Arrays

1.How do you find the missing number in a given integer array of 1 to 100?

i.Sum of the series: Formula: n (n+1)/2( but only work for one missing number)

**public** **class** MissingNumber {

**public** **static** **void** main(String[] args) {

**int**[] iArray = **new** **int**[] { 1, 2, 3, 5 };

**int** missingNum = *getMissingNumber*(iArray, 5);

System.***out***.println(missingNum);

}

**private** **static** **int** getMissingNumber(**int**[] numbers, **int** totalCount) {

**int** expectedSum = totalCount \* ((totalCount + 1) / 2);

**int** actualSum = 0;

**for** (**int** i : numbers) {

actualSum += i;

}

**return** expectedSum - actualSum;

}

}

**2.How do you find the duplicate number on a given integer array and remove?**

**0 to n-1: let n be 7 and array be {1, 2, 3, 1, 3, 6, 6}, the answer should be 1, 3 and 6.**

**public class FindAndReoveDuplicatesInArray {**

**void dupAll(int[] array){**

**for (int i = 0; i < array.length; i++) {**

**for (int j = i + 1 ; j < array.length; j++) {**

**if (array[i]==array[j]) {**

**// got the duplicate element } } }**

**}**

**}**

**}**

**}**

**// find duplicates elements from 0 to n-1**

**static void printRepeating(int arr[], int size) {**

**int i;**

**for (i = 0; i < size; i++) {**

**if (arr[Math.abs(arr[i])] >= 0)**

**arr[Math.abs(arr[i])] = -arr[Math.abs(arr[i])];**

**else**

**System.out.print(Math.abs(arr[i]) + " ");**

**}**

**System.out.println("");**

**}**

**static void printRepeatingLatest(int arr[], int n) {**

**for (int i = 0; i < n; i++) {**

**int index = arr[i] % n;**

**arr[index] += n;**

**}**

**for (int i = 0; i < n; i++) {**

**if ((arr[i] / n) > 1)**

**System.out.print(i + " ");**

**}**

**System.out.println("");**

**}**

**// remove duplicates for sorted array**

**static int removeDuplicatesInSortedArray(int arr[], int n) {**

**if (n == 0 || n == 1)**

**return n;**

**int j = 0;**

**for (int i = 0; i < n - 1; i++)**

**if (arr[i] != arr[i + 1])**

**arr[j++] = arr[i];**

**arr[j++] = arr[n - 1];**

**return j;**

**}**

**public static int[] removeDuplicates(int[] arr) {**

**int end = arr.length;**

**for (int i = 0; i < end; i++) {**

**for (int j = i + 1; j < end; j++) {**

**if (arr[i] == arr[j]) {**

**int shiftLeft = j;**

**for (int k = j + 1; k < end; k++, shiftLeft++) {**

**arr[shiftLeft] = arr[k];**

**}**

**//Or arr[j]=arr[end-1];**

**end--;**

**j--;**

**}**

**}**

**}**

**int[] whitelist = new int[end];**

**for (int i = 0; i < end; i++) {**

**whitelist[i] = arr[i];**

**}**

**return whitelist;**

**}**

**public static void main(String[] args) {**

**int arr[] = { 1, 6, 3, 1, 3, 6, 6 };**

**int arr\_size = arr.length;**

**// printRepeating(arr, arr\_size); //output 1 3 6 6**

**printRepeatingLatest(arr, arr\_size); // output 1 3 6**

**printArray(arr);**

**int sortedArray[] = { 1, 2, 2, 3, 4, 4, 4, 5, 5 };**

**int n = sortedArray.length;**

**n = removeDuplicatesInSortedArray(sortedArray, n);// output 1 2 3 4 5**

**for (int i = 0; i < n; i++)**

**System.out.print(sortedArray[i] + " ");**

**}**

**private static void printArray(int[] arr) {**

**int n = arr.length;**

**for (int i = 0; i < n; i++) {**

**System.out.print((arr[i] % n) + " ");**

**}**

**System.out.println("");**

**}**

**}**

**3. How do you find the largest(second) and smallest(second) number in an unsorted integer array?**

**import java.util.Arrays;**

**public class LargestAndSmallestInArray {**

**public static void largestAndSmallest(int[] numbers) {**

**int largest = Integer.MIN\_VALUE;**

**int smallest = Integer.MAX\_VALUE;**

**for (int number : numbers) {**

**if (number > largest) {**

**largest = number;**

**} if (number < smallest) {**

**smallest = number;**

**}**

**}**

**System.out.println("Given integer array : " + Arrays.toString(numbers));**

**System.out.println("Largest number in array is : " + largest);**

**System.out.println("Smallest number in array is : " + smallest);**

**}**

**public static void secondLargest(int[] numbers) {**

**int largest = Integer.MIN\_VALUE;**

**int scondLarge=Integer.MIN\_VALUE;**

**for (int number : numbers) {**

**if (number > largest) {**

**scondLarge=largest;**

**largest = number;**

**} else if (scondLarge < number) {**

**scondLarge = number;**

**}**

**}**

**System.out.println("Largest number in array is : " + largest);**

**System.out.println("Second Largest number in array is : " + scondLarge);**

**}**

**public static void secondMin(int[] numbers) {**

**int min = Integer.MAX\_VALUE;**

**int secondMin= Integer.MAX\_VALUE;**

**for (int number : numbers) {**

**if (number < min) {**

**secondMin=min;**

**min = number;**

**} else if (secondMin > number) {**

**secondMin = number;**

**}**

**}**

**System.out.println("Min number in array is : " + min);**

**System.out.println("Second Min number in array is : " + secondMin);**

**}**

**public static void main(String[] args) {**

**largestAndSmallest(new int[] { -20, 34, 21, -87, 92, Integer.MAX\_VALUE });**

**largestAndSmallest(new int[] { 10, Integer.MIN\_VALUE, -2 });**

**largestAndSmallest(new int[] { Integer.MAX\_VALUE, 40, Integer.MAX\_VALUE });**

**largestAndSmallest(new int[] { 1, -1, 0 });**

**secondLargest(new int[] { -20, 34, 21, -87, 92, Integer.MAX\_VALUE });**

**secondMin(new int[] { -20, 34, 21, -87, 92, Integer.MAX\_VALUE });**

**}**

**}**

**4.How do you find all pairs of an integer array whose sum is equal to a given number?**

**public class SumEQGivenNumber {**

**public static void printPairs(int[] array, int sum) {**

**for (int i = 0; i < array.length; i++) {**

**int first = array[i];**

**for (int j = i + 1; j < array.length; j++) {**

**int second = array[j];**

**if ((first + second) == sum) {**

**System.*out*.printf("(%d, %d) %n", first, second);**

**}**

**}**

**}**

**}**

**public static void printPairsUsingTwoPointers(int[] numbers, int k) {**

**if (numbers.length < 2) {**

**return;**

**}**

**Arrays.*sort*(numbers);**

**int left = 0;**

**int right = numbers.length - 1;**

**while (left < right) {**

**int sum = numbers[left] + numbers[right];**

**if (sum == k) {**

**System.*out*.printf("(%d, %d) %n", numbers[left], numbers[right]);**

**left = left + 1;**

**right = right - 1;**

**} else if (sum < k) {**

**left = left + 1;**

**} else if (sum > k) {**

**right = right - 1;**

**}**

**}**

**}**

**public static void printPairsUsingSet(int[] numbers, int n) {**

**if (numbers.length < 2) {**

**return;**

**}**

**Set set = new HashSet(numbers.length);**

**for (int value : numbers) {**

**int target = n - value;**

**if (!set.contains(target)) {**

**set.add(value);**

**} else {**

**System.*out*.printf("(%d, %d) %n", value, target);**

**}**

**}**

**}**

**public static void main(String[] args) {**

***printPairsUsingTwoPointers*(new int[] { 12, 14, 17, 15, 19, 20, -11 }, 9);**

**}**

**}**

**5. Find the two repeating elements in a given array**

**class RepeatElement**

**{**

**void printRepeating(int arr[], int size)**

**{**

**int count[] = new int[size];**

**int i;**

**System.out.println("Repeated elements are : ");**

**for (i = 0; i < size; i++)**

**{**

**if (count[arr[i]] == 1)**

**System.out.print(arr[i] + " ");**

**else**

**count[arr[i]]++;**

**}**

**}**

**void printRepeatingElementsCounts(int arr[], int size) {**

**int count[] = new int[size];**

**int i;**

**System.*out*.println("Repeated elements are : ");**

**for (i = 0; i < size; i++) {**

**count[arr[i]]++;**

**}**

**for (i = 0; i < size; i++) {**

**if (count[arr[i]] >= 1) {**

**System.*out*.println(arr[i] + " : " + count[arr[i]]);**

**count[arr[i]] = 0;**

**}**

**}**

**}**

**public static void main(String[] args)**

**{**

**RepeatElement repeat = new RepeatElement();**

**int arr[] = {4, 2, 4, 5, 2, 3, 1};**

**int arr\_size = arr.length;**

**repeat.printRepeating(arr, arr\_size);**

**}**

**}**

**6. Subarray with given sum**

**int subArraySum(int arr[], int n, int sum)**

**{**

**int curr\_sum = arr[0], start = 0, i;**

**// Pick a starting point**

**for (i = 1; i <= n; i++)**

**{**

**// If curr\_sum exceeds the sum, then remove the starting elements**

**while (curr\_sum > sum && start < i-1)**

**{**

**curr\_sum = curr\_sum - arr[start];**

**start++;**

**}**

**// If curr\_sum becomes equal to sum, then return true**

**if (curr\_sum == sum)**

**{**

**int p = i-1;**

**System.out.println("Sum found between indexes " + start**

**+ " and " + p);**

**return 1;**

**}**

**// Add this element to curr\_sum**

**if (i < n)**

**curr\_sum = curr\_sum + arr[i];**

**}**

**System.out.println("No subarray found");**

**return 0;**

**}**

**7. Print largest contiguous array sum**

**import java.util.\*;**

**class MaxSubArraySum**

**{**

**public static void main (String[] args)**

**{**

**int [] a = {-2, -3, 4, -1, -2, 1, 5, -3};**

**System.out.println("Maximum contiguous sum is " + maxSubArraySum(a));**

**}**

**static int maxSubArraySum(int a[])**

**{**

**int size = a.length;**

**int max\_so\_far = Integer.MIN\_VALUE, max\_ending\_here = 0;**

**for (int i = 0; i < size; i++)**

**{**

**max\_ending\_here = max\_ending\_here + a[i];**

**if (max\_ending\_here < 0)**

**max\_ending\_here = 0;**

**/\* Do not compare for all elements. Compare only when max\_ending\_here > 0\*/**

**else if (max\_so\_far < max\_ending\_here)**

**max\_so\_far = max\_ending\_here;**

**}**

**return max\_so\_far;**

**}**

**static int maxSubArraySumMaxNegative(int a[], int size) {**

**int max\_so\_far = a[0];**

**int curr\_max = a[0];**

**for (int i = 1; i < size; i++) {**

**curr\_max = Math.*max*(a[i], curr\_max + a[i]);**

**max\_so\_far = Math.*max*(max\_so\_far, curr\_max);**

**}**

**return max\_so\_far;**

**}**

**}**

**8.check expression balance**

**import java.util.Stack;**

**public class Snippet {**

**static String isBalanced(char[] s) {**

**Stack<Character> stk = new Stack<Character>();**

**for (int i = 0; i < s.length; i++) {**

**if (s[i] == '(')**

**stk.push(')');**

**if (s[i] == '[')**

**stk.push(']');**

**if (s[i] == '{')**

**stk.push('}');**

**if (s[i] == '}' || s[i] == ')' || s[i] == ']') {**

**if (stk.peek() == s[i])**

**stk.pop();**

**}**

**}**

**if (stk.empty())**

**return "YES";**

**else**

**return "NO";**

**}**

**public static void main(String args[]) {**

**System.out.println(isBalanced("{(a+b\*c)/d}".toCharArray()));**

**}**

**}**

**9.print all distinct elements of an array**

**class DistinctElementsOfArray {**

**static void printDistinct(int arr[], int n)**

**{**

**for (int i = 0; i < n; i++)**

**{**

**int j;**

**for (j = 0; j < i; j++)**

**if (arr[i] == arr[j])**

**break;**

**// If not printed earlier, then print it**

**if (i == j)**

**System.out.print( arr[i] + " ");**

**}**

**}**

**public static void main (String[] args)**

**{**

**int arr[] = {6, 10, 5, 4, 9, 120, 4, 6, 10};**

**int n = arr.length;**

**printDistinct(arr, n);**

**}**

**}**

**10.Rotations of Array:**

**class RotateArray {**

**/\*Function to left rotate arr[] of size n by d\*/**

**void leftRotate(int arr[], int d, int n)**

**{**

**for (int i = 0; i < d; i++)**

**leftRotatebyOne(arr, n);**

**}**

**void leftRotatebyOne(int arr[], int n)**

**{**

**int i, temp;**

**temp = arr[0];**

**for (i = 0; i < n - 1; i++)**

**arr[i] = arr[i + 1];**

**arr[i] = temp;**

**}**

**/\* utility function to print an array \*/**

**void printArray(int arr[], int n)**

**{**

**for (int i = 0; i < n; i++)**

**System.out.print(arr[i] + " ");**

**}**

**// Driver program to test above functions**

**public static void main(String[] args)**

**{**

**RotateArray rotate = new RotateArray();**

**int arr[] = { 1, 2, 3, 4, 5, 6, 7 };**

**rotate.leftRotate(arr, 2, 7);**

**rotate.printArray(arr, 7);**

**}**

**}**

**METHOD 1 :(Using temp array)**

**Input arr[] = [1, 2, 3, 4, 5, 6, 7], d = 2, n =7**

**1) Store d elements in a temp array**

**temp[] = [1, 2]**

**2) Shift rest of the arr[]**

**arr[] = [3, 4, 5, 6, 7, 6, 7]**

**3) Store back the d elements**

**arr[] = [3, 4, 5, 6, 7, 1, 2]**

**11.check power of 2 or not**

**private static boolean powerOfTwo(int number){**

**int square = 1;**

**while(number >= square){**

**if(number == square){**

**return true;**

**}**

**square = square\*2;**

**}**

**return false;**

**}**

**12. Write a program to check whether a number is a palindrome or not?**

**Given number=reverse🡺palindrome**

**private static int reverse(int number){  
        int reverse = 0;  
        while(number != 0){  
          reverse = reverse\*10 + number%10;   
          number = number/10;  
        }             
        return reverse;  
    }**

**13.Armstrong:153**

**private static boolean isArmStrong(int number) {**

**int result = 0;**

**int orig = number;**

**while (number != 0) {**

**int remainder = number % 10;**

**result = result + remainder \* remainder \* remainder;**

**number = number / 10;**

**}**

**if (orig == result) {**

**return true;**

**}**

**return false;**

**}**

**14. find all prime number up to a given number?  
for(int number = 2; number<=limit; number++){  
          //print prime numbers only  
          if(isPrime(number)){  
              System.out.println(number);  
          }  
      }  
  
    }  
  
    /\*  
     \* Prime number is not divisible by any number other than 1 and itself  
     \* @return true if number is prime  
     \*/  
    public static boolean isPrime(int number){  
        for(int i=2; i<number; i++){  
           if(number%i == 0){  
               return false; //number is divisible so its not prime  
           }  
        }  
        return true; //number is prime now  
    }**

**15. Nth Fibonacci number?**

**public static int fibonacci(int number) {**

**if (number == 1 || number == 2) {**

**return 1;**

**}**

**return fibonacci(number - 1) + fibonacci(number - 2);**

**}**

**public static int fibonacci2(int number) {**

**if (number == 1 || number == 2) {**

**return 1;**

**}**

**int fibo1 = 1, fibo2 = 1, fibonacci = 1;**

**for (int i = 3; i <= number; i++) {**

**fibonacci = fibo1 + fibo2;**

**fibo1 = fibo2;**

**fibo2 = fibonacci;**

**}**

**return fibonacci;**

**}**

**16. find the sum of digits of a number**

**temp = num;**

**while (num > 0)**

**{**

**digit = num % 10;**

**sum = sum + digit;**

**num /= 10;**

**}**

**printf("Sum of the digits %ld = %ld\n", temp, sum);**

**17. swap two numbers without using temp variable 🡺a=a+b,b=a-b,a=a-b;**

**18. add two integers without using arithmetic operator**

**static int Add(int x, int y)**

**{**

**while (y != 0)**

**{**

**// carry now contains common set bits of x and y**

**int carry = x & y;**

**// Sum of bits of x and  y where at least one of the bits is not set**

**x = x ^ y;**

**// Carry is shifted by  one so that adding it  to x gives the required sum**

**y = carry << 1;**

**}**

**return x;**

**}**

**19.** [**Reverse digits**](https://practice.geeksforgeeks.org/problems/reverse-digit/0)

**int rev\_num = 0;**

**while (num > 0)**

**{**

**rev\_num = rev\_num\*10 + num%10;**

**num = num/10;**

**}**

**return rev\_num;**

**20. How to find continuous sub array whose sum is equal to given number**

**21.decimal to binary**

**while(number > 0){**

**binary[index++] = number%2;**

**number = number/2;**

**}**

**1.HashMap**

**static class Entry<K ,V> implements Map.Entry<K, V>**

**{**

**final K k;**

**V v;**

**Entry<K ,V> next;**

**final int hash;**

**Entry(K k,V v,hash)**

**{**

**this.k=k;**

**this.v= v;**

**this.hash=hash;**

**}**

**}**

**Class HashMap<K,V>{**

**Entry[] table;**

**HashMap(int size){**

**table=new Entry[size];**

**}**

**}**

**static int indexFor(int h, int length) {  
 return h & (length-1);  
}**

**public V put(K key, V value) {**

**if (key == null)**

**return putForNullKey(value);**

**int hash = hash(key.hashCode());**

**int i = indexFor(hash, table.length);**

**for (Entry<K , V> e = table[i]; e != null; e = e.next) {**

**Object k;**

**if (e.hash == hash && ((k = e.key) == key || key.equals(k))) {**

**V oldValue = e.value;**

**e.value = value;**

**e.recordAccess(this);**

**return oldValue;**

**}**

**}**

**modCount++;**

**addEntry(hash, key, value, i);**

**return null;**

**}**

**public V get(Object key) {**

**if (key == null)**

**return getForNullKey();**

**int hash = hash(key.hashCode());**

**for (Entry<K , V> e = table[indexFor(hash, table.length)]; e != null; e = e.next) {**

**Object k;**

**if (e.hash == hash && ((k = e.key) == key || key.equals(k)))**

**return e.value;**

**}**

**return null;**

**}**

**2.Linked HashMap**

**\*which allows us to store data in key-value pair form.**

**\* It maintains insertion order, uses DoublyLinkedList for doing so.**

**\* If key which already exists is added again, its value is overridden but**

**\* insertion order does not change,**

**\* BUT, if key-value pair is removed and value is again added than insertion order changes(which is quite natural behavior).**

**class LinkedHashMapCustom<K, V> {**

**private Entry<K,V>[] table;   //Array of Entry.**

**private int capacity= 4;  //Initial capacity of HashMap**

**private Entry<K,V> header; //head of the doubly linked list.**

**private Entry<K,V> last; //last of the doubly linked list.**

**/\* before and after are used for maintaining insertion order.\*/**

**static class Entry<K, V> {**

**K key;**

**V value;**

**Entry<K,V> next;**

**Entry<K,V> before,after;**

**public Entry(K key, V value, Entry<K,V> next){**

**this.key = key;**

**this.value = value;**

**this.next = next;**

**}**

**}**

**@SuppressWarnings("unchecked")**

**public LinkedHashMapCustom(){**

**table = new Entry[capacity];**

**}**

**/\*\***

**\* Method allows you put key-value pair in LinkedHashMapCustom.**

**\* If the map already contains a mapping for the key, the old value is replaced.**

**\* Note: method does not allows you to put null key thought it allows null values.**

**\* Implementation allows you to put custom objects as a key as well.**

**\* Key Features: implementation provides you with following features:-**

**\*     >provide complete functionality how to override equals method.**

**\*  >provide complete functionality how to override hashCode method.**

**public void put(K newKey, V data){**

**if(newKey==null)**

**return;    //does not allow to store null.**

**int hash=hash(newKey);**

**Entry<K,V> newEntry = new Entry<K,V>(newKey, data, null);**

**maintainOrderAfterInsert(newEntry);**

**if(table[hash] == null){**

**table[hash] = newEntry;**

**}else{**

**Entry<K,V> previous = null;**

**Entry<K,V> current = table[hash];**

**while(current != null){ //we have reached last entry of bucket.**

**if(current.key.equals(newKey)){**

**if(previous==null){  //node has to be insert on first of bucket.**

**newEntry.next=current.next;**

**table[hash]=newEntry;**

**return;**

**}**

**else{**

**newEntry.next=current.next;**

**previous.next=newEntry;**

**return;**

**}**

**}**

**previous=current;**

**current = current.next;**

**}**

**previous.next = newEntry;**

**}**

**}**

**/\*\***

**\* below method helps us in ensuring insertion order of LinkedHashMapCustom**

**\* after new key-value pair is added.**

**\*/**

**private void maintainOrderAfterInsert(Entry<K, V> newEntry) {**

**if(header==null){**

**header=newEntry;**

**last=newEntry;**

**return;**

**}**

**if(header.key.equals(newEntry.key)){**

**deleteFirst();**

**insertFirst(newEntry);**

**return;**

**}**

**if(last.key.equals(newEntry.key)){**

**deleteLast();**

**insertLast(newEntry);**

**return;**

**}**

**Entry<K, V> beforeDeleteEntry=    deleteSpecificEntry(newEntry);**

**if(beforeDeleteEntry==null){**

**insertLast(newEntry);**

**}**

**else{**

**insertAfter(beforeDeleteEntry,newEntry);**

**}**

**}**

**/\*\***

**\* below method helps us in ensuring insertion order of LinkedHashMapCustom,**

**\* after deletion of key-value pair.**

**\*/**

**private void maintainOrderAfterDeletion(Entry<K, V> deleteEntry) {**

**if(header.key.equals(deleteEntry.key)){**

**deleteFirst();**

**return;**

**}**

**if(last.key.equals(deleteEntry.key)){**

**deleteLast();**

**return;**

**}**

**deleteSpecificEntry(deleteEntry);**

**}**

**/\*\***

**\* returns entry after which new entry must be added.**

**\*/**

**private void insertAfter(Entry<K, V> beforeDeleteEntry, Entry<K, V> newEntry) {**

**Entry<K, V> current=header;**

**while(current!=beforeDeleteEntry){**

**current=current.after; //move to next node.**

**}**

**newEntry.after=beforeDeleteEntry.after;**

**beforeDeleteEntry.after.before=newEntry;**

**newEntry.before=beforeDeleteEntry;**

**beforeDeleteEntry.after=newEntry;**

**}**

**/\*\***

**\* deletes entry from first.**

**\*/**

**private void deleteFirst(){**

**if(header==last){ //only one entry found.**

**header=last=null;**

**return;**

**}**

**header=header.after;**

**header.before=null;**

**}**

**\* inserts entry at first.**

**private void insertFirst(Entry<K, V> newEntry){**

**if(header==null){ //no entry found**

**header=newEntry;**

**last=newEntry;**

**return;**

**}**

**newEntry.after=header;**

**header.before=newEntry;**

**header=newEntry;**

**}**

**/\*\***

**\* inserts entry at last.**

**\*/**

**private void insertLast(Entry<K, V> newEntry){**

**if(header==null){**

**header=newEntry;**

**last=newEntry;**

**return;**

**}**

**last.after=newEntry;**

**newEntry.before=last;**

**last=newEntry;**

**}**

**/\*\***

**\* deletes entry from last.**

**\*/**

**private void deleteLast(){**

**if(header==last){**

**header=last=null;**

**return;**

**}**

**last=last.before;**

**last.after=null;**

**}**

**\* deletes specific entry and returns before entry.**

**private Entry<K, V> deleteSpecificEntry(Entry<K, V> newEntry){**

**Entry<K, V> current=header;**

**while(!current.key.equals(newEntry.key)){**

**if(current.after==null){   //entry not found**

**return null;**

**}**

**current=current.after; //move to next node.**

**}**

**Entry<K, V> beforeDeleteEntry=current.before;**

**current.before.after=current.after;**

**current.after.before=current.before; //entry deleted**

**return beforeDeleteEntry;**

**}**

**\* Method returns value corresponding to key.**

**public V get(K key){**

**int hash = hash(key);**

**if(table[hash] == null){**

**return null;**

**}else{**

**Entry<K,V> temp = table[hash];**

**while(temp!= null){**

**if(temp.key.equals(key))**

**return temp.value;**

**temp = temp.next; //return value corresponding to key.**

**}**

**return null;   //returns null if key is not found.**

**}**

**}**

**/\*\***

**\* Method removes key-value pair from HashMapCustom.**

**\* @param key**

**\*/**

**public boolean remove(K deleteKey){**

**int hash=hash(deleteKey);**

**if(table[hash] == null){**

**return false;**

**}else{**

**Entry<K,V> previous = null;**

**Entry<K,V> current = table[hash];**

**while(current != null){ //we have reached last entry node of bucket.**

**if(current.key.equals(deleteKey)){**

**maintainOrderAfterDeletion(current);**

**if(previous==null){  //delete first entry node.**

**table[hash]=table[hash].next;**

**return true;**

**}**

**else{**

**previous.next=current.next;**

**return true;**

**}**

**}**

**previous=current;**

**current = current.next;**

**}**

**return false;**

**}**

**}**

**/\*\***

**\* Method displays all key-value pairs present in HashMapCustom.,**

**\* insertion order is not guaranteed, for maintaining insertion order**

**\* refer linkedHashMapCustom.**

**\* @param key**

**\*/**

**public void display(){**

**Entry<K, V> currentEntry=header;**

**while(currentEntry!=null){**

**System.*out*.print("{"+currentEntry.key+"="+currentEntry.value+"}" +" ");**

**currentEntry=currentEntry.after;**

**}**

**}**

**/\*\***

**\* Method implements hashing functionality, which helps in finding the appropriate**

**\* bucket location to store our data.**

**\* This is very important method, as performance of HashMapCustom is very much**

**\* dependent on this method's implementation.**

**\* @param key**

**\*/**

**private int hash(K key){**

**return Math.*abs*(key.hashCode()) % capacity;**

**}**

**}**

Example 1: Program to print half pyramid using \*

**\***

**\* \***

**\* \* \***

**\* \* \* \***

**\* \* \* \* \***

**Source Code**

**#include <stdio.h>**

**int main()**

**{**

**int i, j, rows;**

**printf("Enter number of rows: ");**

**scanf("%d",&rows);**

**for(i=1; i<=rows; ++i)**

**{**

**for(j=1; j<=i; ++j)**

**{**

**printf("\* ");**

**}**

**printf("\n");**

**}**

**return 0;**

**}**

### Example 2: Program to print half pyramid a using numbers

**1**

**1 2**

**1 2 3**

**1 2 3 4**

**1 2 3 4 5**

**Source Code**

**#include <stdio.h>**

**int main()**

**{**

**int i, j, rows;**

**printf("Enter number of rows: ");**

**scanf("%d",&rows);**

**for(i=1; i<=rows; ++i)**

**{**

**for(j=1; j<=i; ++j)**

**{**

**printf("%d ",j);**

**}**

**printf("\n");**

**}**

**return 0;**

**}**

### Example 3: Program to print half pyramid using alphabets

**A**

**B B**

**C C C**

**D D D D**

**E E E E E**

**Source Code**

**#include <stdio.h>**

**int main()**

**{**

**int i, j;**

**char input, alphabet = 'A';**

**printf("Enter the uppercase character you want to print in last row: ");**

**scanf("%c",&input);**

**for(i=1; i <= (input-'A'+1); ++i)**

**{**

**for(j=1;j<=i;++j)**

**{**

**printf("%c", alphabet);**

**}**

**++alphabet;**

**printf("\n");**

**}**

**return 0;**

**}**

## Programs to print inverted half pyramid using \* and numbers

### Example 4: Inverted half pyramid using \*

**\* \* \* \* \***

**\* \* \* \***

**\* \* \***

**\* \***

**\***

**Source Code**

**#include <stdio.h>**

**int main()**

**{**

**int i, j, rows;**

**printf("Enter number of rows: ");**

**scanf("%d",&rows);**

**for(i=rows; i>=1; --i)**

**{**

**for(j=1; j<=i; ++j)**

**{**

**printf("\* ");**

**}**

**printf("\n");**

**}**

**return 0;**

**}**

### Example 5: Inverted half pyramid using numbers

**1 2 3 4 5**

**1 2 3 4**

**1 2 3**

**1 2**

**1**

**Source Code**

**#include <stdio.h>**

**int main()**

**{**

**int i, j, rows;**

**printf("Enter number of rows: ");**

**scanf("%d",&rows);**

**for(i=rows; i>=1; --i)**

**{**

**for(j=1; j<=i; ++j)**

**{**

**printf("%d ",j);**

**}**

**printf("\n");**

**}**

**return 0;**

**}**

## Programs to display pyramid and inverted pyramid using \* and digits

### Example 6: Program to print full pyramid using \*

**\***

**\* \* \***

**\* \* \* \* \***

**\* \* \* \* \* \* \***

**\* \* \* \* \* \* \* \* \***

**Source Code**

**#include <stdio.h>**

**int main()**

**{**

**int i, space, rows, k=0;**

**printf("Enter number of rows: ");**

**scanf("%d",&rows);**

**for(i=1; i<=rows; ++i, k=0)**

**{**

**for(space=1; space<=rows-i; ++space)**

**{**

**printf(" ");**

**}**

**while(k != 2\*i-1)**

**{**

**printf("\* ");**

**++k;**

**}**

**printf("\n");**

**}**

**return 0;**

**}**

### Example 7: Program to print pyramid using numbers

**1**

**2 3 2**

**3 4 5 4 3**

**4 5 6 7 6 5 4**

**5 6 7 8 9 8 7 6 5**

**Source Code**

**#include <stdio.h>**

**int main()**

**{**

**int i, space, rows, k=0, count = 0, count1 = 0;**

**printf("Enter number of rows: ");**

**scanf("%d",&rows);**

**for(i=1; i<=rows; ++i)**

**{**

**for(space=1; space <= rows-i; ++space)**

**{**

**printf(" ");**

**++count;**

**}**

**while(k != 2\*i-1)**

**{**

**if (count <= rows-1)**

**{**

**printf("%d ", i+k);**

**++count;**

**}**

**else**

**{**

**++count1;**

**printf("%d ", (i+k-2\*count1));**

**}**

**++k;**

**}**

**count1 = count = k = 0;**

**printf("\n");**

**}**

**return 0;**

**}**

### Example 8: Inverted full pyramid using \*

**\* \* \* \* \* \* \* \* \***

**\* \* \* \* \* \* \***

**\* \* \* \* \***

**\* \* \***

**\***

**Source Code**

**#include<stdio.h>**

**int main()**

**{**

**int rows, i, j, space;**

**printf("Enter number of rows: ");**

**scanf("%d",&rows);**

**for(i=rows; i>=1; --i)**

**{**

**for(space=0; space < rows-i; ++space)**

**printf(" ");**

**for(j=i; j <= 2\*i-1; ++j)**

**printf("\* ");**

**for(j=0; j < i-1; ++j)**

**printf("\* ");**

**printf("\n");**

**}**

**return 0;**

**}**

### Example 9: Print Pascal's triangle

**1**

**1 1**

**1 2 1**

**1 3 3 1**

**1 4 6 4 1**

**1 5 10 10 5 1**

**Source Code**

**#include <stdio.h>**

**int main()**

**{**

**int rows, coef = 1, space, i, j;**

**printf("Enter number of rows: ");**

**scanf("%d",&rows);**

**for(i=0; i<rows; i++)**

**{**

**for(space=1; space <= rows-i; space++)**

**printf(" ");**

**for(j=0; j <= i; j++)**

**{**

**if (j==0 || i==0)**

**coef = 1;**

**else**

**coef = coef\*(i-j+1)/j;**

**printf("%4d", coef);**

**}**

**printf("\n");**

**}**

**return 0;**

**}**

### Example 10: Print Floyd's Triangle.

**1**

**2 3**

**4 5 6**

**7 8 9 10**

**Source Code**

**#include <stdio.h>**

**int main()**

**{**

**int rows, i, j, number= 1;**

**printf("Enter number of rows: ");**

**scanf("%d",&rows);**

**for(i=1; i <= rows; i++)**

**{**

**for(j=1; j <= i; ++j)**

**{**

**printf("%d ", number);**

**++number;**

**}**

**printf("\n");**

**}**

**return 0;**

**}**

**String**

**1.How do you print duplicate characters from a string?**

**package String;**

**public class DuplicatesInString {**

**static final int NO\_OF\_CHARS = 256;**

**/\* Fills count array with frequency of characters \*/**

**static void fillCharCounts(String str, int[] count) {**

**for (int i = 0; i < str.length(); i++)**

**count[str.charAt(i)]++;**

**}**

**/\* Print duplicates present in the passed string \*/**

**static void printDups(String str) {**

**// Create an array of size 256 and fill count of every character in it**

**int count[] = new int[NO\_OF\_CHARS];**

**fillCharCounts(str, count);**

**for (int i = 0; i < NO\_OF\_CHARS; i++)**

**if (count[i] > 1)**

**System.out.printf("%c, count = %d \n", i, count[i]);**

**}**

**public static void printDuplicateCharacters(String word) {**

**char[] characters = word.toCharArray();**

**Map<Character, Integer> charMap = new HashMap<Character, Integer>();**

**for (Character ch : characters) {**

**if (charMap.containsKey(ch)) {**

**charMap.put(ch, charMap.get(ch) + 1);**

**} else {**

**charMap.put(ch, 1);**

**}**

**}**

**Set<Map.Entry<Character, Integer>> entrySet = charMap.entrySet();**

**System.*out*.printf("List of duplicate characters in String '%s' %n", word);**

**for (Map.Entry<Character, Integer> entry : entrySet) {**

**if (entry.getValue() > 1) {**

**System.*out*.printf("%s : %d %n", entry.getKey(), entry.getValue());**

**}**

**}**

**}**

**public static void main(String[] args) {**

**String str = "test string";**

**printDups(str);**

**}**

**}**

**2. Check whether two strings are anagram of each other**

**package String;**

**import java.util.Arrays;**

**public class AnagramString {**

**public static int NO\_OF\_CHARS = 256;**

**static boolean areAnagram(char str1[], char str2[]) {**

**int count[] = new int[NO\_OF\_CHARS];**

**Arrays.fill(count, 0);**

**int i;**

**for (i = 0; i < str1.length && i < str2.length; i++) {**

**count[str1[i]]++;**

**count[str2[i]]--;**

**}**

**if (str1.length != str2.length)**

**return false;**

**for (i = 0; i < NO\_OF\_CHARS; i++)**

**if (count[i] > 0)**

**return false;**

**return true;**

**}**

**public static void main(String[] args) {**

**char str1[] = ("geeksforgeeks").toCharArray();**

**char str2[] = ("forgeeksgeeks").toCharArray();**

**areAnagram(str1, str2);**

**}**

**}**

**3. Reverse words in a given string**

**class ReverseWordsString {**

**static String wordReverse(String str) {**

**int i = str.length() - 1;**

**int start, end = i + 1;**

**String result = "";**

**while (i >= 0) {**

**if (str.charAt(i) == ' ') {**

**start = i + 1;**

**while (start != end)**

**result += str.charAt(start++);**

**result += ' ';**

**end = i;**

**}**

**i--;**

**}**

**start = 0;**

**while (start != end)**

**result += str.charAt(start++);**

**return result;**

**}**

**public static void main(String[] args) {**

**String str = "I AM A GEEK";**

**System.out.print(wordReverse(str));**

**}**

**}**

**4. All permutations of a given string**

**public class PermutationsString {**

**public static void main(String[] args)**

**{**

**String str = "ABC";**

**int n = str.length();**

**PermutationsString permutation = new PermutationsString();**

**permutation.permute(str, 0, n-1);**

**}**

**private void permute(String str, int l, int r)**

**{**

**if (l == r)**

**System.out.println(str);**

**else**

**{**

**for (int i = l; i <= r; i++)**

**{**

**str = swap(str,l,i);**

**permute(str, l+1, r);**

**str = swap(str,l,i);**

**}**

**}**

**}**

**public String swap(String a, int i, int j)**

**{**

**char temp;**

**char[] charArray = a.toCharArray();**

**temp = charArray[i] ;**

**charArray[i] = charArray[j];**

**charArray[j] = temp;**

**return String.valueOf(charArray);**

**}**

**}**

**5.Reverse Array:**

**static void rvereseArray(int arr[],**

**int start, int end)**

**{**

**int temp;**

**while (start < end)**

**{**

**temp = arr[start];**

**arr[start] = arr[end];**

**arr[end] = temp;**

**start++;**

**end--;**

**}**

**}**

**6. first non-repeating character**

**static final int NO\_OF\_CHARS = 256;**

**static char count[] = new char[NO\_OF\_CHARS];**

**static void getCharCountArray(String str)**

**{**

**for (int i = 0; i < str.length();  i++)**

**count[str.charAt(i)]++;**

**}**

**static int firstNonRepeating(String str)**

**{**

**getCharCountArray(str);**

**int index = -1, i;**

**for (i = 0; i < str.length();  i++)**

**{**

**if (count[str.charAt(i)] == 1)**

**{**

**index = i;**

**break;**

**}**

**}**

**return index;**

**}**

**7. strings are rotations of each other or not**

**static boolean areRotations(String str1, String str2)**

**{**

**// There lengths must be same and str2 must be a substring of str1 concatenated with str1.**

**return (str1.length() == str2.length()) && ((str1 + str1).indexOf(str2) != -1);**

**}**

**8. Given string is a substring of another string**

**class checkSubstring {**

**static int isSubstring(String s1, String s2)**

**{**

**int M = s1.length();**

**int N = s2.length();**

**for (int i = 0; i <= N - M; i++) {**

**int j;**

**for (j = 0; j < M; j++)**

**if (s2.charAt(i + j) != s1.charAt(j))**

**break;**

**if (j == M)**

**return i;**

**}**

**return -1;**

**}**

**/\* Driver program to test above function \*/**

**public static void main(String args[])**

**{**

**String s1 = "for";**

**String s2 = "geeksforgeeks";**

**int res = isSubstring(s1, s2);**

**if (res == -1)**

**System.out.println("Not present");**

**else**

**System.out.println("Present at index "+ res);**

**}**

**}**

**9. Program to print all substrings of a given string**

**public class AllSubstrings {**

**static void subString(char str[], int n)**

**{**

**for (int len = 1; len <= n; len++)**

**{**

**for (int i = 0; i <= n - len; i++)**

**{**

**int j = i + len - 1;**

**for (int k = i; k <= j; k++)**

**System.out.print(str[k]);**

**System.out.println("");**

**}**

**}**

**}**

**public static void main(String args[])**

**{**

**String s="abc";**

**subString(s.toCharArray(), s.length());**

**}**

**}**

**10. Write code to check a String is palindrome or not?   
void isPalindrome(char str[]) {**

**// Start from leftmost and rightmost corners of str**

**int l = 0;**

**int h = strlen(str) - 1;**

**// Keep comparing characters while they are same**

**while (h > l)**

**{**

**if (str[l++] != str[h--])**

**{**

**printf("%s is Not Palindrome", str);**

**return;**

**}**

**}**

**printf("%s is palindrome", str);**

**}**

**DS**

1. **How do you find the middle element of a singly linked list in one pass?**
2. **How to find the 3rd/nth element from the end of linked list in Java?**
3. **How do you check if a given linked list contains a cycle? How do you find the starting node of the cycle?**
4. **How do you reverse a single and double linked list with and without recursion?**
5. **How are duplicate nodes removed in an unsorted/sorted linked list?**
6. **How do you find the sum of two linked lists using Stack?**
7. [**Rotate a Linked List**](https://practice.geeksforgeeks.org/problems/rotate-a-linked-list/1)
8. [**Merge two sorted linked lists**](https://practice.geeksforgeeks.org/problems/merge-two-sorted-linked-lists/1)
9. [**Intersection point of two Linked Lists**](https://practice.geeksforgeeks.org/problems/intersection-point-in-y-shapped-linked-lists/1)
10. [**Add two numbers represented by linked lists**](https://practice.geeksforgeeks.org/problems/add-two-numbers-represented-by-linked-lists/1)
11. [**Implement Queue using Linked List**](https://practice.geeksforgeeks.org/problems/implement-queue-using-linked-list/1)
12. [**Implement Stack using Linked List**](https://practice.geeksforgeeks.org/problems/implement-stack-using-linked-list/1)
13. [**Delete without head pointer**](https://practice.geeksforgeeks.org/problems/delete-without-head-pointer/1)
14. **How is a binary search tree implemented?**
15. **How do you perform preorder(inorder,postorder) traversal in a given binary tree?**
16. **How do you traverse a given binary tree in preorder(inorder,postorder) without recursion?**
17. **How do you find the depth of a binary tree?**
18. **How do you print all nodes of a given binary tree using inorder traversal without recursion?**
19. **How are all leaves of a binary search tree printed?**
20. **How do you count a number of leaf nodes in a given binary tree?**
21. **How do you perform a binary search in a given array**
22. [**Check for BST**](https://practice.geeksforgeeks.org/problems/check-for-bst/1)
23. **BFS,DFS**
24. [**Lowest Common Ancestor in a BST**](https://practice.geeksforgeeks.org/problems/lowest-common-ancestor-in-a-bst/1)
25. [**Height of Binary Tree**](https://practice.geeksforgeeks.org/problems/height-of-binary-tree/1)
26. [**Maximum Path Sum**](https://practice.geeksforgeeks.org/problems/maximum-path-sum/1)
27. [**Number of leaf nodes**](https://practice.geeksforgeeks.org/problems/count-leaves-in-binary-tree/1)
28. **Sortings**
29. **Java program to find middle of linked list**

**class FindMiddleNodeInLinkedList**

**{**

**Node head; // head of linked list**

**/\* Linked list node \*/**

**class Node**

**{**

**int data;**

**Node next;**

**Node(int d)**

**{**

**data = d;**

**next = null;**

**}**

**}**

**/\* Function to print middle of linked list \*/**

**void printMiddle()**

**{**

**Node slow\_ptr = head;**

**Node fast\_ptr = head;**

**if (head != null)**

**{**

**while (fast\_ptr != null && fast\_ptr.next != null)**

**{**

**fast\_ptr = fast\_ptr.next.next;**

**slow\_ptr = slow\_ptr.next;**

**}**

**System.*out*.println("The middle element is [" + slow\_ptr.data + "] \n");**

**}**

**}**

**/\* Inserts a new Node at front of the list. \*/**

**public void push(int new\_data)**

**{**

**/\* 1 & 2: Allocate the Node & Put in the data\*/**

**Node new\_node = new Node(new\_data);**

**/\* 3. Make next of new Node as head \*/**

**new\_node.next = head;**

**/\* 4. Move the head to point to new Node \*/**

**head = new\_node;**

**}**

**/\* This function prints contents of linked list starting from the given node \*/**

**public void printList()**

**{**

**Node tnode = head;**

**while (tnode != null)**

**{**

**System.*out*.print(tnode.data+"->");**

**tnode = tnode.next;**

**}**

**System.*out*.println("NULL");**

**}**

**}**

**2. Java program to find n'th node from end using slow and**

**// fast pointers**

**class NthNodeFromEndLinkedList**

**{**

**Node head; // head of the list**

**/\* Linked List node \*/**

**/\* Function to get the nth node from end of list \*/**

**void printNthFromLast(int n)**

**{**

**Node main\_ptr = head;**

**Node ref\_ptr = head;**

**int count = 0;**

**if (head != null)**

**{**

**while (count < n)**

**{**

**if (ref\_ptr == null)**

**{**

**System.out.println(n+" is greater than the no "+ " of nodes in the list");**

**return;**

**}**

**ref\_ptr = ref\_ptr.next;**

**count++;**

**}**

**while (ref\_ptr != null)**

**{**

**main\_ptr = main\_ptr.next;**

**ref\_ptr = ref\_ptr.next;**

**}**

**System.out.println("Node no. "+n+" from last is "+main\_ptr.data);**

**}**

**}**

**/\* Inserts a new Node at front of the list. \*/**

**public void push(int new\_data)**

**{**

**/\* 1 & 2: Allocate the Node & Put in the data\*/**

**Node new\_node = new Node(new\_data);**

**/\* 3. Make next of new Node as head \*/**

**new\_node.next = head;**

**/\* 4. Move the head to point to new Node \*/**

**head = new\_node;**

**}**

**3. Java program to detect and remove loop in linked list**

**class DetectAndRemoveLoopLinkedList {**

**static Node head;**

**// Function that detects loop in the list**

**int detectAndRemoveLoop(Node node) {**

**Node slow = node, fast = node;**

**while (slow != null && fast != null && fast.next != null) {**

**slow = slow.next;**

**fast = fast.next.next;**

**// If slow and fast meet at same point then loop is present**

**if (slow == fast) {**

**removeLoop(slow, node);**

**return 1;**

**}**

**}**

**return 0;**

**}**

**// Function to remove loop**

**void removeLoop(Node loop, Node curr) {**

**Node ptr1 = null, ptr2 = null;**

**/\* Set a pointer to the beging of the Linked List and**

**move it one by one to find the first node which is**

**part of the Linked List \*/**

**ptr1 = curr;**

**while (1 == 1) {**

**/\* Now start a pointer from loop\_node and check if it ever**

**reaches ptr2 \*/**

**ptr2 = loop;**

**while (ptr2.next != loop && ptr2.next != ptr1) {**

**ptr2 = ptr2.next;**

**}**

**/\* If ptr2 reahced ptr1 then there is a loop. So break the**

**loop \*/**

**if (ptr2.next == ptr1) {**

**break;**

**}**

**/\* If ptr2 did't reach ptr1 then try the next node after ptr1 \*/**

**ptr1 = ptr1.next;**

**}**

**/\* After the end of loop ptr2 is the last node of the loop. So make next of ptr2 as NULL \*/**

**ptr2.next = null;**

**}**

**// Function to print the linked list**

**void printList(Node node) {**

**while (node != null) {**

**System.out.print(node.data + " ");**

**node = node.next;**

**}**

**}**

**}**

**4(i).Reverse Single Linked List:**

**Node reverse(Node node) {**

**Node prev = null;**

**Node current = node;**

**Node next = null;**

**while (current != null) {**

**next = current.next;**

**current.next = prev;**

**prev = current;**

**current = next;**

**}**

**node = prev;**

**return node;**

**}**

**public Node reverseRec(Node prev, Node curr) {**

**if (curr == null)**

**return null;**

**if (curr.next == null) {**

**curr.next = prev;**

**return curr;**

**} else {**

**Node temp = curr.next;**

**curr.next = prev;**

**return reverseRec(curr, temp);**

**}**

**}**

**4(ii). Reverse double Linked List:**

**class ReverseDoubleLinkedList {**

**static Node head;**

**void reverse() {**

**Node temp = null;**

**Node current = head;**

**/\* swap next and prev for all nodes of**

**doubly linked list \*/**

**while (current != null) {**

**temp = current.prev;**

**current.prev = current.next;**

**current.next = temp;**

**current = current.prev;**

**}**

**/\* Before changing head, check for the cases like empty**

**list and list with only one node \*/**

**if (temp != null) {**

**head = temp.prev;**

**}**

**}**

**function to reverse a doubly linked list :**

**Node Reverse(Node node)**

**{**

**// If empty list, return**

**if (!node)**

**return NULL;**

**// Otherwise, swap the next and prev**

**Node temp = node.next;**

**Node.next = node.prev;**

**Node.prev = temp;**

**// If the prev is now NULL, the list has been fully reversed**

**if (node.prev==null)**

**return node;**

**// Otherwise, keep going**

**return Reverse(node->prev);**

**}**

**5:Remove duplicates from sorted Linkedlist**

**void removeDuplicates()**

**{**

**/\*Another reference to head\*/**

**Node current = head;**

**/\* Pointer to store the next pointer of a node to be deleted\*/**

**Node next\_next;**

**/\* do nothing if the list is empty \*/**

**if (head == null)**

**return;**

**/\* Traverse list till the last node \*/**

**while (current.next != null) {**

**/\*Compare current node with the next node \*/**

**if (current.data == current.next.data) {**

**next\_next = current.next.next;**

**current.next = null;**

**current.next = next\_next;**

**}**

**else // advance if no deletion**

**current = current.next;**

**}**

**}**