Indore Institute of Science & Technology
Department of Computer Science & Allied Department

List of Programs
SET-3 Algolution

1)In the theory of numbers, square free numbers have a special place. A square free number is one that is not divisible by a perfect square (other than 1). Thus 72 is divisible by 36 (a perfect square), and is not a square free number, but 70 has factors 1, 2, 5, 7, 10, 14, 35 and 70. As none of these are perfect squares (other than 1), 70 is a square free number.

For some algorithms, it is important to find out the square free numbers that divide a number. Note that 1 is not considered a square free number.
In this problem, you are asked to write a program to find the number of square free numbers that divide a given number.

Input:
The only line of the input is a single integer N which is divisible by no prime number larger than 19

Output:
One line containing an integer that gives the number of square free numbers (not including 1)

Constraints:
N < 10^9

Difficulty Level:
Simple

Time Limit (secs):
1

Examples:
Example 1:

Input:
20

Output:
3

Explanation
N=20

If we list the numbers that divide 20, they are
1, 2, 4, 5, 10, 20

1 is not a square free number, 4 is a perfect square, and 20 is divisible by 4, a perfect square. 2 and 5, being prime, are square free, and 10 is divisible by 1,2,5 and 10, none of which are perfect squares. Hence the square free numbers that divide 20 are 2, 5, 10. Hence the result is 3.

Example 2:
Input:
72

Output:
3

Explanation:
N=72. The numbers that divide 72 are

1, 2, 3, 4, 6, 8, 9, 12, 18, 24, 36, 72

1 is not considered square free. 4, 9 and 36 are perfect squares, and 8,12,18,24 and 72 are divisible by one of them. Hence only 2, 3 and 6 are square free. (It is easily seen that none of them are divisible by a perfect square). The result is 3.

---

2) Given an array A of size N where, 1<= N <= 10^5. The task is to find the OR of all possible sub-arrays of A and then the OR of all these results.

Examples:
Input : 1 4 6
Output : 7
All possible subarrays are
{1}, {1, 4}, {4, 6} and {1, 4, 6}
ORs of these subarrays are 1, 5, 6
and 7. OR of these ORs is 7.
Input : 10 100 1000
Output : 1006
(Note: This question can also be asked in this way: after finding ORs of subarrays just return all distinct values of ORs)

---

3)Given an array and an integer K, find the maximum for each and every contiguous subarray of size K.

Examples :
Input: arr[] = {1, 2, 3, 1, 4, 5}, K = 3
Output: 3 3 4 5
Explanation: Maximum of 1, 2, 3 is 3
Maximum of 2, 3, 1 is 3
Maximum of 3, 1, 4 is 4
Maximum of 1, 4, 5 is 5

Input: arr[] = {8, 5, 10, 7, 9, 4, 15, 12, 90, 13}, K = 4
Output: 10 10 10 15 15 90 90
Explanation: Maximum of first 4 elements is 10, similarly for next 4 elements (i.e from index 1 to 4) is 10, So the sequence generated is 10 10 10 15 15 90 90

Input: arr[] = {20, 10, 30}, K = 1
Output: 20 10 30

---

4)Given a string s containing only three types of characters: '(', ')' and '*', return true if s is valid.
The following rules define a valid string:
Any left parenthesis '(' must have a corresponding right parenthesis ')'.
Any right parenthesis ')' must have a corresponding left parenthesis '('.
Left parenthesis '(' must go before the corresponding right parenthesis ')'.

Example 1:
Input: s = "()"
Output: true

Example 2:
Input: s = "(())"
Output: true

Example 3:
Input: s = ")("
Output: false

Example 4:

Input: s = "()("
Output: false

Constraints:
1 <= s.length <= 104
s consists of parentheses only '()'.

---

5) Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

Open brackets must be closed by the same type of brackets.
Open brackets must be closed in the correct order.
Every close bracket has a corresponding open bracket of the same type.

Example 1:
Input: s = "()"
Output: true

Example 2:
Input: s = "()[]{}"
Output: true

Example 3:
Input: s = "(]"
Output: false

Example 4:
Input: s = "([])"
Output: true

Constraints:
1 <= s.length <= 104
s consists of parentheses only '()[]{}'.

Example 1:
Input: s =”()”
Output: true

Example 2:
Input: s = “()[]{}”
Output: true

Example 3:
Input: s = ”(]”
Output: false

---

6) Cyclic Palindrome.
Description:
A string is said to be palindrome, if it reads the same from both the ends. Given a string S, you are allowed to perform cyclic shifts. More formally, you can pick any one character from any end (head or tail) and you can append that character at the other end. For example, if the string is "abc", then if we do a shift using the character at head position then the string becomes "bca". Similarly, if we do the shift using the character at the tail then the input string becomes "cab". Your task is to find out the minimum number of shifts needed to make the given string, a palindrome. In case, we can't convert the string to palindrome then print -1

Input Format:
First line starts with T i.e. number of test cases, and then T lines will follow each containing a string "S".

Output Format:
Print the minimum number of cyclic shifts for each string if it can be made a palindrome, else -1.

Sample Input:
```
4
abbb
aaabb
aabb
abc
```

Sample Output:
```
-1
1
1
-1
```

Explanation:
For Test Case 2 (aaabb):

Shift the character at the tail to the head and the result will be "baaab", which is a palindrome. This is an operation which requires minimum number of shifts to make the given string a palindrome.

For Test Case 3 (aabb):

One way to convert the given string to palindrome is, shift the character at the head to the tail, and the result will be "abba", which is a palindrome. Another way is to shift the character at the tail to the head, and the result will be "baab", which is also a palindrome. Both require only one shift.

---

7)One person hands over the list of digits to Mr. String, But Mr. String understands only strings. Within strings also he understands only vowels. Mr. String needs your help to find the total number of pairs which add up to a certain digit D.The rules to calculate digit D are as follow :-
Take all digits and convert them into their textual representation.
Next, sum up the number of vowels i.e. {a, e, i, o, u} from all textual representations.
This sum is digit D
Now, once digit D is known, find out all unordered pairs of numbers in input whose sum is equal to D. Refer example section for better understanding.

Constraints
1 <= N <= 100
1 <= value of each element in second line of input <= 100
Number 100, if and when it appears in input should be converted to textual representation as hundred and not as one hundred. Hence number of vowels in number 100 should be 2 and not 4

Input
First line contains an integer N which represents number of elements to be processed as input
Second line contains N numbers separated by space

Output
Lower case representation of textual representation of number of pairs in input that sum up to digit D
Note: – (If the count exceeds 100 print "greater 100")
Examples

Input : 5
1 2 3 4 5
Output : one
Input : 3
7 4 2
Output : zero

---

8)Given an integer array of coins[ ] of size N representing different types of denominations and an integer sum, the task is to count all combinations of coins to make a given value sum.
Note: Assume that you have an infinite supply of each type of coin.

Examples:

Input: sum = 4, coins[] = {1,2,3}
Output: 4
Explanation: there are four solutions: {1, 1, 1, 1}, {1, 1, 2}, {2, 2} and {1, 3}

Input: sum = 10, coins[] = {2, 5, 3, 6}
Output: 5
Explanation: There are five solutions:
{2,2,2,2,2}, {2,2,3,3}, {2,2,6}, {2,3,5} and {5,5}

Input: sum = 10, coins[] = {10}
Output: 1
Explanation: The only is to pick 1 coin of value 10.

Input: sum = 5, coins[] = {4}
Output: 0
Explanation: We cannot make sum 5 with the given coins

7)Given two strings text1 and text2, return the length of their longest common subsequence. If there is no common subsequence, return 0.

A subsequence of a string is a new string generated from the original string with some characters (can be none) deleted without changing the relative order of the remaining characters.

For example, "ace" is a subsequence of "abcde".
A common subsequence of two strings is a subsequence that is common to both strings.

Example 1:
Input: text1 = "abcde", text2 = "ace"
Output: 3
Explanation: The longest common subsequence is "ace" and its length is 3.
Example 2:
Input: text1 = "abc", text2 = "abc"
Output: 3
Explanation: The longest common subsequence is "abc" and its length is 3.

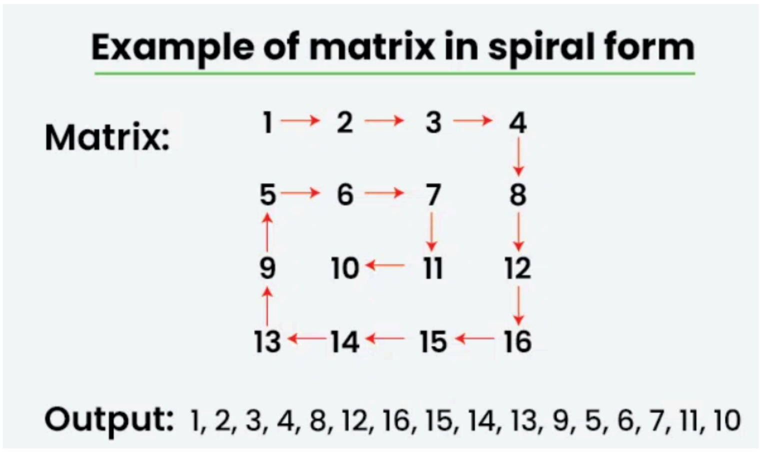
Example 3:
Input: text1 = "abc", text2 = "def"
Output: 0
Explanation: There is no such common subsequence, so the result is 0.

Constraints:
1 <= text1.length, text2.length <= 1000
text1 and text2 consist of only lowercase English characters.

---

9) Given an m x n matrix, the task is to print all elements of the matrix in spiral form.

Examples:
```
Input: matrix = {{1,   2,  3,  4},
                 {5,   6,  7,  8},
                 {9,  10, 11, 12},
                 {13, 14, 15, 16 }}
```


Example of matrix in spiral form
Matrix:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
Output: 1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9, 5, 6, 7, 11, 10

Explanation: The output is matrix in spiral format.

```
Input: matrix = {{1,  2,  3,  4,  5,  6},
                {7,  8,  9, 10, 11, 12},
                {13, 14, 15, 16, 17, 18}}
```

Output: 1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11
Explanation: The output is matrix in spiral format.

---

10) Given an array of n positive integers. We are required to write a program to print the minimum product of k integers of the given array.
Examples:
Input : 198 76 544 123 154 675
        k = 2
Output : 9348
We get minimum product after multiplying
76 and 123.
Input : 11 8 5 7 5 100
       k = 4
Output : 1400

---

11) Anadrome - An Anagramic Palindrome
Description:
An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once. A Palindrome is a word, phrase, number, or other sequence of characters which reads the same backward as forward. Brijesh knows Anagrams and Palindromes. His teacher gives him a string and asks him to find if there exists any anagram of the given string such that it is also a Palindrome. If so then print Yes, otherwise print No.

Input Format
The first line contains t, which represents number of strings. The next t lines contain strings.

Constraints
1<=t<=5
1<=x<=10000 where x denotes string length
Strings contain only lowercase english alphabets.

Output Format
For each string print Yes if there exists an Anagram that is a Palindrome, else print No, in a new line

Sample Input:

```
3
daamm
rfeer
motor
```

Sample Output:

Yes
Yes
No

Explanation:
The string daamm has an anagram --> madam which is a palindrome. So print Yes.
The string rfeer has an anagram --> refer which is a palindrome. So print Yes.
The string motor has no anagram which is also a palindrome. So print No.

---

12) Write an algorithm to determine if a number n is happy.
A happy number is a number defined by the following process:
Starting with any positive integer, replace the number by the sum of the squares of its digits.
Repeat the process until the number equals 1 (where it will stay), or it loops endlessly in a cycle which does not include 1.
Those numbers for which this process ends in 1 are happy.
Return true if n is a happy number, and false if not.

Example 1:
Input: n = 19
Output: true
Explanation:
12 + 92 = 82
82 + 22 = 68
62 + 82 = 100
12 + 02 + 02 = 1

Example 2:
Input: n = 2
Output: false

Constraints:
1 <= n <= 231 - 1

______________________XXX_________________________