
            char temp = chars[i];
            chars[i] = chars[i + half];
```

```
bit12)) + 5) != z2 or (2 * (x ^ (1 << bit23)) + 5) != z3:
raise ValueError("Invalid value for X")

return x

# Example usage
z1 = 42
z2 = 38
z3 = 36
original_number = find_original_number(z1, z2, z3)

print(f"Original number: {original_number}")
```

```
{
                    chessboard[k][l] = 1;
                    k++;
                    l--;
                }
                k = i + 1;
                l = j + 1;
                while (k < rows && l < columns) {
                    chessboard[k][l] = 1;
                    k++;
                    l++;
                } }    } }
        int count = 0;
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < columns; j++) {
                if (chessboard[i][j] == 1) {
                    count++;
                }} }
        return count; }
    public static void main(String[] args)
{
        int rows = Integer.parseInt(args[0]);
        int columns = Integer.parseInt(args[1]);
        int maxSquares = maxSquaresBishopCanReach(rows, columns);
        System.out.println("The maximum number of squares the bishop can reach is: " + maxSquares);
    }
}
```

```
            chars[i + half] = temp;
        }

        return new String(chars);
    }

    public static void main(String[] args)
{
        String inputString = "String"; // Replace with your input string
        int[] operations = {1, 2, 1}; // Replace with your array of operations

        String result = performOperations(inputString, operations);
        System.out.println("Final String: " + result);
    }
}
```

```
public class AntOnRailing
{
    public static int finalPosition(int[] moves)
    {
        int position = 0;
        for (int move : moves)
        {
            position += move;
        }
        return position;
    }
    public static void main(String[] args)
    {
        int[] antMoves = {1, -1, 1, 1, -1};

        int finalPosition = finalPosition(antMoves);
        System.out.println("Final position of the ant: " + finalPosition);
    }
}
```

```
>import java.util.Scanner;
public class MagicStringTransformation
{
    public static String makeMagicString(String S)
    {
        if (S.isEmpty())
        {
            return S;
        }
        char firstChar = S.charAt(0);
        String magicString = S.replaceAll("[a-zA-Z]", String.valueOf(firstChar));
        return magicString;
    }
    public static void main(String[] args)
    {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the string S: ");
        String inputString =
```

```
>def remove_unique_characters(S):
    frequency = {}
    non_unique_chars = set()
    # Count the frequency of each character
    for char in S:
        frequency[char] = frequency.get(char, 0) + 1
    # Identify non-unique characters
    for char, count in frequency.items():
        if count > 1:
            non_unique_chars.add(char)
    result = []
    # Traverse the string and keep non-unique characters
    for char in S:
        if char in non_unique_chars:
            result.append(char)
    return ''.join(result)
# Example usage:
S = input("Enter the string: ")
result = remove_unique_characters(S)
print(f"Modified string: {result}")
```

	scanner.nextLine(); String magicString = makeMagicString(inputString); System.out.println("Magic string: " + magicString); scanner.close(); } }	

import java.util.Arrays;
 	
public class DivideArray
{
 	
public static void divideArray(int[] arr)
{
int N = arr.length;
 	
for (int i = 1; i <= N; i++)
{
int[] firstPart = Arrays.copyOfRange(arr, 0, i);
int[] secondPart = Arrays.copyOfRange(arr, i, N);
 	
System.out.println("First part (i = " + i + "): " + Arrays.toString(firstPart));
System.out.println("Second part (N - i = " + (N - i) + "): " + Arrays.toString(secondPart));
System.out.println();
}
}
 	
public static void main(String[] args)
{
int[] A = {1, 2, 3, 4, 5};
System.out.println("Dividing the array A into two parts:");
divideArray(A);
}
}
	>def calculate_qualifying_score(S1, S2, P):
	qualifying_score = 35
	# Step 1: Subtract marks obtained in semester 1 from semester 2
	score_diff = [S2[i] - S1[i] for i in range(len(S1))]
	# Step 2: Add marks of up to P subjects with high scores
	qualifying_score += sum(sorted(score_diff, reverse=True)[:P])
	# Determine qualification status
	result = "qualified" if qualifying_score >= 35 else "disqualified"
	return result
	# Example usage:
	subjects_count = int(input("Enter the number of subjects: "))
	S1 = [float(input(f"Enter marks for subject {i + 1} in semester 1: ")) for i in range(subjects_count)]
	S2 = [float(input(f"Enter marks for subject {i + 1} in semester 2: ")) for i in range(subjects_count)]
	P = int(input("Enter the number of subjects to consider for high scores: "))
	
	qualification_result = calculate_qualifying_score(S1, S2, P)
	print(f"The student is {qualification_result} to appear in the competition.")
		>def gcd(a, b):
		while b:
		a, b = b, a % b
		return a
		
		def lcm(a, b):
		return a * b // gcd(a, b)
		
		def find_meeting_time(N, M):
		return lcm(N, M)
		
		# Example usage:
		N = 5 # Time taken by Robotop to complete one lap
		M = 7 # Time taken by Robocop to complete one lap
		
		meeting_time = find_meeting_time(N, M)
		print("The least time at which they me again is:", meeting_time, "seconds")

➢ def solve(input1, input2, input3):
N = input1
A = input2
B = input3
uniq = set()
uniq.add(N)
for
in range(N):
cur = list(uniq)
for value in cur:
if value A > Θ: uniq.add(value - A)
if value B> 0: uniq.add(value - B)
return len(uniq)
	String swap :
	import java.util.Scanner;
	public class StringSwap {
	public static void main(String[] args)
	{
	Scanner scanner = new Scanner(System.in);
	System.out.print("Enter the initial string: ");
	String inputString = scanner.nextLine();
	System.out.print("Enter the length of array M: ");
	int m = scanner.nextInt();
	int[] operations = new int[m];
	System.out.print("Enter the array A of length M: ");
		>public class keyboardPress {
		Run | Debug
		public static void main(String[] args) {
		String s = "60004";
		int result = minKeyPress(s);
		System.out.println(result);
		public static int minKeyPress(String s)
		int currentNo = 1;
		int keyPress = -1;
		for (charc: s.toCharArray()) {
		int digit = c '0';
		} currentNo = currentNo + 10 + digit;
		} } }

```
        for (int i = 0; i < m; i++) {
            operations[i] = scanner.nextInt();
        }
        String result =
performOperations(inputString,
operations);
        System.out.println("Final string
after performing all operations: " +
result);
    }
   private static String
performOperations(String str, int[]
operations) {
        int n = str.length();
        for (int operation : operations) {
            if (operation == 1) {
                str = swapFirstAndLast(str);
            } else if (operation == 2) {
str = swapFirstAndLastHalf(str);
            }
        }
        return str;
    }
    private static String
swapFirstAndLast(String str) {
        char[] charArray =
str.toCharArray();
        char temp = charArray[0];
        charArray[0] =
charArray[charArray.length - 1];
        charArray[charArray.length - 1] =
temp;
        return new String(charArray);
    }
    private static String
swapFirstAndLastHalf(String str) {
        int n = str.length();
        int half = n / 2;
        char[] charArray =
str.toCharArray();
        for (int i = 0; i < half; i++) {
            char temp = charArray[i];
            charArray[i] = charArray[i +
half];
            charArray[i + half] = temp;
        }
        return new String(charArray);
    }
}
```