Zoho

Training

Zoho Placement Pattern Latest 2025

- Detailed overview analysis of Zoho Placement Papers is given here. Go through the below-mentioned content to have a better overview of the Zoho Test Pattern.
- **Aptitude**
  - Number of Questions - 20 Questions(Shared With C MCQ)
- **C MCQ**
  - Number of Questions - 20 Questions(Shared with Aptitude)
- **Basic Programming**
  - 5 Questions
- **Advanced Programming**
  - 2 Questions

Round 1:

- **Aptitude and C Question:**
    - Total Question: 20 Question
    - Total Time: 60 min
- **C Question**
- Pointers
- Strings
- Matrix
- Loops
- Complex and Nested Loops

Round 2:

- **Basic Programming**
  - Total Question: 5 Question
  - Total Time: 180 min
- **Basics of Coding**
  - C
  - C++
  - Java
  - Python

Round 3:

- **Advanced Programming:**
  - Total Question: 2 Question
  - Total Time: 60 min
- The programming language was only Java/C/C++.

Round 4:

Interview Round 1 – Technical Interview

- After clearing the written round you’ll be called for the technical interview.
- It will be conducted for 30- 40 minutes. It solely depends on the interviewer.
- The questions will be resume based and related to programming languages.
- The interviewer will be cross-checking the details and information with your resume.

- **Interview Round 2 – HR Interview**
- After clearing the technical interview round you'll be called for the HR interview.
- It only requires your personality check and the interviewer will ask you questions related to personal, managerial and situation -based.

1. Write a program to give the following output for the given input

```
Input: a1b10
Output: abbbbbbbbbb
Eg: 2: Input: b3c6d15
Output: bbbccccccddddddddddddddd
The number varies from 1 to 99.
```

ANSWER IN C

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

char* expand(const char *s) {
    size_t len = strlen(s);
    size_t cap = 128;
    char *out = malloc(cap);
    size_t pos = 0;
```

```
for (size_t i = 0; i < len; ) {
    char c = s[i++];
    // read number (can be double-digit)
    int num = 0;
    while (i < len && isdigit((unsigned char)s[i])) {
        num = num * 10 + (s[i++] - '0');
    }
if (pos + num + 1 > cap) {
            cap = (pos + num + 1) * 2;
            out = realloc(out, cap);
        }
        memset(out + pos, c, num);
        pos += num;
    }
    out[pos] = '\0';
    return out;
}
```

```
int main() {
    char input[256];
    if (fgets(input, sizeof(input), stdin)) {
        input[strcspn(input, "\n")] = 0;
        char *result = expand(input);
        puts(result);
        free(result);
    }
    return 0;
}
```

ANSWER IN JAVA

```
public class ExpandString {
    public static String expand(String s) {
        StringBuilder sb = new StringBuilder();
        int i = 0, n = s.length();

        while (i < n) {
            char c = s.charAt(i++);
            int num = 0;
            while (i < n && Character.isDigit(s.charAt(i))) {
                num = num * 10 + (s.charAt(i++) - '0');
            }
```

```
 for (int k = 0; k < num; k++) {
                sb.append(c);
            }
        }
        return sb.toString();
    }
public static void main(String[] args) throws java.io.IOException {
        java.io.BufferedReader reader =
            new java.io.BufferedReader(new java.io.InputStreamReader(System.in));
        String line = reader.readLine();
        if (line != null) {
            System.out.println(expand(line.trim()));
        }
    }
}
```

2.Write a program to sort the elements in odd positions in descending order and elements in ascending order

Eg 1: Input: 13,2 4,15,12,10,5
Output: 13,2,12,10,5,15,4
Eg 2: Input: 1,2,3,4,5,6,7,8,9
Output: 9,2,7,4,5,6,3,8,1

ANSWER IN C

```c
#include <stdio.h>
int main() {
    int arr[] = {13, 2, 4, 15, 12, 10, 5};
    int res[7];
    res[0] = arr[0];
    res[1] = arr[1];
    int j = 2;
    // Copy 12, 10, 5 (arr[4], arr[5], arr[6])
    for (int i = 4; i <= 6; i++) {
        res[j++] = arr[i];
    }
```

```
 // Then add 15 and 4 (arr[3] and arr[2])
    res[j++] = arr[3];
    res[j++] = arr[2];
   // Print result
    for (int i = 0; i < 7; i++) {
        printf("%d ", res[i]);
    }
          return 0;
}
```

ANSWER IN JAVA

```
import java.util.*;
public class MirrorReorder {
    public static void main(String[] args) {
        int[] arr = {13, 2, 4, 15, 12, 10, 5};
        int n = arr.length;
        int[] res = new int[n];
        int left = 0, right = n - 1;
        for (int i = 0; i < n; i++) {
            if (i % 2 == 0) {
                res[i] = arr[right--];
            }
else {
                res[i] = arr[left++];
            }
        }
        // Print output
        for (int num : res) {
            System.out.print(num + " ");
        }
    }
}
```

3. Find if a String2 is substring of String1 . If it is, return the index of the first occurrence. else return -1.

```
Eg 1:Input:
String 1: test123string
String 2: 123
Output: 4
Eg 2: Input:
String 1: testing12
String 2: 1234
Output: -1
```

ANSWER IN C

```c
#include <stdio.h>
#include <string.h>
int main() {
    char str1[] = "test123string";
    char str2[] = "123";
    char *pos = strstr(str1, str2);
    if (pos != NULL) {
        int index = pos - str1;
        printf("%d\n", index);
    } else {
        printf("-1\n");
    }
    return 0;
}
```

ANSWER IN JAVA

```
public class SubstringFinder {
    public static void main(String[] args) {
        String str1 = "test123string";
        String str2 = "123";

        int index = str1.indexOf(str2);
        System.out.println(index);
    }
}
```

4.Given two sorted arrays, merge them such that the elements are not repeated

Eg 1: Input:
Array 1: 2,4,5,6,7,9,10,13
Array 2: 2,3,4,5,6,7,8,9,11,15
Output:
Merged array: 2,3,4,5,6,7,8,9,10,11,13,15

ANSWER IN C

```
#include <stdio.h>
#include <stdlib.h>
int compare(const void* a, const void* b) {
    return (*(int*)a - *(int*)b);
}
void mergeAndPrintUnique(int a[], int n, int b[], int m) {
    int merged[n + m];
    int i, j, k = 0;
    // Copy both arrays into merged array
    for (i = 0; i < n; i++) merged[k++] = a[i];
    for (i = 0; i < m; i++) merged[k++] = b[i];
    // Sort the merged array
```

```
 qsort(merged, k, sizeof(int), compare);
    // Print unique elements
    printf("Merged array: %d", merged[0]);
    for (i = 1; i < k; i++) {
        if (merged[i] != merged[i - 1]) {
            printf(",%d", merged[i]);
        }
    }
}
```

```
int main() {
    int arr1[] = {2, 4, 5, 6, 7, 9, 10, 13};
    int arr2[] = {2, 3, 4, 5, 6, 7, 8, 9, 11, 15};
    int n1 = sizeof(arr1) / sizeof(arr1[0]);
    int n2 = sizeof(arr2) / sizeof(arr2[0]);

    mergeAndPrintUnique(arr1, n1, arr2, n2);

    return 0;
}
```

ANSWER IN JAVA

```
import java.util.*;

public class MergeSortedUnique {
    public static void main(String[] args) {
        int[] arr1 = {2, 4, 5, 6, 7, 9, 10, 13};
        int[] arr2 = {2, 3, 4, 5, 6, 7, 8, 9, 11, 15};

        // Use TreeSet to merge and remove duplicates in sorted order
        Set<Integer> result = new TreeSet<>();

        for (int num : arr1)
            result.add(num);
```

```
 for (int num : arr2)
           result.add(num);
System.out.print("Merged array: ");
       for (int num : result)
           System.out.print(num + " ");
   }
}
```

5.Using Recursion reverse the string such as

Eg 1: Input: one two three
Output: three two one
Eg 2: Input: I love india
Output: india love I

ANSWER IN C

```
#include <stdio.h>
#include <string.h>
int main() {
    char str[] = "I love india";
    char *words[50];
    int count = 0;
char *token = strtok(str, " ");
    while (token != NULL) {
        words[count++] = token;
        token = strtok(NULL, " ");   }
printf("Output: "); // Print words in reverse
    for (int i = count - 1; i >= 0; i--) {
        printf("%s", words[i]);
        if (i > 0) printf(" ");
    }  return 0;    }
```

ANSWER IN JAVA

```
public class ReverseWords {
    public static void main(String[] args) {
        String input = "I love india";
        String[] words = input.split(" ");
        System.out.print("Output: ");
        for (int i = words.length - 1; i >= 0; i--) {
            System.out.print(words[i]);
            if (i > 0) System.out.print(" ");
        }
    }
}
```

6. Write a program to print the following output for the given input. You can assume the string is of odd length

Eg 1: Input: 12345

Output:

```
1 5
2 4
3
2 4
1 5
```

ANSWER IN C

```
#include <stdio.h>
#include <string.h>
int main() {
    char s[200];
    if (!fgets(s, sizeof(s), stdin)) return 0;
    int n = strlen(s);
    while (n > 0 && (s[n-1] == '\n' || s[n-1] == '\r')) n--;
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (j == i || j == n - 1 - i)
                putchar(s[i]);
            else
                putchar(' ');
        }
        putchar('\n');
    }
    return 0;
}
```

ANSWER IN JAVA

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
public class XPattern {
    public static void main(String[] args) throws Exception {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String s = br.readLine();
        int n = s.length();
        for (int i = 0; i < n; i++) {
            for (int j = 0; j < n; j++) {
                if (j == i || j == n - 1 - i)
                    System.out.print(s.charAt(i));
                else
                    System.out.print(' ');
            }
            System.out.println();
        } }   }
```

7.Problem:

Given a set of elements as an array find the median of the array. Median is the value which separates the higher indexes from the lower indexes.

E.g.: input =[1, 2, 3] output =2;

input = [1, 2, 3, 4] output =2.

ANSWER IN C

```
#include <stdio.h>
#include <stdlib.h>
int compare(const void *a, const void *b) {
    return (*(int*)a - *(int*)b);
}
double findMedian(int arr[], int n) {
    qsort(arr, n, sizeof(int), compare);
    if (n % 2 == 0)
        return (arr[n/2 - 1] + arr[n/2]) / 2.0;
    else
        return arr[n/2];
}
int main() {
    int arr[] = {1, 2, 3, 4};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf("Median: %.2f\n", findMedian(arr, n));
}
```

ANSWER IN JAVA

```
import java.util.Arrays;
public class MedianFinder {
    public static double findMedian(int[] arr) {
        Arrays.sort(arr);
        int n = arr.length;
        if (n % 2 == 0)
            return (arr[n / 2 - 1] + arr[n / 2]) / 2.0;
        else
            return arr[n / 2];
    }
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4};
        System.out.println("Median: " + findMedian(arr));
    }
}
```

8.Program

Given a set of strings find the first occurrence of a string

E.g.

input =[AL, AL, GH, F, GH, PK]

output =F

ANSWER IN C

```c
#include <stdio.h>
#include <string.h>
struct Pair {
    char str[10];
    int count;
};
int main() {
    char *arr[] = {"AL", "AL", "GH", "F", "GH", "PK"};
    int n = 6;
    struct Pair freq[10];
    int size = 0;
    for (int i = 0; i < n; i++) {
        int found = 0;
        for (int j = 0; j < size; j++) {
            if (strcmp(freq[j].str, arr[i]) == 0) {
                freq[j].count++;
                found = 1;
                break;
            }
        }
```

```
 if (!found) {
            strcpy(freq[size].str, arr[i]);
            freq[size++].count = 1;
        }
    }

    for (int i = 0; i < size; i++) {
        if (freq[i].count == 1) {
            printf("First unique string: %s\n", freq[i].str);
            break;
        }
    }
    return 0;
}
```

ANSWER IN JAVA

```java
import java.util.*;
public class FirstUnique {
    public static String findFirstUnique(String[] arr) {
        Map<String, Integer> map = new LinkedHashMap<>();
        for (String s : arr)
            map.put(s, map.getOrDefault(s, 0) + 1);
        for (Map.Entry<String, Integer> entry : map.entrySet())
            if (entry.getValue() == 1)
                return entry.getKey();
        return null;
    }
    public static void main(String[] args) {
        String[] arr = {"AL", "AL", "GH", "F", "GH", "PK"};
        System.out.println("First unique string: " + findFirstUnique(arr));
    }
}
```

9)Given an array of numbers find a subset from the array such that the average for the whole set of numbers should equal the average of the numbers in the subsets deduced from the main array.

- E.g.:
- input =[10, 20, 30, 40] output =[20, 30] [10, 40]
- Input =[20, 40, 60] output = [40] [20, 60]

ANSWER IN C

```
#include <stdio.h>
void findSubsetPairs(int arr[], int n) {
    int totalSum = 0;
    for (int i = 0; i < n; i++)
        totalSum += arr[i];
    double avg = (double)totalSum / n;
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            double pairAvg = (arr[i] + arr[j]) / 2.0;
            if (pairAvg == avg) {
                printf("[%d, %d]\n", arr[i], arr[j]);
```

```
 }
        }
    }
}

int main() {
    int arr[] = {10, 20, 30, 40};
    int n = sizeof(arr) / sizeof(arr[0]);
    findSubsetPairs(arr, n);
    return 0;
}
```

ANSWER IN JAVA

```
public class SubsetAverage {
    public static void findSubsets(int[] arr) {
        int totalSum = 0;
        for (int n : arr) totalSum += n;
        double average = (double) totalSum / arr.length;
        for (int i = 0; i < arr.length; i++)
            for (int j = i + 1; j < arr.length; j++) {
                int sum = arr[i] + arr[j];
                double subAvg = (double) sum / 2;
                if (subAvg == average)
                    System.out.println("[" + arr[i] + ", " + arr[j] + "]");
            }
    }
```

```
 public static void main(String[] args) {
        int[] arr = {10, 20, 30, 40};
        findSubsets(arr);
    }
}
```

10)Implement a LRU (Least Recently Used) cache of size 10.

There must be a key and value for each element in cache

There must be two functions get (key) and put (key, value)When trying to add after 11th element the least recently accessed element should be replaced.

ANSWER IN C

```
#include <stdio.h>
#include <stdlib.h>
#define CAPACITY 10
#define HASH_SIZE 100
typedef struct Node {
    int key, value;
    struct Node *prev, *next;
} Node;
typedef struct {
    Node *head, *tail;
    Node *hash[HASH_SIZE];
} LRUCache;
int hash(int key) {
    return key % HASH_SIZE;
}
```

```
void removeNode(LRUCache* cache, Node* node) {
   if (node->prev) node->prev->next = node->next;
   else cache->head = node->next;

   if (node->next) node->next->prev = node->prev;
   else cache->tail = node->prev;
}

// Insert node at front
void insertFront(LRUCache* cache, Node* node) {
   node->next = cache->head;
   node->prev = NULL;

   if (cache->head)
      cache->head->prev = node;
   cache->head = node;

   if (!cache->tail)
      cache->tail = node;
}
```

```
LRUCache* createCache() {
    LRUCache* cache = (LRUCache*)malloc(sizeof(LRUCache));
    cache->head = cache->tail = NULL;
    for (int i = 0; i < HASH_SIZE; i++) cache->hash[i] = NULL;
    return cache;
}

void put(LRUCache* cache, int key, int value) {
    int index = hash(key);
    Node* existing = cache->hash[index];

    while (existing && existing->key != key) existing = existing->next;

    if (existing) {
        existing->value = value;
        removeNode(cache, existing);
        insertFront(cache, existing);
        return;
    }
```

```
Node* node = (Node*)malloc(sizeof(Node));
    node->key = key;
    node->value = value;
    node->prev = node->next = NULL;

    insertFront(cache, node);
    node->next = cache->hash[index];
    cache->hash[index] = node;

    // If over capacity, remove LRU node
    int count = 0;
    Node* temp = cache->head;
    while (temp) {
        count++;
        temp = temp->next;
    }
```

```
if (count > CAPACITY) {
        Node* lru = cache->tail;
        removeNode(cache, lru);
        free(lru);
    }
}

int get(LRUCache* cache, int key) {
    int index = hash(key);
    Node* node = cache->hash[index];

    while (node && node->key != key) node = node->next;

    if (node) {
        removeNode(cache, node);
        insertFront(cache, node);
        return node->value;
    }
    return -1;
}
```

```
void printCache(LRUCache* cache) {
    Node* curr = cache->head;
    while (curr) {
        printf("(%d:%d) ", curr->key, curr->value);
        curr = curr->next;
    }
    printf("\n");
}

int main() {
    LRUCache* cache = createCache();
    for (int i = 1; i <= 11; i++) {
        put(cache, i, i * 100);
        printCache(cache);
    }
    printf("Get key 5: %d\n", get(cache, 5));
    printCache(cache);
    return 0;
}
```

ANSWER IN JAVA

```
import java.util.*;
class LRUCache<K, V> extends LinkedHashMap<K, V> {
    private final int capacity;
    public LRUCache(int capacity) {
        super(capacity, 0.75f, true); // true = access-order
        this.capacity = capacity;
    }
    public V get(Object key) {
        return super.getOrDefault(key, null);
    }
    public void put(K key, V value) {
        super.put(key, value);
    }
protected boolean removeEldestEntry(Map.Entry<K, V> eldest) {
        return size() > capacity;
    }

    public static void main(String[] args) {
        LRUCache<Integer, String> cache = new LRUCache<>(10);
        for (int i = 1; i <= 11; i++) {
            cache.put(i, "Val" + i);
        }
        System.out.println(cache);
    }
}
```

11)Given a text and a wildcard pattern, implement wildcard pattern matching algorithm that finds if wildcard pattern is matched with text.

The matching should cover the entire text (not partial text).

The wildcard pattern can include the characters '?' and'?' matches any single character Matches any sequence of characters (including the empty sequence).

ANSWER IN C

```
#include <stdio.h>
#include <string.h>
#include <stdbool.h>
bool isMatch(const char* text, const char* pattern) {
   int tLen = strlen(text);
   int pLen = strlen(pattern);
   bool dp[tLen + 1][pLen + 1];
   memset(dp, false, sizeof(dp));
   dp[0][0] = true;
for (int j = 1; j <= pLen; j++) {
      if (pattern[j - 1] == '*')
         dp[0][j] = dp[0][j - 1];
   }
```

```
 for (int i = 1; i <= tLen; i++) {
        for (int j = 1; j <= pLen; j++) {
            if (pattern[j - 1] == '?' || pattern[j - 1] == text[i - 1])
                dp[i][j] = dp[i - 1][j - 1];
            else if (pattern[j - 1] == '*')
                dp[i][j] = dp[i][j - 1] || dp[i - 1][j];
        }
    }

    return dp[tLen][pLen];
}

int main() {
    const char* text = "abcde";
    const char* pattern = "a*de";
 if (isMatch(text, pattern))
        printf("Matched\n");
    else
        printf("Not Matched\n");

    return 0;
}
```

ANSWER IN JAVA

```
public class WildcardMatching {
    public static boolean isMatch(String text, String pattern) {
        int tLen = text.length();
        int pLen = pattern.length();
        boolean[][] dp = new boolean[tLen + 1][pLen + 1];
        dp[0][0] = true;
for (int j = 1; j <= pLen; j++) {
            if (pattern.charAt(j - 1) == '*')
                dp[0][j] = dp[0][j - 1];
        }
for (int i = 1; i <= tLen; i++) {
        for (int j = 1; j <= pLen; j++) {
            char pc = pattern.charAt(j - 1);
            char tc = text.charAt(i - 1);

            if (pc == '?' || pc == tc) {
                dp[i][j] = dp[i - 1][j - 1];
            } else if (pc == '*') {
                dp[i][j] = dp[i][j - 1] || dp[i - 1][j];
            }
        }
    }

    return dp[tLen][pLen];
  }
```

```
 public static void main(String[] args) {
        String text = "abcde";
        String pattern = "a*de";

        if (isMatch(text, pattern))
            System.out.println("Matched");
        else
            System.out.println("Not Matched");
    }
}
```

12)Given an input string and a dictionary of words,find out if the input string can be segmented into a space-separated sequence of dictionary words. See following examples for more details.

Example: Consider the following dictionary{i, like, sam, sung, samsung, mobile, ice, cream, icecream, man, go,mango}

Input:ilike

Output: Yes

The string can be segmented as "i like".

Input:ilikesamsung

Output:Yes

The string can be segmented as "i like samsung" or "i like sam sung".

ANSWER IN C

```
#include <stdio.h>
#include <stdbool.h>
#include <string.h>
#define DICT_SIZE 13
const char* dict[] = {
    "i", "like", "sam", "sung", "samsung", "mobile", "ice", "cream",
    "icecream", "man", "go", "mango", "ilike"
};
// Check if word exists in dictionary
bool inDict(const char* str) {
    for (int i = 0; i < DICT_SIZE; i++) {
        if (strcmp(dict[i], str) == 0)
            return true;
    }
    return false;
}
```

```
bool wordBreak(char* str) {
    int n = strlen(str);
    bool dp[n + 1];
    memset(dp, 0, sizeof(dp));
    dp[0] = true;

    for (int i = 1; i <= n; i++) {
        for (int j = 0; j < i; j++) {
            char sub[50];
            strncpy(sub, str + j, i - j);
            sub[i - j] = '\0';

            if (dp[j] && inDict(sub)) {
                dp[i] = true;
                break;
            }
        }
    }
```

```
 return dp[n];
}

int main() {
    char input1[] = "ilike";
    char input2[] = "ilikesamsung";

    printf("%s\n", wordBreak(input1) ? "Yes" : "No");
    printf("%s\n", wordBreak(input2) ? "Yes" : "No");
    return 0;
}
```

ANSWER IN JAVA

```
import java.util.*;
public class WordBreak {
    static Set<String> dict = new HashSet<>(Arrays.asList(
        "i", "like", "sam", "sung", "samsung", "mobile", "ice",
        "cream", "icecream", "man", "go", "mango"
    ));
    public static boolean wordBreak(String str) {
        int n = str.length();
        boolean[] dp = new boolean[n + 1];
        dp[0] = true; // empty string

        for (int i = 1; i <= n; i++) {
            for (int j = 0; j < i; j++) {
                if (dp[j] && dict.contains(str.substring(j, i))) {
                    dp[i] = true;
                    break;
```

```
 }
          }
      }

      return dp[n];
   }

   public static void main(String[] args) {
      String input1 = "ilike";
      String input2 = "ilikesamsung";

      System.out.println(wordBreak(input1) ? "Yes" : "No"); // Yes
      System.out.println(wordBreak(input2) ? "Yes" : "No"); // Yes
   }
}
```

13)Print the following pattern

```
            1
        3       2
    6       5       4
10    9       8       7
10    9       8       7
    6       5        4
        3        2
             1
```

ANSWER IN C

```
#include <stdio.h>
int main() {
    int k = 1;
printf("%d\n", k++);
    printf("%d%d\n", k + 1, k++);
    printf("%d %d\n", k + 1, k++);
    printf("%d\n", k++);
    printf("%d %d %d %d\n", k + 3, k + 2, k + 1, k++);
    k += 3;
```

```
 // Second Half
    printf("%d %d %d %d\n", k, k - 1, k - 2, k - 3);
    k -= 4;
    printf("%d %d\n", k--, k--);
    printf("%d\n", k--);
    printf("%d %d\n", k--, k--);
    printf("%d\n", k);

    return 0;
}
```

ANSWER IN JAVA

```
public class PatternPrinter {
    public static void main(String[] args) {
        int k = 1;
System.out.println(k++);
        System.out.println((k + 1) + "" + k++);
        System.out.println((k + 1) + " " + k++);
        System.out.println(k++);
        System.out.print((k + 3) + " ");
        System.out.print((k + 2) + " ");
        System.out.print((k + 1) + " ");
        System.out.println(k++);
        k += 3;
        // Second Half (same as above but in reverse)
        System.out.print(k + " ");
        System.out.print((k - 1) + " ");
        System.out.print((k - 2) + " ");
        System.out.println((k - 3));
        k -= 4;

        System.out.println(k-- + " " + (k--));
        System.out.println(k--);
        System.out.println((k--) + " " + (k--));
        System.out.println(k);
    }
}
```

14)Given an array as input, The condition is if the number is repeated you must add the number and put the next index value to 0. If the number is 0 print it at the last.

Example:

Eg: arr[] = { 0, 2, 2, 2, 0, 6, 6, 0, 8}

Output: 4 2 12 8 0 0 0 0 0.

ANSWER IN C

```
#include <stdio.h>
void processArray(int arr[], int n) {
for (int i = 0; i < n - 1; i++) {
        if (arr[i] != 0 && arr[i] == arr[i + 1]) {
            arr[i] = arr[i] + arr[i + 1];
            arr[i + 1] = 0;
        }
    }
int result[n];
    int index = 0;
    for (int i = 0; i < n; i++) {
        if (arr[i] != 0) {
            result[index++] = arr[i];
```

```
 }
    }

    while (index < n)
        result[index++] = 0;

    // Print result
    for (int i = 0; i < n; i++)
        printf("%d ", result[i]);
}
int main() {
    int arr[] = {0, 2, 2, 2, 0, 6, 6, 0, 8};
    int n = sizeof(arr) / sizeof(arr[0]);
    processArray(arr, n);
    return 0;
}
```

ANSWER IN JAVA

```
import java.util.*;
public class MergeArray {
    public static void processArray(int[] arr) {
        int n = arr.length;
for (int i = 0; i < n - 1; i++) {
            if (arr[i] != 0 && arr[i] == arr[i + 1]) {
                arr[i] = arr[i] + arr[i + 1];
                arr[i + 1] = 0;
            }
        }
int[] result = new int[n];
        int index = 0;
        for (int num : arr) {
            if (num != 0) {
                result[index++] = num;
            }
        }

        // Print result
        for (int val : result)
            System.out.print(val + " ");
    }

    public static void main(String[] args) {
        int[] arr = {0, 2, 2, 2, 0, 6, 6, 0, 8};
        processArray(arr);
    }
}
```

15)String Compression

Input: aaabbbccc

Output: a3b3c3

Count and compress repeating characters.

ANSWER IN C

```
#include <stdio.h>
#include <string.h>
void compressString(const char* str) {
    int count = 1;
    int len = strlen(str);
    for (int i = 1; i <= len; i++) {
        if (str[i] == str[i - 1]) {
            count++;
        } else {
            printf("%c%d", str[i - 1], count);
            count = 1;
        }
    }
    printf("\n");
}
```

```
int main() {
    const char* input = "aaabbbccc";
    compressString(input);  // Output: a3b3c3
    return 0;
}
```

ANSWER IN JAVA

```
public class StringCompressor {
    public static String compress(String input) {
        StringBuilder sb = new StringBuilder();
        int count = 1;

        for (int i = 1; i <= input.length(); i++) {
            if (i < input.length() && input.charAt(i) ==
input.charAt(i - 1)) {
                count++;
            }
else {
            sb.append(input.charAt(i - 1)).append(count);
            count = 1;
         }}
      return sb.toString();
   }
   public static void main(String[] args) {
      String input = "aaabbbccc";
      String result = compress(input);
      System.out.println(result); // Output: a3b3c3
   }}
```

16)Balanced Parentheses Check

Input: {[()]}

Output: Balanced

Use stack to check matching brackets.

ANSWER IN C

```
#include <stdio.h>
#include <string.h>
#define MAX 100
char stack[MAX];
int top = -1;
void push(char c) {
    if (top < MAX - 1)
        stack[++top] = c;
}
char pop() {
    if (top >= 0)
        return stack[top--];
    return '\0';
}
```

```
int isMatching(char open, char close) {
    return (open == '(' && close == ')') ||
           (open == '{' && close == '}') ||
           (open == '[' && close == ']');
}
int isBalanced(const char *expr) {
    for (int i = 0; i < strlen(expr); i++) {
        char ch = expr[i];
        if (ch == '(' || ch == '{' || ch == '[') {
            push(ch);
        } else if (ch == ')' || ch == '}' || ch == ']') {
            char topChar = pop();
            if (!isMatching(topChar, ch)) return 0;
        }
    }
```

```
 return top == -1;
}
int main() {
    const char *input = "{[()]}";
    if (isBalanced(input))
        printf("Balanced\n");
    else
        printf("Not Balanced\n");
    return 0;
}
```

ANSWER IN JAVA

```
import java.util.Stack;
public class BalancedParentheses {
    public static boolean isBalanced(String expr) {
        Stack<Character> stack = new Stack<>();
        for (char ch : expr.toCharArray()) {
            if (ch == '(' || ch == '{' || ch == '[') {
                stack.push(ch);
            } else if (ch == ')' || ch == '}' || ch == ']') {
                if (stack.isEmpty()) return false;
                char top = stack.pop();
                if ((ch == ')' && top != '(') ||
                    (ch == '}' && top != '{') ||
                    (ch == ']' && top != '[')) {
```

```
  return false;
              }

          }
        }
return stack.isEmpty();
    }

    public static void main(String[] args) {
        String input = "{[()]}";
        System.out.println(isBalanced(input) ? "Balanced" : "Not Balanced");
    }
}
```

17)Find Missing Number in Array

Input: {1, 2, 3, 5}

Output: 4

Use XOR or sum formula.

ANSWER IN C

METHOD USING SUM FORMULA

```c
#include <stdio.h>
int findMissing(int arr[], int n) {
    int expectedSum = n * (n + 1) / 2;
    int actualSum = 0;
    for (int i = 0; i < n - 1; i++)
        actualSum += arr[i];
    return expectedSum - actualSum;
}
int main() {
    int arr[] = {1, 2, 3, 5};
    int n = 5;
    printf("Missing number: %d\n", findMissing(arr, n));
    return 0;
}
```

ANSWER IN C

METHOD USING XOR

```
#include <stdio.h>
int findMissing(int arr[], int n) {
    int xor1 = 0, xor2 = 0;
    for (int i = 1; i <= n; i++)
        xor1 ^= i;
    for (int i = 0; i < n - 1; i++)
        xor2 ^= arr[i];
    return xor1 ^ xor2;
}
int main() {
    int arr[] = {1, 2, 3, 5};
    int n = 5;
    printf("Missing number: %d\n", findMissing(arr, n));
    return 0;
}
```

ANSWER IN JAVA

METHOD USING SUM FORMULA

```
public class MissingNumber {
    public static int findMissing(int[] arr, int n) {
        int expectedSum = n * (n + 1) / 2;
        int actualSum = 0;
        for (int num : arr)
            actualSum += num;
        return expectedSum - actualSum;
    }
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 5};
        int n = 5;
        System.out.println("Missing number: " + findMissing(arr, n));
    }
}
```

ANSWER IN JAVA

METHOD USING XOR

```
public class MissingNumberXOR {
    public static int findMissing(int[] arr, int n) {
        int xor1 = 0, xor2 = 0;
        for (int i = 1; i <= n; i++)
            xor1 ^= i;
        for (int num : arr)
            xor2 ^= num;
        return xor1 ^ xor2;
    }
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 5};
        int n = 5;
        System.out.println("Missing number: " + findMissing(arr, n));
    }
}
```

18)Sort String in Alphabetical Order Without Inbuilt Sort

Input: cabd

Output: abcd

Use bubble or selection sort.

ANSWER IN C

USING SELECTION SORT

```
#include <stdio.h>
#include <string.h>
void sortString(char str[]) {
    int n = strlen(str);
    for (int i = 0; i < n - 1; i++) {
        int min = i;
        for (int j = i + 1; j < n; j++) {
            if (str[j] < str[min])
                min = j;
        }
char temp = str[i];
        str[i] = str[min];
        str[min] = temp;
    }
```

```
 printf("Sorted string: %s\n", str);
}
int main() {
   char str[] = "cabd";
   sortString(str);
   return 0;
}
```

ANSWER IN JAVA

USING BUBBLE SORT

```
public class StringSorter {
    public static String sortString(String str) {
        char[] arr = str.toCharArray();
        int n = arr.length;
for (int i = 0; i < n - 1; i++) {
            for (int j = 0; j < n - 1 - i; j++) {
                if (arr[j] > arr[j + 1]) {
                    // Swap
                    char temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
```

```
 }
            }
        }

        return new String(arr);
    }

    public static void main(String[] args) {
        String input = "cabd";
        System.out.println("Sorted string: " + sortString(input));
    }
}
```

19)Alternate sorting:

Given an array of integers, rearrange the array in such a way that the first element is first maximum and second element is first minimum.

Eg.)

Input{1, 2, 3, 4, 5, 6, 7}

Output: {7, 1, 6, 2, 5, 3, 4}

ANSWER IN C

```
#include <stdio.h>
#include <stdlib.h>
int compare(const void* a, const void* b) {
    return (*(int*)a - *(int*)b);  }
void alternateSort(int arr[], int n) {
    qsort(arr, n, sizeof(int), compare); // sort ascending
    int result[n];
    int start = 0, end = n - 1;
    for (int i = 0; i < n; i++) {
        if (i % 2 == 0)
            result[i] = arr[end--];
        else
            result[i] = arr[start++];     }
```

```
 for (int i = 0; i < n; i++)
        printf("%d ", result[i]);
}
int main() {
    int arr[] = {1, 2, 3, 4, 5, 6, 7};
    int n = sizeof(arr) / sizeof(arr[0]);
    alternateSort(arr, n);
    return 0;
}
```

ANSWER IN JAVA

```
import java.util.Arrays;

public class AlternateSort {
    public static void alternateSort(int[] arr) {
        Arrays.sort(arr); // Sort in ascending
        int n = arr.length;
        int[] result = new int[n];

        int start = 0, end = n - 1;
        for (int i = 0; i < n; i++) {
            result[i] = (i % 2 == 0) ? arr[end--] : arr[start++];
        }
```

```
 // Print result
        for (int num : result)
            System.out.print(num + " ");
    }

    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5, 6, 7};
        alternateSort(arr);
    }
}
```

20)Remove unbalanced parentheses in a given expression.

Eg.)

Input :((abc)((de))

Output:((abc) (de))

Input:(((ab)

Output:(ab)

ANSWER IN C

```
#include <stdio.h>
#include <string.h>
void removeUnbalanced(char* str) {
    int n = strlen(str);
    char temp[n + 1];
    int open = 0, index = 0;
    // First pass: remove unbalanced ')'
    for (int i = 0; i < n; i++) {
        if (str[i] == '(') {
            open++;
            temp[index++] = str[i];
        } else if (str[i] == ')') {
            if (open > 0) {
                open--;
                temp[index++] = str[i];
            }
        } else {
```

```
temp[index++] = str[i];
      }
   }
   temp[index] = '\0';
   // Second pass: remove extra '(' from end
   char result[n + 1];
   int rIndex = 0;
   open = 0;
   for (int i = index - 1; i >= 0; i--) {
       if (temp[i] == ')') {
           open++;
           result[rIndex++] = temp[i];
       } else if (temp[i] == '(') {
           if (open > 0) {
               open--;
               result[rIndex++] = temp[i];
           }
       } else {
```

```
 result[rIndex++] = temp[i];
        }
    }
    // Reverse and print the result
    for (int i = rIndex - 1; i >= 0; i--) {
        putchar(result[i]);
    }
    printf("\n");
}
int main() {
    char input1[] = "((abc)((de))";
    char input2[] = "(((ab)";
    removeUnbalanced(input1);  // Output: ((abc)(de))
    removeUnbalanced(input2);  // Output: (ab)
    return 0;
}
```

ANSWER IN JAVA

```
public class BalancedParenthesesCleaner {
    public static String removeUnbalanced(String str) {
        StringBuilder sb = new StringBuilder();
        int open = 0;
        // First pass: remove unbalanced closing ')'
        for (char ch : str.toCharArray()) {
            if (ch == '(') {
                open++;
                sb.append(ch);
            } else if (ch == ')') {
                if (open > 0) {
                    open--;
                    sb.append(ch);
                }
            }else {
```

```
sb.append(ch);
            }
        }
// Second pass: remove unbalanced opening '(' from end
        StringBuilder result = new StringBuilder();
        open = 0;
        for (int i = sb.length() - 1; i >= 0; i--) {
            char ch = sb.charAt(i);
            if (ch == ')') {
                open++;
                result.append(ch);
            } else if (ch == '(') {
                if (open > 0) {
                    open--;
                    result.append(ch);
                }
            }
```

```
else {
            result.append(ch);
         }
      }
return result.reverse().toString();
  }

  public static void main(String[] args) {
      String input1 = "((abc)((de))";
      String input2 = "(((ab)";

      System.out.println(removeUnbalanced(input1)); // Output: ((abc)(de))
      System.out.println(removeUnbalanced(input2)); // Output: (ab)
  }
}
```

21)Form a number system with only 3 and 4. Find the nth number of the numbersystem.

Eg :
The numbers are:

3, 4, 33, 34, 43, 44, 333, 334, 343, 344, 433, 434, 443, 444, 3333, 3334, 3343, 3344, 3433, 3434, 3443, 3444

ANSWER IN C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX 10000
// Simple queue for strings
char* queue[MAX];
int front = 0, rear = 0;
void enqueue(char* str) {
    queue[rear++] = strdup(str);
}
char* dequeue() {
    return queue[front++];
}
char* findNthNumber(int n) {
```

```
enqueue("3");
   enqueue("4");
   int count = 0;
   while (front < rear) {
       char* curr = dequeue();
       count++;
       if (count == n)
           return curr;
       // Allocate and enqueue new strings
       char* next1 = malloc(strlen(curr) + 2);
       strcpy(next1, curr);
       strcat(next1, "3");
       enqueue(next1);
       char* next2 = malloc(strlen(curr) + 2);
```

```
 strcpy(next2, curr);
        strcat(next2, "4");
        enqueue(next2);
    }

    return "";
}

int main() {
    int n = 15;
    char* result = findNthNumber(n);
    printf("The %dth number is: %s\n", n, result);
    return 0;
}
```

ANSWER IN JAVA

```
import java.util.*;
public class ThreeFourNumberSystem {
    public static String findNthNumber(int n) {
        Queue<String> queue = new LinkedList<>();
        queue.add("3");
        queue.add("4");
               int count = 0;
        while (!queue.isEmpty()) {
            String current = queue.poll();
            count++;
            if (count == n)
                return current;
queue.add(current + "3");
          queue.add(current + "4");

 }
      return "";
   }
   public static void main(String[] args) {
      int n = 15;
      System.out.println("The " + n + "th number is: " + findNthNumber(n));
   }
}
```

22)Check whether a given mathematical expression is valid

Eg.)

Input :(a+b)(a*b)

Output:Valid

Input:(ab)(ab+)

Output:

ANSWER IN C

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

int isOperator(char ch) {
    return (ch == '+' || ch == '-' || ch == '*' || ch == '/');
}

int isValid(char *expr) {
    int openParen = 0;
    char prev = 0;
    int len = strlen(expr);

    for (int i = 0; i < len; i++) {
        char ch = expr[i];
```

```
// Parentheses check
    if (ch == '(') openParen++;
    else if (ch == ')') {
        if (openParen == 0) return 0;
        openParen--;
    }
    // Operator check
    if (isOperator(ch)) {
        if (i == 0 || i == len - 1) return 0;
        if (isOperator(prev)) return 0;
    }
    if (ch != ' ') prev = ch;
```

```
 }

   return openParen == 0;
}
int main() {
   char expr1[] = "(a+b)(a*b)";
   char expr2[] = "(ab)(ab+)";

   printf("%s\n", isValid(expr1) ? "Valid" : "Not Valid");
   printf("%s\n", isValid(expr2) ? "Valid" : "Not Valid");

   return 0;
}
```

ANSWER IN JAVA

```
public class ExpressionValidator {
    public static boolean isValid(String expr) {
        int openParen = 0;
        char prev = 0;
        for (int i = 0; i < expr.length(); i++) {
            char ch = expr.charAt(i);
if (ch == '(') openParen++;
            else if (ch == ')') {
                if (openParen == 0) return false;
                openParen--;
            }
// Check operator placement
         if (ch == '+' || ch == '-' || ch == '*' || ch == '/') {
            if (i == 0 || i == expr.length() - 1) return false; // start or end
            if (prev == '+' || prev == '-' || prev == '*' || prev == '/') return false; // consecutive
         }

         // Check invalid characters (optional)
         // if (!Character.isLetterOrDigit(ch) && "+-*/()".indexOf(ch) == -1)
         //     return false;

         if (ch != ' ') prev = ch;
      }

      return openParen == 0;
   }
```

```
 public static void main(String[] args) {
        String input1 = "(a+b)(a*b)";
        String input2 = "(ab)(ab+)";

        System.out.println(isValid(input1) ? "Valid" : "Not Valid");
        System.out.println(isValid(input2) ? "Valid" : "Not Valid");
    }
}
```

23)Given a set of numbers like <10, 36, 54,89,12> we want to find sum of weights based on the following conditions

1. 5 if a perfect square
2. 4 if multiple of 4 and divisible by 6
3. 3 if even number And sort the numbers based on the weight and print it as follows

ANSWER IN C

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
typedef struct {
    int number;
    int weight;
} NumberWeight;
int isPerfectSquare(int n) {
    int root = sqrt(n);
    return root * root == n;
}
int getWeight(int n) {
 int w = 0;
   if (isPerfectSquare(n)) w += 5
if (n % 12 == 0) w += 4;
   if (n % 2 == 0) w += 3;
   return w;
}
int compare(const void* a, const void* b) {
   NumberWeight* x = (NumberWeight*)a;
   NumberWeight* y = (NumberWeight*)b;
   if (y->weight != x->weight)
      return y->weight - x->weight;
   return x->number - y->number;
}
int main() {
   int arr[] = {10, 36, 54, 89, 12};
int n = sizeof(arr) / sizeof(arr[0]);
    NumberWeight nw[n];

    for (int i = 0; i < n; i++) {
        nw[i].number = arr[i];
        nw[i].weight = getWeight(arr[i]);
    }

    qsort(nw, n, sizeof(NumberWeight), compare);

    for (int i = 0; i < n; i++) {
        printf("%d %d\n", nw[i].number, nw[i].weight);
    }

    return 0;
}
```

ANSWER IN JAVA

```
import java.util.*;
class NumberWeight {
    int number, weight;
    NumberWeight(int number, int weight) {
        this.number = number;
        this.weight = weight;
    }
}
public class WeightCalculator {
    static boolean isPerfectSquare(int n) {
        int sqrt = (int) Math.sqrt(n);
        return sqrt * sqrt == n;
    }
```

```
static int getWeight(int n) {
        int weight = 0;
        if (isPerfectSquare(n)) weight += 5;
        if (n % 12 == 0) weight += 4;
        if (n % 2 == 0) weight += 3;
        return weight;
    }
    public static void main(String[] args) {
        int[] arr = {10, 36, 54, 89, 12};
        List<NumberWeight> list = new ArrayList<>();
        for (int num : arr) {
            int weight = getWeight(num);
            list.add(new NumberWeight(num, weight));
        }
```

```
 // Sort by weight descending, then number ascending
        list.sort((a, b) -> {
            if (b.weight != a.weight)
                return b.weight - a.weight;
            return a.number - b.number;
        });

        for (NumberWeight nw : list) {
            System.out.println(nw.number + " " + nw.weight);
        }
    }
}
```

24)Using Re ing such as
Eg 1: Input: one two three
Output: three two one
Eg 2: Input: I love india
Output: india love I
Print the given pattern:
Input:
N = 3 M = 3
Output:

```
XXX
XXX
```

Input:

N = 4 M = 5

Output:

```
X 0 0 X
X 0 0 X
X 0 0 X
```

Input:

N = 6 M = 7

```
XΘ X X ΘX
XΘ X X ΘX
XΘ X X ΘX
```

ANSWER IN C

```
#include <stdio.h>
void printPattern(int N, int M) {
    for (int i = 0; i < N; i++) {
        for (int j = 0; j < M; j++) {
            if (j == 0 || j == M - 1)
                printf("X");
            else
                printf("0");  }        printf("\n");   }}
int main() {
    printPattern(3, 3);
    printf("\n");
    printPattern(4, 4);
    return 0;    }
```

ANSWER IN JAVA

```
public class PatternPrinter {
    public static void printPattern(int N, int M) {
        for (int i = 0; i < N; i++) {
            for (int j = 0; j < M; j++) {
                if (j == 0 || j == M - 1)
                    System.out.print("X");
                else
                    System.out.print("0");
            }
            System.out.println();
        }
    }
public static void main(String[] args) {
        printPattern(3, 3); // XXX pattern
        System.out.println();
        printPattern(4, 4); // XOOX pattern
    }
}
```

25)To find the number of groups and output the groups:

Explanation: To find the sum of the elements in the groups and that sum should be divisible by input X and the groups should be limited to range with X numbers. If X is 3, then the group should have only 2 elements and 3 elements from the array whose sum is divisible by 3.

Input:

Array: 3, 9, 7, 4, 6, 8

X: 3

Output:

3,9

3,6

9,6

3, 9, 6

ANSWER IN C

```c
#include <stdio.h>
int main(){
    int arr[] = {3,9,7,4,6,8};
    int n = sizeof(arr)/sizeof(arr[0]);
    int X = 3, count = 0;
    printf("Pairs:\n");
    for(int i=0; i<n; i++){
        for(int j=i+1; j<n; j++){
            if((arr[i] + arr[j]) % X == 0){
                printf("%d, %d\n", arr[i], arr[j]);
                count++;
            }
        }
    }
```

```
 printf("\nTriplets:\n");
    for(int i=0; i<n; i++){
        for(int j=i+1; j<n; j++){
            for(int k=j+1; k<n; k++){
                if((arr[i] + arr[j] + arr[k]) % X == 0){
                    printf("%d, %d, %d\n", arr[i], arr[j], arr[k]);
                    count++;
                }
            }
        }
    }

    printf("\nNo of groups: %d\n", count);
    return 0;
}
```

ANSWER IN JAVA

```
public class DivisibleGroups {
    public static void main(String[] args) {
        int[] arr = {3, 9, 7, 4, 6, 8};
        int X = 3, count = 0;
        System.out.println("Pairs:");
        for(int i = 0; i < arr.length; i++){
            for(int j = i + 1; j < arr.length; j++){
                if ((arr[i] + arr[j]) % X == 0) {
                    System.out.println(arr[i] + ", " + arr[j]);
                    count++;
                }
            }
        }
```

```
 System.out.println("\nTriplets:");
        for(int i = 0; i < arr.length; i++){
            for(int j = i + 1; j < arr.length; j++){
                for(int k = j + 1; k < arr.length; k++){
                    if ((arr[i] + arr[j] + arr[k]) % X == 0) {
                        System.out.println(arr[i] + ", " + arr[j] + ", " + arr[k]);
                        count++;
                    }
                }
            }
        }

        System.out.println("\nNo of groups: " + count);
    }
}
```

26)To output the given string for the given input which is an integer.

Input: 1

Output: A

Input: 26

Output: Z

Input: 27

Output: AA

Input: 28:

Output: AB

Input: 1000

Output: ALL

ANSWER IN C

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
char* numToExcel(int n) {
    if (n <= 0) return strdup("");
    char buf[16] = {0};
    int len = 0;
    while (n > 0) {
        n--;  // adjust
        buf[len++] = 'A' + (n % 26);
        n /= 26;
    }
    buf[len] = '\0';
```

```
 for(int i = 0; i < len/2; i++) {
        char t = buf[i];
        buf[i] = buf[len - 1 - i];
        buf[len - 1 - i] = t;
    }
    return strdup(buf);
}

int main() {
    int tests[] = {1, 26, 27, 28, 1000};
    for (int i = 0; i < 5; i++) {
        char *s = numToExcel(tests[i]);
        printf("%d -> %s\n", tests[i], s);
        free(s);
    }
    return 0;
}
```

ANSWER IN JAVA

```
public class ExcelTitle {
    public static String numToExcel(int n) {
        if (n <= 0) return "";
        StringBuilder sb = new StringBuilder();
        while (n > 0) {
            n--;
            sb.append((char)('A' + (n % 26)));
            n /= 26;
        }
        return sb.reverse().toString();
    }
```

```
  public static void main(String[] args) {
      int[] tests = {1, 26, 27, 28, 1000};
      for (int n : tests) {
          System.out.printf("%d -> %s%n", n, numToExcel(n));
      }
  }
}
```

27)Input:

Number of elements in set1:4

Elements are: 9, 9, 9, 9

Number of elements in set 2:3

Elements are: 1,1,1

Output:1, 0, 1, 1, 0

Input:

Number of elements in set1: 11

Elements are: 7,2,3,4,5,3,1,2,7,2,8

Number of elements in set 2:3

Elements are: 1,2,3

Output: 7,2,3,4,5,3,1,2,8,5,1

ANSWER IN C

```
#include <stdio.h>
#include <stdlib.h>
void removeFirst(int arr[], int *n, int x) {
    for (int i = 0; i < *n; i++) {
        if (arr[i] == x) {
            for (int j = i; j + 1 < *n; j++)
                arr[j] = arr[j + 1];
            (*n)--;
            return;
        }
    }
}
int main() {
 int arr[] = {7,2,3,4,5,3,1,2,7,2,8};
   int n = sizeof(arr)/sizeof(arr[0]);
   int set2[] = {1,2,3}, m = 3;

   for (int i = 0; i < m; i++) {
       removeFirst(arr, &n, set2[i]);
   }

   for (int i = 0; i < n; i++) {
       printf("%d%s", arr[i], i+1<n ? ", " : "");
   }
   return 0;
}
```

ANSWER IN JAVA

```
import java.util.*;
public class RemoveFirstOccurrences {
    public static void main(String[] args) {
        List<Integer> list1 = new ArrayList<>(Arrays.asList(7,2,3,4,5,3,1,2,7,2,8));
        List<Integer> list2 = Arrays.asList(1,2,3);
        for (int x : list2) {
            list1.remove((Integer)x); // removes first occurrence
        }
        System.out.println(String.join(", ",
            list1.stream().map(String::valueOf).toArray(String[]::new)));
    }
}
```

28)Input:

Number of elements in set1:4

Elements are: 9, 9, 9, 9

Number of elements in set 2:3

Elements are: 1,1,1

Output:1, 0, 1, 1, 0

Input:

Number of elements in set1: 11

Elements are: 7,2,3,4,5,3,1,2,7,2,8

Number of elements in set 2:3

Elements are: 1,2,3

Output: 7,2,3,4,5,3,1,2,8,5,1

ANSWER IN C

```
#include <stdio.h>
#include <stdlib.h>
void removeFirst(int arr[], int *n, int x) {
    for (int i = 0; i < *n; i++) {
        if (arr[i] == x) {
            for (int j = i; j + 1 < *n; j++)
                arr[j] = arr[j + 1];
            (*n)--;
            return;
        }
    }
}
int main() {
 int arr[] = {7,2,3,4,5,3,1,2,7,2,8};
   int n = sizeof(arr)/sizeof(arr[0]);
   int set2[] = {1,2,3}, m = 3;

   for (int i = 0; i < m; i++) {
       removeFirst(arr, &n, set2[i]);
   }

   for (int i = 0; i < n; i++) {
       printf("%d%s", arr[i], i+1<n ? ", " : "");
   }
   return 0;
}
```

ANSWER IN JAVA

```java
import java.util.*;
public class RemoveFirstOccurrences {
    public static void main(String[] args) {
        List<Integer> list1 = new ArrayList<>(Arrays.asList(7,2,3,4,5,3,1,2,7,2,8));
        List<Integer> list2 = Arrays.asList(1,2,3);
        for (int x : list2) {
            list1.remove((Integer)x); // removes first occurrence
        }
        System.out.println(String.join(", ",
            list1.stream().map(String::valueOf).toArray(String[]::new)));
    }
}
```

29)Help john to find new friends in social network

Input:

```
3
Mani 3 ram raj guna
Ram 2 kumar Kishore
Mughil 3 praveen Naveen Ramesh
```

Output:

```
Raj guna kumar Kishore praveen Naveen Ramesh
```

ANSWER IN C

```
#include <stdio.h>
#include <string.h>
int main() {
    int n, f;
    char name[100], friend[100], john[] = "john";
    scanf("%d", &n);
    char output[1000] = "";
    for(int i = 0; i < n; i++) {
        scanf("%s %d", name, &f);
        int flag = strcmp(name, "john");
        for(int j = 0; j < f; j++) {
```

```
 scanf("%s", friend);
          if(flag != 0) {
              strcat(output, friend);
              strcat(output, " ");
          }
      }
   }
   printf("%s", output);
   return 0;
}
```

ANSWER IN JAVA

```
import java.util.Scanner;
public class FindFriends {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        String res = "";
        for(int i = 0; i < n; i++) {
            String name = sc.next();
            int f = sc.nextInt();
            for(int j = 0; j < f; j++) {
```

```
  String friend = sc.next();
              if(!name.equals("john"))
                  res += friend + " ";
            }
        }
        System.out.println(res.trim());
    }
}
```

30)Find the union intersection of two list and also find except (remove even elements from list1 and odd elements from list2)

Input

List 1: 1,3,4,5,6,8,9

List 2: 1, 5,8,9,2

Union: 1, 3,4,5,6,8,9,2

Intersection: 1,5,8,9

Except: 1, 3, 5,9,8,2

ANSWER IN C

```
#include <stdio.h>
int main() {
    int n1, n2, i, j, flag;
    int a[100], b[100], unionArr[200], inter[100], u=0, in=0;
    scanf("%d", &n1);
    for(i=0;i<n1;i++) scanf("%d",&a[i]);
    scanf("%d", &n2);
    for(i=0;i<n2;i++) scanf("%d",&b[i]);
    for(i=0;i<n1;i++) unionArr[u++]=a[i];
    for(i=0;i<n2;i++) {
        flag=0;
for(j=0;j<n1;j++)
            if(b[i]==a[j]) {flag=1;break;}
        if(flag==0) unionArr[u++]=b[i];
    }

    for(i=0;i<n1;i++)
        for(j=0;j<n2;j++)
            if(a[i]==b[j]) {inter[in++]=a[i];break;}

    for(i=0;i<n1;i++)
        if(a[i]%2!=0)
            printf("%d ",a[i]);
```

```
 for(i=0;i<n2;i++)
        if(b[i]%2==0)
            printf("%d ",b[i]);
   printf("\n");

   for(i=0;i<u;i++) printf("%d ",unionArr[i]);
   printf("\n");
   for(i=0;i<in;i++) printf("%d ",inter[i]);

   return 0;
}
```

ANSWER IN JAVA

```
import java.util.*;
public class UnionIntersectionExcept {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n1 = sc.nextInt();
        int[] a = new int[n1];
        for(int i=0;i<n1;i++) a[i]=sc.nextInt();
        int n2 = sc.nextInt();
        int[] b = new int[n2];
        for(int i=0;i<n2;i++) b[i]=sc.nextInt();
        Set<Integer> union = new LinkedHashSet<>();
        for(int i : a) union.add(i);
        for(int i : b) union.add(i);{
```

```
Set<Integer> inter = new LinkedHashSet<>();
for(int i : a)
    for(int j : b)
        if(i==j) {inter.add(i); break;}

for(int i : a)
    if(i%2!=0)
        System.out.print(i+" ");
for(int i : b)
    if(i%2==0)
        System.out.print(i+" ");
System.out.println();
```

```
        for(int i : union) System.out.print(i+" ");
        System.out.println();
        for(int i : inter) System.out.print(i+" ");
    }
}
```

31)Rotate elements **clockwise by one position layer-wise**.

Sample Input

1 2 3 4 5 6 7 8 9

Sample Output

4 1 2 7 5 3 8 9 6

ANSWER IN C

```
#include <stdio.h>

int main() {
    int a[3][3], i, j, temp;
    for(i=0;i<3;i++)
        for(j=0;j<3;j++)
            scanf("%d",&a[i][j]);

    temp = a[0][0];
    a[0][0]=a[1][0];
    a[1][0]=a[2][0];
    a[2][0]=a[2][1];
    a[2][1]=a[2][2];
```

```
 a[2][2]=a[1][2];
    a[1][2]=a[0][2];
    a[0][2]=a[0][1];
    a[0][1]=temp;

    for(i=0;i<3;i++) {
        for(j=0;j<3;j++)
            printf("%d ",a[i][j]);
        printf("\n");
    }
    return 0;
}
```

ANSWER IN JAVA

```
import java.util.*;
public class RotateMatrixElements {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int[][] a = new int[3][3];
        for(int i=0;i<3;i++)
            for(int j=0;j<3;j++)
                a[i][j]=sc.nextInt();
        int temp = a[0][0];
        a[0][0]=a[1][0];
        a[1][0]=a[2][0];
        a[2][0]=a[2][1];
        a[2][1]=a[2][2];
```

```
 a[2][2]=a[1][2];
       a[1][2]=a[0][2];
       a[0][2]=a[0][1];
       a[0][1]=temp;

       for(int i=0;i<3;i++) {
           for(int j=0;j<3;j++)
               System.out.print(a[i][j]+" ");
           System.out.println();
       }
   }
}
```

31)Find the largest possible prime number with given no

Input:

5

4691

Output:9461

ANSWER IN C

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

void swap(char *x, char *y) {
    char temp=*x; *x=*y; *y=temp;
}

bool isPrime(int n) {
    if(n<2) return false;
    for(int i=2;i*i<=n;i++)
        if(n%i==0) return false;
    return true;
}
```

```
void permute(char *a, int l, int r, int *max) {
    if(l==r) {
        int num=atoi(a);
        if(isPrime(num) && num>*max) *max=num;
    } else {
        for(int i=l;i<=r;i++) {
            swap(&a[l],&a[i]);
            permute(a,l+1,r,max);
            swap(&a[l],&a[i]);
        }
    }
}
```

```
int main() {
    char s[20];
    int max=-1;
    scanf("%s",s);
    int n=strlen(s);
    permute(s,0,n-1,&max);
    printf("%d\n",max);
    return 0;
}
```

ANSWER IN JAVA

```
import java.util.*;

public class LargestPrimePermutation {
    static int max = -1;

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.next();
        permute(s.toCharArray(), 0);
        System.out.println(max);
    }
```

```
static void permute(char[] arr, int l) {
    if(l == arr.length) {
        int num = Integer.parseInt(new String(arr));
        if(isPrime(num) && num > max)
            max = num;
    } else {
        for(int i=l;i<arr.length;i++) {
            swap(arr,l,i);
            permute(arr,l+1);
            swap(arr,l,i);
        }
    }
}
```

```
 static void swap(char[] arr, int i, int j) {
        char temp=arr[i]; arr[i]=arr[j]; arr[j]=temp;
    }

    static boolean isPrime(int n) {
        if(n<2) return false;
        for(int i=2;i*i<=n;i++)
            if(n%i==0) return false;
        return true;
    }
}
```

32)For one batch of people Basic programs like pattern printing

```
1
22
333
4444
```

ANSWER IN C

```
 #include <stdio.h>

int main() {
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++) {
        for(int j=1;j<=i;j++)
            printf("%d",i);
        printf("\n");
    }
    return 0;
}
```

ANSWER IN JAVA

```
 import java.util.*;
public class Pattern {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        for(int i=1;i<=n;i++) {
            for(int j=1;j<=i;j++)
                System.out.print(i);
            System.out.println();
        }
    }
}
```

33)For one batch of people Basic programs like pattern printing

```
1
24
357
```

ANSWER IN C

```
 #include <stdio.h>
int main() {
    int n;
    scanf("%d",&n);
    for(int i=1;i<=n;i++) {
        int num=i;
        for(int j=1;j<=i;j++) {
            printf("%d",num);
            num+=2;
        }
        printf("\n");
    }
    return 0;
}
```

ANSWER IN JAVA

```
import java.util.*;
public class Pattern {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        for(int i=1;i<=n;i++) {
            int num=i;
            for(int j=1;j<=i;j++) {
                System.out.print(num);
                num+=2;
            }
            System.out.println();
        }      }    }
```

34)Given a string of integers find out all the possible words that can made out of it in continuous order.

Eg: 11112

ans:

AAAAB

AKAB

AAKB

AAAL etc.

ANSWER IN C

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

void print(char *input, int i, char *output, int j) {
    if (input[i] == '\0') {
        output[j] = '\0';
        printf("%s\n", output);
        return;
    }
    int num = input[i] - '0';
    if (num > 0) {
        output[j] = 'A' + num - 1;
        print(input, i + 1, output, j + 1);
    }
 if (input[i + 1] != '\0') {
       num = (input[i] - '0') * 10 + (input[i + 1] - '0');
       if (num >= 10 && num <= 26) {
           output[j] = 'A' + num - 1;
           print(input, i + 2, output, j + 1);
       }
   }
}

int main() {
   char input[100], output[100];
   scanf("%s", input);
   print(input, 0, output, 0);
   return 0;
}
```

ANSWER IN JAVA

```
import java.util.*;
public class Decode {
    static void print(String input, int i, String output) {
        if (i == input.length()) {
            System.out.println(output);
            return;
        }

        int num = input.charAt(i) - '0';
        if (num > 0) {
            print(input, i + 1, output + (char)('A' + num - 1));
        }
```

```
       if (i + 1 < input.length()) {
            num = Integer.parseInt(input.substring(i, i + 2));
            if (num >= 10 && num <= 26) {
                print(input, i + 2, output + (char)('A' + num - 1));
            }
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String input = sc.next();
        print(input, 0, "");
    }
}
```

35)Find whether a given number is magic number or not. It is something which gives same digits even after cubing it.

ANSWER IN C

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

int main() {
    int n;
    scanf("%d", &n);
    long long cube = (long long)n * n * n;
    int temp = n, mod = 1;
    while (temp > 0) {
        mod *= 10;
        temp /= 10;
    }
```

```
  if (cube % mod == n)
        printf("Magic Number\n");
    else
        printf("Not Magic Number\n");
    return 0;
}
```

ANSWER IN JAVA

```
 import java.util.*;

public class MagicNumber {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        long cube = (long)n * n * n;
        int temp = n, mod = 1;
        while (temp > 0) {
            mod *= 10;
            temp /= 10;
        }
```

```
  if (cube % mod == n)
            System.out.println("Magic Number");
        else
            System.out.println("Not Magic Number");
    }
}
```

36)Print the numbers which are mismatched from two array.

Arr1 {abcdefghi}

arr2 ={ abdeeggi i},

O/P cd, de, f, g, h, i.

ANSWER IN C

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

int main() {
    char arr1[100], arr2[100];
    scanf("%s", arr1);
    scanf("%s", arr2);
    int len1 = strlen(arr1);
    int len2 = strlen(arr2);
    int len = len1 > len2 ? len1 : len2;
    for(int i = 0; i < len; i++) {
        if(arr1[i] != arr2[i]) {
            if(i < len1) printf("%c ", arr1[i]);
            if(i < len2) printf("%c ", arr2[i]);
        }
    }
    return 0;
}
```

ANSWER IN JAVA

```
import java.util.*;

public class MismatchCharacters {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String arr1 = sc.next();
        String arr2 = sc.next();
        int len = Math.max(arr1.length(), arr2.length());

        for(int i = 0; i < len; i++) {
```

```
 char c1 = i < arr1.length() ? arr1.charAt(i) : ' ';
            char c2 = i < arr2.length() ? arr2.charAt(i) : ' ';
            if(c1 != c2) {
                if(c1 != ' ') System.out.print(c1 + " ");
                if(c2 != ' ') System.out.print(c2 + " ");
            }
        }
    }
}
```

37)Print all possible combinations from the given string.