

TCS APRIL-MAY 2024 QUESTION LIST

{SOLUTION In C++/JAVA}

Date- 26 April Shift-1

Q-Find Sub-Arrays with a given sum in array. Given integer array find subarray with a given sum in it.

I/P- arr = [3,4,-7,1,3,3,1,-4] target= 7

O/P- [3,4] , [3,4,-7,1,3,3] , [1,3,3] , [3,3,1]

Approach-

Brute Force Approach is simple , we directly use two for loops to find sub Array and third loop to print the required Array.

In Optimized approach uses a hash map to efficiently find subarrays that sum to a target value. It maintains a cumulative sum (curSum) as it iterates through the array. For each element, it checks if curSum equals the target or if curSum - target is present in the hash map. If so, it prints the subarray that sums to the target. The hash map stores the cumulative sum and its corresponding index, allowing for quick lookups and subarray identification.

Brute Force-

```
4 void findSubarraysWithTargetSum(vector<int>& nums, int size, int targetSum) {
5     for (int start = 0; start < size; start++) {
6         int currentSum = 0;
7         for (int end = start; end < size; end++) {
8             currentSum += nums[end];
9             if (currentSum == targetSum) {
10                 for (int k = start; k <= end; k++) {
11                     cout << nums[k] << " ";
12                 }
13                 cout << endl;
14             }
15         }
16     }
17 }
18
```

JAVA-

```

76 void findSubarraysWithTargetSum(int[] nums, int size, int targetSum) {
77     for (int start = 0; start < size; start++) {
78         int currentSum = 0;
79         for (int end = start; end < size; end++) {
80             currentSum += nums[end];
81             if (currentSum == targetSum) {
82                 for (int k = start; k <= end; k++) {
83                     System.out.print(nums[k] + " ");
84                 }
85                 System.out.println();
86             }
87         }
88     }
89 }

```

Optimized Solution-

```

19 void findSubarraysWithTargetSumOptimized(vector<int>& nums, int size, int targetSum) {
20     unordered_map<int, int> prefixSumMap;
21     int currentSum = 0;
22     for (int i = 0; i < size; i++) {
23         currentSum += nums[i];
24
25         // Check if the current cumulative sum equals the target sum
26         if (currentSum == targetSum) {
27             for (int j = 0; j <= i; j++) {
28                 cout << nums[j] << " ";
29             }
30             cout << endl;
31         }
32
33         // Check if there's a prefix sum that, when removed, equals the target sum
34         if (prefixSumMap.find(currentSum - targetSum) != prefixSumMap.end()) {
35             int startIndex = prefixSumMap[currentSum - targetSum] + 1;
36             for (int k = startIndex; k <= i; k++) {
37                 cout << nums[k] << " ";
38             }
39             cout << endl;
40         }
41
42         // Store the current cumulative sum and its index in the map
43         prefixSumMap[currentSum] = i;
44     }
45 }

```

JAVA-

```

50 void findSubarraysWithTargetSumOptimized(int[] nums, int size, int targetSum) {
51     Map<Integer, Integer> prefixSumMap = new HashMap<>();
52     int currentSum = 0;
53
54     for (int i = 0; i < size; i++) {
55         currentSum += nums[i];
56
57         if (currentSum == targetSum) {
58             for (int j = 0; j <= i; j++) {
59                 System.out.print(nums[j] + " ");
60             }
61             System.out.println();
62         }
63
64         if (prefixSumMap.containsKey(currentSum - targetSum)) {
65             int startIndex = prefixSumMap.get(currentSum - targetSum) + 1;
66             for (int k = startIndex; k <= i; k++) {
67                 System.out.print(nums[k] + " ");
68             }
69             System.out.println();
70         }
71
72         prefixSumMap.put(currentSum, i);
73     }
74 }

```

Q2- There is a robot on an $m \times n$ grid. The robot is initially located at the top-left corner (i.e., `grid[0][0]`).

The robot tries to move to the bottom-right corner (i.e., `grid[m - 1][n - 1]`).

The robot can only move either down or right at any point in time.

Given the two integers m and n , return the number of possible unique paths that the robot can take to reach

the bottom-right corner.

I/O: $m = 3, n = 2$

O/P: 3

Approach-

This approach calculates the number of unique paths in an $m \times n$ grid using recursion. Starting from the top-left corner (0,0), it recursively explores moving right and down until it reaches the bottom-right corner ($m-1, n-1$). The base cases handle reaching the grid boundaries or the destination. The function sums the paths from moving right and down to compute the total number of unique paths.

```

1  class Solution {
2  public:
3  int recursion(int m,int n,int i,int j){
4
5      if(i==m || j==n){
6          return 0;
7      }
8
9      if(i==m-1 && j==n-1){
10         return 1;
11     }
12
13     return recursion(m,n,i+1,j)+recursion(m,n,i,j+1);
14 }
15
16 int uniquePaths(int m, int n) {
17     return recursion(m,n,0,0);
18 }
19 }
20
21 };

```

Date- 26 April S2

Q1- Given two Integer, find sum of cubes of all numbers in the range of n & m.

I/P- n=1 , m=13

O/P- 8281

Approach-

The CubeSum function calculates the sum of cubes of all integers from start to end (inclusive). It iterates through each integer in this range, computes its cube, and accumulates these values in the variable Sum. After completing the iteration, it prints the total sum.

```

5 void cubeSum(int start, int end) {
6     int Sum = 0;
7     for (int i = start; i <= end; i++) {
8         Sum += i * i * i;
9     }
10    cout << Sum << endl;
11 }

```

JAVA-

```

15 void cubeSum(int start, int end) {
16     int sum = 0;
17     for (int i = start; i <= end; i++) {
18         sum += i * i * i;
19     }
20     System.out.println(sum);
21 }

```

Q2-You are given a grocery List which consists of three parameters ITEMS, QUANTITY,PRICE.

Your task is to find 1)Higher Selling item 2)Total Selling item 3)Avg. Selling item

I/P- Pen ,20,10

Notebook,100,10

Pencil,10,10

O/P- Notebook , 1300,433.33

Approach-

1. Input and Calculation: The program reads item details, calculates each item's total cost, and updates the overall total cost.
2. Maximum Cost Tracking: It keeps track of the item with the highest total cost.
3. Average Cost: It computes the average cost by dividing the total cost by the number of items.
4. Output: Finally, it prints the most expensive item, the total cost formatted to two decimal places, and the average cost.

```

7- int main() {
8     int numItems;
9     cin >> numItems;
10
11     int maxItemCost = 0;
12     string maxItemCostName = "";
13     double totalCost = 0;
14     double averageCost = 0;
15
16-    for (int i = 0; i < numItems; i++) {
17        string itemName;
18        int itemQuantity, itemPrice;
19        cin >> itemName >> itemQuantity >> itemPrice;
20
21        int currentItemCost = itemQuantity * itemPrice;
22        totalCost += currentItemCost;
23
24-        if (currentItemCost > maxItemCost) {
25            maxItemCost = currentItemCost;
26            maxItemCostName = itemName;
27        }
28
29        averageCost = totalCost / (i + 1);
30    }
31
32    cout << "Most Expensive Item: " << maxItemCostName << "\nTotal Cost: "
33         << fixed << setprecision(2) << totalCost << "\nAverage Cost: "
34         << averageCost << endl;
35
36    return 0;

```

JAVA-

```

3- public class Main {
4-     public static void main(String[] args) {
5         Scanner scanner = new Scanner(System.in);
6
7         int numItems = scanner.nextInt();
8         int maxItemCost = 0;
9         String maxItemCostName = "";
10        double totalCost = 0;
11        double averageCost = 0;
12
13-        for (int i = 0; i < numItems; i++) {
14            String itemName = scanner.next();
15            int itemQuantity = scanner.nextInt();
16            int itemPrice = scanner.nextInt();
17
18            int currentItemCost = itemQuantity * itemPrice;
19            totalCost += currentItemCost;
20
21-            if (currentItemCost > maxItemCost) {
22                maxItemCost = currentItemCost;
23                maxItemCostName = itemName;
24            }
25        }
26
27        averageCost = totalCost / numItems;
28
29        System.out.println("Most Expensive Item: " + maxItemCostName);
30        System.out.printf("Total Cost: %.2f\n", totalCost);
31        System.out.printf("Average Cost: %.2f\n", averageCost);
32
33        scanner.close();
34    }
35 }

```

Date- 29 April S1

Q1- Given an Integer , we need to find the sum of values of that table.

I/P- 12

O/P- 660

Approach-

The program reads an integer `number` from the user and then calculates the total sum of `number` multiplied by each integer from 1 up to `number`. It does this by iterating through values from 1 to `number`, adding the product of `number` and each iterator value `i` to a running total. After completing the iteration, it prints the total sum, which represents the sum of all these products.

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      int number;
7      cin >> number;
8
9      int totalSum = 0;
10     for (int i = 1; i <= number; i++) {
11         totalSum += number * i;
12     }
13
14     cout << totalSum << endl;
15     return 0;
16 }
17
```

JAVA-

```
17  import java.util.Scanner;
18
19  public class Main {
20      public static void main(String[] args) {
21          Scanner scanner = new Scanner(System.in);
22
23          int number = scanner.nextInt();
24          int totalSum = 0;
25
26          for (int i = 1; i <= number; i++) {
27              totalSum += number * i;
28          }
29
30          System.out.println(totalSum);
31          scanner.close();
32      }
33  }
```

Q2-Given an Array and a integer k , we need to find the maximum element in each of the contiguous subarrays.

I/P- [2,3,6,1,5,1]

k=3

O/P- [6,6,6,5]

Approach-

The function finds the maximum element in each subarray of size windowSize from a given array. It does this by iterating through the array and for each subarray, it uses a max heap (priority queue) to determine the maximum value within that subarray. The maximum value is then stored in a result vector. Finally, the function prints all the maximum values for each subarray.

```
20 void findMaxInSubarraysPriorityQueue(vector<int>& nums, int windowSize) {
21     vector<int> maxElements;
22     for (int i = 0; i <= nums.size() - windowSize; i++) {
23         priority_queue<int> maxHeap;
24         for (int j = i; j < i + windowSize; j++) {
25             maxHeap.push(nums[j]);
26         }
27         maxElements.push_back(maxHeap.top());
28     }
29     for (int num : maxElements) {
30         cout << num << " ";
31     }
32     cout << endl;
33 }
```

JAVA-

```
51 void findMaxInSubarraysPriorityQueue(int[] nums, int windowSize) {
52     List<Integer> maxElements = new ArrayList<>();
53     for (int i = 0; i <= nums.length - windowSize; i++) {
54         PriorityQueue<Integer> maxHeap = new PriorityQueue<>(Collections.reverseOrder());
55         for (int j = i; j < i + windowSize; j++) {
56             maxHeap.add(nums[j]);
57         }
58         maxElements.add(maxHeap.peek());
59     }
60     for (int num : maxElements) {
61         System.out.print(num + " ");
62     }
63     System.out.println();
64 }
```

Date- 29 April S2

Q1-Calculate the sum of N terms if Fibonacci Series

Note: fib(0)=1

fib(1)=1

I/P: n=7

O/P: 33

Approach-

The function calculates the sum of Fibonacci numbers up to the n-th position. It initializes the first two Fibonacci numbers and starts the sum with 1. In a loop, it computes each subsequent Fibonacci number, adds it to the total sum, and updates the previous and current Fibonacci numbers accordingly. The final result is the total sum of Fibonacci numbers up to the n-th position. If n is less than 0, it returns -1 as an error code.

```
5  int sumOfFibonacciNumbers(int n) {
6      int prev = 0, current = 1, totalSum = 1;
7      if (n < 0) {
8          return -1;
9      } else if (n == 0) {
10         return prev;
11     } else if (n == 1) {
12         return totalSum;
13     } else {
14         for (int i = 2; i < n; ++i) {
15             int next = prev + current; // Calculate the next Fibonacci number
16             totalSum += next; // Add the next Fibonacci number to the total sum
17             prev = current; // Update previous Fibonacci number
18             current = next; // Update current Fibonacci number
19         }
20         return totalSum;
21     }
22 }
```

JAVA-

```
48  int sumOfFibonacciNumbers(int n) {
49      int prev = 0, current = 1, totalSum = 1;
50      if (n < 0) {
51          return -1;
52      } else if (n == 0) {
53          return prev;
54      } else if (n == 1) {
55          return totalSum;
56      } else {
57          for (int i = 2; i < n; ++i) {
58              int next = prev + current; // Calculate the next Fibonacci number
59              totalSum += next; // Add the next Fibonacci number to the total sum
60              prev = current; // Update previous Fibonacci number
61              current = next; // Update current Fibonacci number
62          }
63          return totalSum;
64      }
65 }
```

Q2- Given an integer array arr, return the number of distinct bitwise ORs of all the non-empty subarrays of arr. The bitwise OR of a subarray is the bitwise OR of each integer in the subarray. The bitwise OR of a subarray of one integer is that integer.

I/P: arr = [1,1,2]

O/P: 3

Approach-

The function calculates the number of unique results of bitwise OR operations on all possible subarrays of a given array. It iterates through each element, updating a list of bitwise OR results by combining the current number with each existing result. This ensures that all possible subarray OR values are considered. Finally, it uses a set to determine and return the number of unique bitwise OR results.

```
6  int countUniqueBitwiseORs(vector<int>& nums) {
7      vector<int> bitwiseORs;
8      int start = 0;
9
10     for (int num : nums) {
11         int end = bitwiseORs.size();
12         bitwiseORs.push_back(num); // Add the current number as a new subarray OR
13         for (int i = start; i < end; ++i) {
14             int newOR = bitwiseORs[i] | num;
15             if (bitwiseORs.back() != newOR) {
16                 bitwiseORs.push_back(newOR);
17             }
18         }
19         start = end; // Update the start index for the next iteration
20     }
21
22     // Use a set to get the count of unique OR results
23     return unordered_set<int>(bitwiseORs.begin(), bitwiseORs.end()).size();
24 }
```

JAVA-

```
27  int countUniqueBitwiseORs(List<Integer> nums) {
28      List<Integer> bitwiseORs = new ArrayList<>();
29      int start = 0;
30
31      for (int num : nums) {
32          int end = bitwiseORs.size();
33          bitwiseORs.add(num); // Add the current number as a new subarray OR
34          for (int i = start; i < end; ++i) {
35              int newOR = bitwiseORs.get(i) | num;
36              if (!bitwiseORs.get(bitwiseORs.size() - 1).equals(newOR)) {
37                  bitwiseORs.add(newOR);
38              }
39          }
40          start = end; // Update the start index for the next iteration
41      }
42
43      // Use a set to get the count of unique OR results
44      return new HashSet<>(bitwiseORs).size();
45 }
```

Date- 30 April S1

Q1- Find majority elements, where majority elements is the element which occurs more than or equal to $N/2$ times the array, where 'N' is the size of the array.

I/P- 8

[5,5,4,4,5,4,5,4]

O/P- 5,4

Approach-

The function identifies and prints elements in an array that appear at least $\lfloor N/2 \rfloor$ times, where $N/2$ is the size of the array. It uses an unordered map to count the frequency of each element. After populating the map, it iterates through the map entries and prints elements that meet or exceed the frequency threshold of $N/2$. This approach ensures that only majority elements are output.

```
9 void MajorityElements(const vector<int>& arr, int N) {
10     unordered_map<int, int> frequencyMap;
11
12     // Count occurrences of each element
13     for (int num : arr) {
14         frequencyMap[num]++;
15     }
16
17     // Print elements that appear more than or equal to N / 2 times
18     for (const auto& entry : frequencyMap) {
19         if (entry.second >= N / 2) {
20             cout << entry.first << " ";
21         }
22     }
23     cout << endl; // Print newline for better output formatting
24 }
```

JAVA-

```
28 void MajorityElements(List<Integer> arr, int N) {
29     Map<Integer, Integer> frequencyMap = new HashMap<>();
30
31     // Count occurrences of each element
32     for (int num : arr) {
33         frequencyMap.put(num, frequencyMap.getOrDefault(num, 0) + 1);
34     }
35
36     // Print elements that appear more than or equal to N / 2 times
37     for (Map.Entry<Integer, Integer> entry : frequencyMap.entrySet()) {
38         if (entry.getValue() >= N / 2) {
39             System.out.print(entry.getKey() + " ");
40         }
41     }
42     System.out.println(); // Print newline for better output formatting
43 }
44
```

Q2-In a database there are N students, the fields of the table are NAME,AGE,GENDER. Your task is to return

the students Name who are greater than 20 years old & Calculate the average of grades using ascii value of female Candidates

I/P - 3

AAA 21 A Female

BBB 22 B Male

CCC 24 C Female

O/P - AAA BBB CCC

66

Approach-

The program processes a list of people and performs two tasks. First, it collects and prints names of individuals older than 20. Second, it calculates the average grade for female individuals by converting their grades from characters to integers, summing them up, and then dividing by the number of females. The results are printed at the end, showing names of older individuals and the average female grade.

```
26 #include <iostream>
27 #include <vector>
28 #include <string>
29
30 using namespace std;
31
32 int main() {
33     int numberOfEntries;
34     cin >> numberOfEntries;
35
36     vector<string> namesAbove20;
37     int femaleCount = 0;
38     int totalFemaleGrades = 0;
39     int totalFemales = 0;
40
41     while (numberOfEntries-- > 0) {
42         string personName, personGender;
43         char personGrade;
44         int personAge;
45         cin >> personName >> personAge >> personGrade >> personGender;
46
47         if (personAge > 20) {
48             namesAbove20.push_back(personName);
49         }
50
51         if (personGender == "Female") {
52             totalFemales++;
53             totalFemaleGrades += personGrade - '0'; // Convert grade from char to int
54         }
55     }
56
57     // Output names of people older than 20
58     for (const string& name : namesAbove20) {
59         cout << name << endl;
60     }
61
62     // Calculate and output average grade for females
63     double averageFemaleGrade = (totalFemales > 0) ? (double)totalFemaleGrades / totalFemales : 0.0;
64     cout << averageFemaleGrade << endl;
65
66     return 0;
67 }
```

JAVA-

```

89 import java.util.ArrayList;
90 import java.util.List;
91 import java.util.Scanner;
92
93 public class Main {
94     public static void main(String[] args) {
95         Scanner scanner = new Scanner(System.in);
96         int numberOfEntries = scanner.nextInt();
97         scanner.nextLine(); // Consume the newline
98
99         List<String> namesAbove20 = new ArrayList<>();
100         int totalFemaleGrades = 0;
101         int totalFemales = 0;
102
103         while (numberOfEntries-- > 0) {
104             String personName = scanner.next();
105             int personAge = scanner.nextInt();
106             char personGrade = scanner.next().charAt(0);
107             String personGender = scanner.next();
108
109             if (personAge > 20) {
110                 namesAbove20.add(personName);
111             }
112
113             if (personGender.equals("Female")) {
114                 totalFemales++;
115                 totalFemaleGrades += personGrade - '0'; // Convert grade from char to int
116             }
117         }
118
119         // Output names of people older than 20
120         for (String name : namesAbove20) {
121             System.out.println(name);
122         }
123
124         // Calculate and output average grade for females
125         double averageFemaleGrade = (totalFemales > 0) ? (double) totalFemaleGrades / totalFemales : 0.0;
126         System.out.println(averageFemaleGrade);
127     }
128 }
129

```

Date- 3 May S1

Q1- The Organization has data warehouse there will be given a three digit number.
Check Whether number is divisible by 9 or not ?

I/P- 162

O/P- Number 162 is divisible by 9.

I/P 236

O/P- Number 236 is not divisible by 9.

Approach-

In this approach, the function `isDivisibleByNine` checks if a given number is divisible by 9 by using the modulus operator. In the main function, it reads an integer input and first checks if the number is within the range of 100 to 999. If it is, the function `isDivisibleByNine` is called to determine divisibility. Based on the result, the program prints whether the number is divisible by 9 or not.

```
1  #include <iostream>
2  using namespace std;
3
4  bool isDivisibleByNine(int number) {
5      return number % 9 == 0;
6  }
7
8  int main() {
9      int number;
10     cin >> number;
11     if (number >= 100 && number <= 999) {
12         if (isDivisibleByNine(number)) {
13             cout << "Number " << number << " is divisible by 9" << std::endl;
14         } else {
15             cout << "Number " << number << " is not divisible by 9" << std::endl;
16         }
17     }
18
19     return 0;
20 }
```

JAVA-

```
22  import java.util.Scanner;
23
24  public class Main {
25
26      public static boolean isDivisibleByNine(int number) {
27          return number % 9 == 0;
28      }
29
30      public static void main(String[] args) {
31          Scanner scanner = new Scanner(System.in);
32          int number = scanner.nextInt();
33
34          if (number >= 100 && number <= 999) {
35              if (isDivisibleByNine(number)) {
36                  System.out.println("Number " + number + " is divisible by 9");
37              } else {
38                  System.out.println("Number " + number + " is not divisible by 9");
39              }
40          }
41      }
42  }
```

Q2- We are given a list of numbers we need to return maximum difference b/w smallest & largest Number.

I/P- n=4

[1,5,7,3]

O/P- 6

Approach-

In this approach, the function findMaxDifference calculates the maximum difference between the smallest and largest numbers in a vector. It starts by initializing the minVal and maxVal to the first element of the vector. As it iterates through each element, it updates minVal and maxVal if a smaller or larger value is found. Finally, it returns the difference between the largest and smallest values.

```
21 int findMaxDifference(const vector<int>& numbers) {
22     // Edge case: if the vector is empty
23     if (numbers.empty()) {
24         return 0;
25     }
26
27     int minVal = numbers[0];
28     int maxVal = numbers[0];
29
30
31     for (int num : numbers) {
32         if (num < minVal) {
33             minVal = num;
34         }
35         if (num > maxVal) {
36             maxVal = num;
37         }
38     }
39
40     // Return the difference
41     return maxVal - minVal;
42 }
```

JAVA-

```
68 public int findMaxDifference(int[] numbers) {
69     // Edge case: if the array is empty
70     if (numbers.length == 0) {
71         return 0;
72     }
73
74     int minVal = numbers[0];
75     int maxVal = numbers[0];
76
77     for (int num : numbers) {
78         if (num < minVal) {
79             minVal = num;
80         }
81         if (num > maxVal) {
82             maxVal = num;
83         }
84     }
85
86     // Return the difference
87     return maxVal - minVal;
88 }
```

Date- 3May S2

Q1-A person has many shoes of different sizes and he wants to arrange them, Calculate the numbers of pairs of shoes.

I/P- 6

7L 8L 6R 6L 7L 8R

O/P- 2

Approach-

In this approach, the function `countPairs` calculates the number of shoe pairs from a list of shoes. It uses two unordered maps to count the occurrences of left and right shoes for each size. The function then iterates over the map of left shoes, and for each size, it finds the corresponding count of right shoes. It adds the minimum of the counts for each size to the total number of pairs. Finally, it returns the total number of pairs.

```
11 int countPairs(const vector<string>& shoes) {
12     unordered_map<string, int> leftShoeCount;
13     unordered_map<string, int> rightShoeCount;
14
15     // Count occurrences of left and right shoes
16     for (const string& shoe : shoes) {
17         char side = shoe.back();
18         string size = shoe.substr(0, shoe.size() - 1);
19
20         if (side == 'L') {
21             leftShoeCount[size]++;
22         } else if (side == 'R') {
23             rightShoeCount[size]++;
24         }
25     }
26
27     // Calculate number of pairs
28     int pairs = 0;
29     for (const auto& entry : leftShoeCount) {
30         string size = entry.first;
31         int leftCount = entry.second;
32         int rightCount = rightShoeCount[size];
33         pairs += min(leftCount, rightCount);
34     }
35
36     return pairs;
37 }
```

JAVA-


```

60 public int countPairs(String[] shoes) {
61     Map<String, Integer> leftShoeCount = new HashMap<>();
62     Map<String, Integer> rightShoeCount = new HashMap<>();
63
64     // Count occurrences of left and right shoes
65     for (String shoe : shoes) {
66         char side = shoe.charAt(shoe.length() - 1);
67         String size = shoe.substring(0, shoe.length() - 1);
68
69         if (side == 'L') {
70             leftShoeCount.put(size, leftShoeCount.getOrDefault(size, 0) + 1);
71         } else if (side == 'R') {
72             rightShoeCount.put(size, rightShoeCount.getOrDefault(size, 0) + 1);
73         }
74     }
75
76     // Calculate number of pairs
77     int pairs = 0;
78     for (Map.Entry<String, Integer> entry : leftShoeCount.entrySet()) {
79         String size = entry.getKey();
80         int leftCount = entry.getValue();
81         int rightCount = rightShoeCount.getOrDefault(size, 0);
82         pairs += Math.min(leftCount, rightCount);
83     }
84
85     return pairs;
86 }

```

Q2- In a company there are employees and their efficiency is given in array (can be negative) you need to find the maximum efficiency of 3 employees. The efficiency of 3 employees will be calculated by multiplying their individual efficiencies from the given array.

I/P - 5

[2,-5,7,6,3]

O/P – 84

Approach-

In this approach, the function `findMaxProductOfThree` computes the maximum product of any three numbers from a vector of efficiencies. It first checks if there are at least three employees; if not, it returns an error value. The function then sorts the vector to facilitate easy access to the largest and smallest values. It calculates the maximum product of the top three largest numbers and the product of the two smallest (potentially negative) numbers with the largest number. The function returns the greater of these two products as the maximum product of three numbers.

```

7  int findMaxProductOfThree(const vector<int>& efficiencies) {
8      int n = efficiencies.size();
9
10     // Handle cases where there are fewer than 3 employees
11     if (n < 3) {
12         cout << "Not enough employees." << endl;
13         return INT_MIN; // or some other error value
14     }
15
16     // Copy and sort the array
17     vector<int> sortedEfficiencies = efficiencies;
18     sort(sortedEfficiencies.begin(), sortedEfficiencies.end());
19
20     // Calculate the maximum product of three numbers
21     int maxProduct1 = sortedEfficiencies[n-1] * sortedEfficiencies[n-2] * sortedEfficiencies[n-3];
22     int maxProduct2 = sortedEfficiencies[0] * sortedEfficiencies[1] * sortedEfficiencies[n-1];
23
24     return max(maxProduct1, maxProduct2);
25 }

```

JAVA-

```

27  import java.util.Arrays;
28
29  public int findMaxProductOfThree(int[] efficiencies) {
30      int n = efficiencies.length;
31
32      // Handle cases where there are fewer than 3 employees
33      if (n < 3) {
34          System.out.println("Not enough employees.");
35          return Integer.MIN_VALUE; // or some other error value
36      }
37
38      // Copy and sort the array
39      int[] sortedEfficiencies = efficiencies.clone();
40      Arrays.sort(sortedEfficiencies);
41
42      // Calculate the maximum product of three numbers
43      int maxProduct1 = sortedEfficiencies[n-1] * sortedEfficiencies[n-2] * sortedEfficiencies[n-3];
44      int maxProduct2 = sortedEfficiencies[0] * sortedEfficiencies[1] * sortedEfficiencies[n-1];
45
46      return Math.max(maxProduct1, maxProduct2);
47  }

```

Date- 6 May S1

Q1- Given an array nums of size n, return the majority element. The Majority element is the element that appears more than $n/3$ times. You may assume that the majority element always exists in the array.

I/P- nums=[2,2,2,4,5,5]

O/P-2

Approach-

In this approach, the function `findMajorityElement` identifies and prints elements in the vector that appear at least $(N/3)$ times. It uses an unordered map to count the frequency of each element. After populating the map with counts, it iterates through the map and prints elements whose frequency meets or exceeds the $(N/3)$ threshold. This approach efficiently determines majority elements based on their occurrence in the vector.

```
5 void findMajorityElement(vector<string>& arr, int N) {
6     unordered_map<string, int> freq;
7     for (string& n : arr) {
8         if (freq.find(n) != freq.end()) {
9             freq[n]++;
10        } else {
11            freq[n] = 1;
12        }
13    }
14    for (auto& it : freq) {
15        if (it.second >= N / 3) {
16            cout << it.first << " ";
17        }
18    }
19 }
20
```

JAVA-

```
21 import java.util.HashMap;
22 import java.util.Map;
23
24 public void findMajorityElement(String[] arr, int N) {
25     HashMap<String, Integer> freq = new HashMap<>();
26
27     // Count occurrences of each element
28     for (String n : arr) {
29         freq.put(n, freq.getOrDefault(n, 0) + 1);
30     }
31
32     // Print elements that appear more than or equal to N / 3 times
33     for (Map.Entry<String, Integer> entry : freq.entrySet()) {
34         if (entry.getValue() >= N / 3) {
35             System.out.print(entry.getKey() + " ");
36         }
37     }
38     System.out.println(); // Print newline for better output formatting
39 }
```

Q2: Given an array balls containing numbers like '3', '5', and '6', sort the array in-place so that balls with the same number

are adjacent, in the order '3', '5', '6'.

I/P- balls = [5, 3, 3, 6, 5, 6]

O/P- [3, 3, 5, 5, 6, 6]

Approach-

In this approach, the function sortBalls sorts an array of balls with values 3, 5, and 6 using the Dutch National Flag algorithm. Three pointers (low, mid, and high) are used to manage the positions of these values in the array. The low pointer tracks where the next 3 should be placed, the mid pointer iterates through the array, and the high pointer tracks where the next 6 should be placed. The algorithm swaps values based on their type to ensure that all 3s are at the beginning, 5s in the middle, and 6s at the end of the array.

```
25 void sortBalls(vector<int>& balls) {
26     int low = 0;    // Pointer for the next position of 3
27     int mid = 0;    // Pointer for the current element
28     int high = balls.size() - 1; // Pointer for the next position of 6
29
30     // Process elements using the three pointers
31     while (mid <= high) {
32         if (balls[mid] == 3) {
33             swap(balls[low], balls[mid]);
34             low++;
35             mid++;
36         } else if (balls[mid] == 5) {
37             mid++;
38         } else if (balls[mid] == 6) {
39             swap(balls[mid], balls[high]);
40             high--;
41         }
42     }
43 }
```

JAVA-

```
66 import java.util.Arrays;
67
68 public void sortBalls(int[] balls) {
69     int low = 0;    // Pointer for the next position of 3
70     int mid = 0;    // Pointer for the current element
71     int high = balls.length - 1; // Pointer for the next position of 6
72
73     // Process elements using the three pointers
74     while (mid <= high) {
75         if (balls[mid] == 3) {
76             swap(balls, low, mid);
77             low++;
78             mid++;
79         } else if (balls[mid] == 5) {
80             mid++;
81         } else if (balls[mid] == 6) {
82             swap(balls, mid, high);
83             high--;
84         }
85     }
86 }
87
88 // Helper method to swap elements
89 private void swap(int[] balls, int i, int j) {
90     int temp = balls[i];
91     balls[i] = balls[j];
92     balls[j] = temp;
93 }
```

Date- 6 May S2

Q1- Given an array of size N-1 with integers in the range of [1,N] the task to find the missing number from the first N integers.

I/P- 6

[1,2,3,4,6,7]

O/P- 5

Approach-

In this approach, the function `Missing` finds the missing number in a sequence from 1 to (n). It calculates the sum of all elements in the given array and the expected sum of the first (n) natural numbers using the formula $n(n + 1)/2$. The missing number is then determined by subtracting the actual sum from the expected sum. This method efficiently identifies the missing number with a time complexity of ($O(n)$) and a space complexity of ($O(1)$).

```
6  int Missing(vector<int>& arr, int size) {
7      int Sum = 0;
8      int n = size + 1;
9      for (int num : arr) {
10         Sum += num;
11     }
12     int Formula_Sum = (n * (n + 1)) / 2;
13     return Formula_Sum - Sum;
14 }
```

JAVA-

```
16  public int findMissing(int[] arr) {
17      int size = arr.length;
18      int n = size + 1;
19      int sum = 0;
20      for (int num : arr) {
21         sum += num;
22     }
23     int formulaSum = (n * (n + 1)) / 2;
24     return formulaSum - sum;
25 }
```

Q2-Given two integers L and R, the task is to find the count of total numbers of prime numbers in the range [L, R] whose sum of the digits is also a prime number.

I/P- L=1,R=10

O/P- 4

Approach-

In this approach, the function `countPrimeWithPrimeDigitSum` identifies prime numbers within a given range `[L, R]` where the sum of their digits is also prime. It uses the `isPrime` function to check for primality of both the number and its digit sum. The `sumOfDigits` function computes the sum of digits for each number. The total count of such numbers is returned, providing a count of primes with a prime digit sum in the specified range.

```
16 #include <iostream>
17 #include <cmath>
18
19 using namespace std;
20
21 // Function to check if a number is prime
22 bool isPrime(int num) {
23     if (num <= 1) return false;
24     if (num <= 3) return true;
25     if (num % 2 == 0 || num % 3 == 0) return false;
26     for (int i = 5; i * i <= num; i += 6) {
27         if (num % i == 0 || num % (i + 2) == 0) return false;
28     }
29     return true;
30 }
31
32 // Function to calculate the sum of digits of a number
33 int sumOfDigits(int num) {
34     int sum = 0;
35     while (num > 0) {
36         sum += num % 10;
37         num /= 10;
38     }
39     return sum;
40 }
41
42 // Function to count primes in range [L, R] whose sum of digits is also prime
43 int countPrimeWithPrimeDigitSum(int L, int R) {
44     int count = 0;
45
46     for (int num = L; num <= R; ++num) {
47         if (isPrime(num)) {
48             int digitSum = sumOfDigits(num);
49             if (isPrime(digitSum)) {
50                 ++count;
51             }
52         }
53     }
54
55     return count;
56 }
57
58 int main() {
59     int L, R;
60     cout << "Enter L and R: ";
61     cin >> L >> R;
62
63     int result = countPrimeWithPrimeDigitSum(L, R);
64     cout << "The count of primes with prime digit sums in the range [" << L << ", " << R << "] is: " << result << endl;
65
66     return 0;
67 }
```

JAVA-

```
82 // Function to check if a number is prime
83 public boolean isPrime(int num) {
84     if (num <= 1) return false;
85     if (num <= 3) return true;
86     if (num % 2 == 0 || num % 3 == 0) return false;
87     for (int i = 5; i * i <= num; i += 6) {
88         if (num % i == 0 || num % (i + 2) == 0) return false;
89     }
90     return true;
91 }
92
93 // Function to calculate the sum of digits of a number
94 public int sumOfDigits(int num) {
95     int sum = 0;
96     while (num > 0) {
97         sum += num % 10;
98         num /= 10;
99     }
100     return sum;
101 }
102
103 // Function to count primes in range [L, R] whose sum of digits is also prime
104 public int countPrimeWithPrimeDigitSum(int L, int R) {
105     int count = 0;
106
107     for (int num = L; num <= R; ++num) {
108         if (isPrime(num)) {
109             int digitSum = sumOfDigits(num);
110             if (isPrime(digitSum)) {
111                 count++;
112             }
113         }
114     }
115
116     return count;
117 }
118
```

Date- 8May S1

Q1- Write a program to take input of X and Y in a new line. Print the number which is nearer to the integer when divided by Y.

I/P- X=25

Y=4

O/P-24

Approach –

In this approach, the program first defines a custom rounding function, customRound, that rounds a given double to the nearest integer, rounding up if the decimal part is 0.5 or greater. In the main function, the user inputs two integers x and y. The program calculates the quotient of x divided by y, rounds it using the customRound function, and then multiplies the rounded value by y to get the nearest multiple of y to x, which is then printed

```
1  int customRound(double number) {
2      int integerPart = (int) number;
3      double decimalPart = number - integerPart;
4
5      if (decimalPart >= 0.5) {
6          return integerPart + 1;
7      } else {
8          return integerPart;
9      }
10 }
11
12 int main() {
13     int x, y;
14     cin >> x;
15     cin >> y;
16
17     double nearest = (double) x / y;
18     int roundedNumber = customRound(nearest);
19     cout << roundedNumber * y << endl;
20
21     return 0;
22 }
```

JAVA-

```
12  public int customRound(double number) {
13      int integerPart = (int) number;
14      double decimalPart = number - integerPart;
15
16      if (decimalPart >= 0.5) {
17          return integerPart + 1;
18      } else {
19          return integerPart;
20      }
21 }
```


Q2-Write a program that generates a password adhering to the following conditions:

C1-The password must consist of at least 8 characters.

C2-It must contain at least one integer.

C3-It must contain at least one special character from the set {'#', '@'}.

C4-It must contain at least one uppercase letter and one lowercase letter.

C5-Each character in the password should be incremented by the number of times specified by the second input.

Your program should take two inputs:

1. A string representing the initial password.
2. An integer representing the number of times each character should be incremented.

Your program should then generate and output the modified password.

I/P 1: werV432@

I/P 2: 2

O/P: ygtX653#

Approach-

In this approach, the function `addValueToChars` processes each character of the input string. If the character is a letter, it shifts it by a given value within the alphabet, wrapping around using modulo arithmetic. If the character is a digit, it shifts similarly within the range of digits. Non-alphanumeric characters like `@` are conditionally transformed into `#` and vice versa. The modified characters are then concatenated to form the resulting string, which is returned.

```
string addValueToChars(string inputStr, int value) {
    string result = "";
    for (char ch : inputStr) {
        if (isalpha(ch)) {
            if (islower(ch)) {
                result += ((ch - 'a' + value) % 26) + 'a';
            } else {
                result += ((ch - 'A' + value) % 26) + 'A';
            }
        } else if (isdigit(ch)) {
            result += ((ch - '0' + value) % 10) + '0';
        } else {
            if (ch == '@') {
                result += '#';
            } else {
                result += '@';
            }
        }
    }
    return result;
}
```

JAVA-

```
58 public String addValueToChars(String inputStr, int value) {
59     StringBuilder result = new StringBuilder();
60     for (char ch : inputStr.toCharArray()) {
61         if (Character.isLetter(ch)) {
62             if (Character.isLowerCase(ch)) {
63                 result.append((char) ((ch - 'a' + value) % 26 + 'a'));
64             } else {
65                 result.append((char) ((ch - 'A' + value) % 26 + 'A'));
66             }
67         } else if (Character.isDigit(ch)) {
68             result.append((char) ((ch - '0' + value) % 10 + '0'));
69         } else {
70             result.append(ch == '@' ? '#' : '@');
71         }
72     }
73     return result.toString();
74 }
```

Date- 8 May S2

Q1-Given size of n and list of array elements and we should print if the given element in array is divisible by 3 then replace the element with "Three" and if the element in array is divisible by 5 then replace the element with "Five" if the element divisible by 3 and 5 both then replace the element with "ThreeFive" if the element in the array is not satisfying the above 3 conditions then put the element as it is and print the array

I/P- N=4

[8,3,9,5]

O/P- 8 Three 9 Five

Approach-

In this approach, the function `helperfun` iterates through the given array and checks each element to determine if it is divisible by 3, 5, or both. If a number is divisible by both 3 and 5, it prints "ThreeFive"; if only by 3, it prints "Three"; and if only by 5, it prints "Five". If the number is divisible by neither, it prints the number itself. This is similar to the FizzBuzz problem, where numbers are replaced by strings based on their divisibility.

```
5 void helperfun(const vector<int>& arr) {
6     for (int num : arr) {
7         if (num % 3 == 0 && num % 5 == 0) {
8             cout << "ThreeFive ";
9         } else if (num % 3 == 0) {
10            cout << "Three ";
11        } else if (num % 5 == 0) {
12            cout << "Five ";
13        } else {
14            cout << num << " ";
15        }
16    }
17    cout << endl;
18 }
```

JAVA-

```
21 import java.util.List;
22
23 public class Main {
24     public static void helperfun(List<Integer> arr) {
25         for (int num : arr) {
26             if (num % 3 == 0 && num % 5 == 0) {
27                 System.out.print("ThreeFive ");
28             } else if (num % 3 == 0) {
29                 System.out.print("Three ");
30             } else if (num % 5 == 0) {
31                 System.out.print("Five ");
32             } else {
33                 System.out.print(num + " ");
34             }
35         }
36         System.out.println();
37     }
38
39     public static void main(String[] args) {
40         // Example usage
41         List<Integer> exampleList = List.of(3, 5, 15, 7, 9, 10);
42         helperfun(exampleList);
43     }
44 }
```

Q2-

Task 1- print the collatz sequence upto ending with 1 the sequence should be in the following way

-> if the number is even:

$$F(n)=n//2$$

-> if the number is odd:

$$F(n)=3*n+1$$

This sequence should end until the last element of sequence is 1

Task-2

For the given integer from 1,n it should calculate the sequence of each k value Le,
 $1 \leq k \leq n$

Calculate the maximum length of sequence list of each k value and return the
maximum length of the sequence list of the k value and the k value itself

Task-3

For the given integer from 1.n it should calculate the sequence of each k value le,
 $1 \leq k \leq n$ Calculate the maximum value of the each sequence within the sequence list
of each k value and return the maximum value of the sequence list of the k value and
those k value itself

Input: 5

Output:

[5,16,8,4,2.1]

8,3

16,3

Input: 0

Output: Error!

Input: xyza

output: Error!

Input:-13

Output: Error!

Input: 5.5

Output: Error!

Approach –

In this approach, the code starts by checking if the input string represents a positive integer. If valid, it generates the Collatz sequence starting from the given integer n , where the sequence follows the rule: divide by 2 if even, or multiply by 3 and add 1 if odd, until n reaches 1. It then finds the number k between 1 and n that produces the longest sequence and the highest value within any sequence. The results include the entire sequence for n , the length of the longest sequence, and the highest value encountered. If the input is not valid, an error message is displayed.

```

27 // Function to check if a string represents a positive integer
28 bool isPositiveInteger(const string& str) {
29     for (char c : str) {
30         if (!isdigit(c)) {
31             return false;
32         }
33     }
34     return !str.empty() && stoi(str) > 0;
35 }
36
37 // Function to generate the Collatz sequence for a given number
38 vector<int> generateCollatzSequence(int num) {
39     vector<int> sequence;
40     sequence.push_back(num);
41     while (num != 1) {
42         if (num % 2 == 0)
43             num /= 2;
44         else
45             num = 3 * num + 1;
46         sequence.push_back(num);
47     }
48     return sequence;
49 }
50
51 // Function to find the maximum length of the Collatz sequence and the corresponding number
52 pair<int, int> findMaxLengthAndNumber(int limit) {
53     int maxLength = 0, correspondingNum = 0;
54     for (int i = 1; i <= limit; ++i) {
55         int currentLength = generateCollatzSequence(i).size();
56         if (currentLength > maxLength) {
57             maxLength = currentLength;
58             correspondingNum = i;
59         }
60     }
61     return make_pair(maxLength, correspondingNum);
62 }
63
64 // Function to find the maximum value in the Collatz sequence and the corresponding number
65 pair<int, int> findMaxValueAndNumber(int limit) {
66     int maxValue = 0, correspondingNum = 0;
67     for (int i = 1; i <= limit; ++i) {
68         vector<int> sequence = generateCollatzSequence(i);
69         int currentMaxValue = *max_element(sequence.begin(), sequence.end());
70         if (currentMaxValue > maxValue) {
71             maxValue = currentMaxValue;
72             correspondingNum = i;
73         }
74     }
75     return make_pair(maxValue, correspondingNum);
76 }

```

JAVA-

```
137 import java.util.ArrayList;
138 import java.util.List;
139 import java.util.Scanner;
140 import java.util.Collections;
141
142 public class Collatz {
143
144     // Function to check if a string represents a positive integer
145     public static boolean isPositiveInteger(String str) {
146         try {
147             int num = Integer.parseInt(str);
148             return num > 0;
149         } catch (NumberFormatException e) {
150             return false;
151         }
152     }
153
154     // Function to generate the Collatz sequence for a given number
155     public static List<Integer> generateCollatzSequence(int num) {
156         List<Integer> sequence = new ArrayList<>();
157         sequence.add(num);
158         while (num != 1) {
159             if (num % 2 == 0)
160                 num /= 2;
161             else
162                 num = 3 * num + 1;
163             sequence.add(num);
164         }
165         return sequence;
166     }
167
168     // Function to find the maximum length of the Collatz sequence and the corresponding number
169     public static Pair<Integer, Integer> findMaxLengthAndNumber(int limit) {
170         int maxLength = 0, correspondingNum = 0;
171         for (int i = 1; i <= limit; ++i) {
172             List<Integer> sequence = generateCollatzSequence(i);
173             int currentLength = sequence.size();
174             if (currentLength > maxLength) {
175                 maxLength = currentLength;
176                 correspondingNum = i;
177             }
178         }
179         return new Pair<>(maxLength, correspondingNum);
180     }
181
182     // Function to find the maximum value in the Collatz sequence and the corresponding number
183     public static Pair<Integer, Integer> findMaxValueAndNumber(int limit) {
184         int maxValue = 0, correspondingNum = 0;
185         for (int i = 1; i <= limit; ++i) {
186             List<Integer> sequence = generateCollatzSequence(i);
187             int currentMaxValue = Collections.max(sequence);
188             if (currentMaxValue > maxValue) {
189                 maxValue = currentMaxValue;
190                 correspondingNum = i;
191             }
192         }
193         return new Pair<>(maxValue, correspondingNum);
194     }
195
196     public static void main(String[] args) {
197         Scanner scanner = new Scanner(System.in);
198         String input = scanner.next();
199         scanner.close();
200
201         if (isPositiveInteger(input)) {
202             int num = Integer.parseInt(input);
203             List<Integer> collatzSequence = generateCollatzSequence(num);
204             Pair<Integer, Integer> maxValAndNum = findMaxValueAndNumber(num);
205             Pair<Integer, Integer> maxLengthAndNum = findMaxLengthAndNumber(num);
206
207             // Print the generated sequence
208             System.out.print("Sequence: [");
209             for (int i = 0; i < collatzSequence.size(); ++i) {
210                 System.out.print(collatzSequence.get(i));
211                 if (i != collatzSequence.size() - 1) {
212                     System.out.print(", ");
213                 }
214             }
215             System.out.println("]");
216
217             // Print results
218             System.out.println("Max Length: " + maxLengthAndNum.getFirst() + " for number: " + maxLengthAndNum.getSecond());
219             System.out.println("Max Value: " + maxValAndNum.getFirst() + " for number: " + maxValAndNum.getSecond());
220         } else {
221             System.out.println("Error! Input must be a positive integer.");
222         }
223     }
224 }
```


Date- 9May S1

Q1) given an integer n, return true if it is an armstrong number, otherwise return false.

an armstrong number is a number that is equal to the sum of its own digits each raised to the power of the number of digits.

input:

[153, 371, 108]

output:

[true, true, false]

Approach-

In this approach, the function `isArmStrong` determines if a number `n` is an Armstrong number in `k`-digit space. An Armstrong number (or Narcissistic number) is one where the sum of its digits each raised to the power of `k` equals the number itself. The function computes this by iterating over each digit of `n`, raising it to the power of `k`, and summing these values. Finally, it checks if this sum matches the original number, returning `true` if they are equal and `false` otherwise.

```
4  bool isArmStrong(int n, int k) {
5      int sum = 0;
6      int num = n;
7      while (num > 0) {
8          int digit = num % 10;
9          sum += pow(digit, k);
10         num /= 10;
11     }
12     return sum == n;
13 }
```

JAVA-

```
46  public static boolean isArmstrong(int n, int k) {
47      int sum = 0;
48      int num = n;
49      while (num > 0) {
50          int digit = num % 10;
51          sum += Math.pow(digit, k);
52          num /= 10;
53      }
54      return sum == n;
55  }
```

Q2)Print Top Candidate from the given input

INPUT format: no. of input n ,top k to be sorted, candidate with marks

I/P:3 4 70 Johnny 85 Anmol 270 Vishy

O/P:Vishy:270

Anmol:85

Jhonny 70

Approach-

In this approach, the `main` function reads a line of input, which includes an integer `k`, an integer `N`, followed by pairs of `marks` and `name`. The input is split into tokens using `istringstream` and parsed into a vector `l`. The first two tokens are converted to integers `k` and `N`, respectively. Subsequent tokens are processed into pairs of integers (marks) and strings (names), which are stored in a vector of pairs. This vector is then sorted in descending order based on marks using `sort` with a custom comparator. Finally, the top `k` entries (or fewer if there aren't `k` entries) are printed, showing the names and their corresponding marks.

```
6  int main() {
7      string s;
8      getline(cin, s);
9      vector<string> l;
10     string token;
11     istringstream iss(s);
12     while (iss >> token) {
13         l.push_back(token);
14     }
15     int k = stoi(l[0]);
16     int N = stoi(l[1]);
17     vector<pair<int, string>> arr;
18
19     for (int i = 2; i < l.size(); i += 2) {
20         int marks = stoi(l[i]);
21         string name = l[i + 1];
22         arr.push_back(make_pair(marks, name));
23     }
24
25     sort(arr.begin(), arr.end(), greater<pair<int, string>>());
26
27     for (int top = 0; top < k && top < arr.size(); top++) {
28         cout << arr[top].second << ": " << arr[top].first << endl;
29     }
30
31     return 0;
32 }
```

JAVA-

```
33 import java.util.*;
34
35 public class Main {
36     public static void main(String[] args) {
37         Scanner sc = new Scanner(System.in);
38         String s = sc.nextLine();
39         String[] parts = s.split("\\s+");
40
41         int k = Integer.parseInt(parts[0]);
42         int N = Integer.parseInt(parts[1]);
43
44         List<Pair<Integer, String>> arr = new ArrayList<>();
45         for (int i = 2; i < parts.length; i += 2) {
46             int marks = Integer.parseInt(parts[i]);
47             String name = parts[i + 1];
48             arr.add(new Pair<>(marks, name));
49         }
50
51         arr.sort((a, b) -> b.getKey().compareTo(a.getKey()));
52
53         for (int top = 0; top < k && top < arr.size(); top++) {
54             System.out.println(arr.get(top).getValue() + ": " + arr.get(top).getKey());
55         }
56     }
57 }
58
59 class Pair<K, V> {
60     private K key;
61     private V value;
62
63     public Pair(K key, V value) {
64         this.key = key;
65         this.value = value;
66     }
67
68     public K getKey() {
69         return key;
70     }
71
72     public V getValue() {
73         return value;
74     }
75 }
```

Date-9 May S2

Q1)Print all combinations

I/P- n=3

[0,1,2]

O/P-[0,01,012,0123,1,12,123,2,23,3]

Approach –

In this approach, the `generateCombinations` function recursively generates all possible combinations of elements from the given vector `arr`. It starts at the specified `index` and uses the `current` string to build combinations. For each element at `arr[index]`, the function is called twice: once including the current element in the combination (appending it to `current`), and once excluding it (not modifying `current`). The base case is when `index` reaches the end of the vector, at which point it prints the current combination if it's not empty. This recursive method explores all subsets of the array, including the empty subset.

```
4 void generateCombinations(vector<int>& arr, int index, string current) {
5     // Base case
6     if (index == arr.size()) {
7         if (!current.empty()) {
8             cout << current << " ";
9         }
10        return;
11    }
12
13    //include
14    generateCombinations(arr, index + 1, current + to_string(arr[index]));
15
16    //Exclude
17    generateCombinations(arr, index + 1, current);
18 }
```

JAVA-

```
20 generateCombinations(List<Integer> arr, int index, String current) {
21     // Base case
22     if (index == arr.size()) {
23         if (!current.isEmpty()) {
24             System.out.print(current + " ");
25         }
26         return;
27     }
28
29     // Include
30     generateCombinations(arr, index + 1, current + arr.get(index));
31
32     // Exclude
33     generateCombinations(arr, index + 1, current);
34 }
35 }
```

Date-14May S1

Q1) You are given an integer array num. The unique elements of an array are the elements that appear exactly once in the array.

Return sum of all unique elements of nums.

I/P- [1,2,5,5,7]

O/P-10

Approach –

In this approach, the sumOfUniqueElements function calculates the sum of all unique elements in a vector nums. It first uses an unordered_map to count the frequency of each element. Then, it iterates through this map, adding elements to the sum only if they appear exactly once in the array. The final sum of these unique elements is returned.

```
32 #include <iostream>
33 #include <vector>
34 #include <unordered_map>
35
36 using namespace std;
37
38 int sumOfUniqueElements(vector<int>& nums) {
39     unordered_map<int, int> frequency;
40
41     // Count the frequency of each element
42     for (int num : nums) {
43         frequency[num]++;
44     }
45
46     int sum = 0;
47     for (auto& pair : frequency) {
48         if (pair.second == 1) {
49             sum += pair.first;
50         }
51     }
52
53     return sum;
54 }
55
56 int main() {
57     vector<int> nums = {1, 2, 5, 5, 7};
58
59     int result = sumOfUniqueElements(nums);
60     cout << "The sum of unique elements is: " << result << endl;
61
62     return 0;
63 }
64
```

JAVA-

```
65 private static int sumOfUniqueElements(List<Integer> nums) {  
66     Map<Integer, Integer> frequency = new HashMap<>();  
67  
68     // Count the frequency of each element  
69     for (int num : nums) {  
70         frequency.put(num, frequency.getOrDefault(num, 0) + 1);  
71     }  
72  
73     int sum = 0;  
74     for (Map.Entry<Integer, Integer> entry : frequency.entrySet()) {  
75         if (entry.getValue() == 1) {  
76             sum += entry.getKey();  
77         }  
78     }  
79  
80     return sum;  
81 }
```

Q2) Given an integer array nums, find the subarray with largest sum, and return its sum.

I/P-[-1,2,4,2,-5,3]

O/P-8

Approach-

In this approach, the program reads a string of space-separated integers from the input and stores them in a vector. It then calculates the maximum sum of any contiguous subarray using a modified version of Kadane's algorithm. The currentSum keeps track of the sum of the current subarray, updating the maxSum whenever a higher sum is found. If currentSum becomes negative, it is reset to zero. Finally, the maximum subarray sum is printed.

```

20 #include <iostream>
21 #include <vector>
22 #include <algorithm>
23 #include <sstream>
24
25 using namespace std;
26
27 int main() {
28     string input;
29     getline(cin, input);
30     stringstream ss(input);
31     vector<int> nums;
32     int num;
33     while (ss >> num) {
34         nums.push_back(num);
35     }
36     int maxSum = nums[0];
37     int currentSum = 0;
38     for (int i = 0; i < nums.size(); ++i) {
39         currentSum += nums[i];
40         if (currentSum > maxSum) {
41             maxSum = max(currentSum, maxSum);
42         }
43         if (currentSum < 0) {
44             currentSum = 0;
45         }
46     }
47     cout << maxSum << endl;
48     return 0;
49 }

```

JAVA-

```

87 public class Main {
88     public static void main(String[] args) {
89         Scanner scanner = new Scanner(System.in);
90         String input = scanner.nextLine();
91         scanner.close();
92
93         String[] parts = input.split("\\s+");
94         List<Integer> nums = new ArrayList<>();
95         for (String part : parts) {
96             nums.add(Integer.parseInt(part));
97         }
98
99         int maxSum = nums.get(0);
100         int currentSum = 0;
101
102         for (int i = 0; i < nums.size(); ++i) {
103             currentSum += nums.get(i);
104             if (currentSum > maxSum) {
105                 maxSum = Math.max(currentSum, maxSum);
106             }
107             if (currentSum < 0) {
108                 currentSum = 0;
109             }
110         }
111
112         System.out.println(maxSum);
113     }
114 }

```

Date- 14 May S2

Q1) You are tasked with writing a program to calculate the total shipping cost based on the weight of the package and the distance it needs to travel. The shipping cost is determined by the following criteria:

1. Base money: \$5.00
2. Cost per kilogram: \$2.00
3. Cost per 10 kilometers: \$0.5

Example:

10 (w)

100 (D)

output: \$30.00

Approach –

In this approach, the `computeShippingCost` function calculates the total shipping cost for a package based on its weight and delivery distance. It starts with a base fee of \$5.00. The cost is then increased based on the weight of the package, where each kilogram costs \$2.00. Additionally, the cost is incremented based on the delivery distance, with every 10 kilometers adding \$0.50. The total shipping cost is the sum of these charges.

```
39 double computeShippingCost(int packageWeight, int deliveryDistance) {  
40     double baseFee = 5.00;  
41     double costPerKg = 2.00;  
42     double costPer10Km = 0.50;  
43  
44     double weightCharge = packageWeight * costPerKg;  
45     double distanceCharge = (deliveryDistance / 10) * costPer10Km;  
46  
47     double totalShippingCost = baseFee + weightCharge + distanceCharge;  
48     return totalShippingCost;  
49 }  
50
```

JAVA-

```
69 static double computeShippingCost(int packageWeight, int deliveryDistance) {  
70     double baseFee = 5.00;  
71     double costPerKg = 2.00;  
72     double costPer10Km = 0.50;  
73  
74     double weightCharge = packageWeight * costPerKg;  
75     double distanceCharge = (deliveryDistance / 10) * costPer10Km;  
76  
77     double totalShippingCost = baseFee + weightCharge + distanceCharge;  
78     return totalShippingCost;  
79 }
```


Q2) Given an array of integers nums and an integer k, return the total number of subarrays whose sum equals to k. A subarray is a contiguous non-empty sequence of elements within an array.

Input: nums = [1,2,3,-1,1]

K=3

Output: 6

Approach-

In this approach, the function calculates the number of subarrays whose sum equals a given value `k`. It uses a hash map to keep track of prefix sums and their frequencies. As the function iterates through the array, it maintains a running total (`preSum`) of the elements encountered. For each element, it checks if there is a previous prefix sum that would make the current subarray sum equal to `k` by subtracting `k` from `preSum` and looking it up in the hash map. The frequency of such prefix sums is added to the count. Finally, the hash map is updated with the current prefix sum. This method efficiently computes the count in linear time.

```
4  int subarraySum(vector<int>& arr, int k) {
5      unordered_map<int, int> mpp;
6      mpp[0] = 1;
7      int preSum = 0, cnt = 0;
8      for (int i = 0; i < arr.size(); ++i) {
9          preSum += arr[i];
10         int rest = preSum - k;
11         cnt += mpp[rest];
12         mpp[preSum] += 1;
13     }
14     return cnt;
15 }
```

JAVA-

```
18  import java.util.HashMap;
19  import java.util.Map;
20
21  public class SubarraySum {
22      public static int subarraySum(int[] arr, int k) {
23          Map<Integer, Integer> mpp = new HashMap<>();
24          mpp.put(0, 1);
25          int preSum = 0, cnt = 0;
26          for (int num : arr) {
27              preSum += num;
28              int rest = preSum - k;
29              cnt += mpp.getOrDefault(rest, 0);
30              mpp.put(preSum, mpp.getOrDefault(preSum, 0) + 1);
31          }
32          return cnt;
33      }
34  }
```