Contents

Introductory Questions

1. Add Two Numbers
Question: Write a program to add two numbers.

Test Cases:
1. Input: `a = 5, b = 10` | Output: `15`
2. Input: `a = -3, b = 7` | Output: `4`
3. Input: `a = 0, b = 0` | Output: `0`
4. Input: `a = -5, b = -10` | Output: `-15`
5. Input: `a = 100, b = 200` | Output: `300`
6. Input: `a = 1, b = 9999` | Output: `10000`
7. Input: `a = 2147483647, b = 1` | Output: `2147483648`
8. Input: `a = -2147483648, b = -1` | Output: `-2147483649`
9. Input: `a = 123, b = 456` | Output: `579`
10. Input: `a = 789, b = 321` | Output: `1110`

2. Calculate Simple Interest
Question: Write a program to calculate simple interest.

Test Cases:
1. Input: `P = 1000, R = 5, T = 2` | Output: `100`
2. Input: `P = 2000, R = 3.5, T = 5` | Output: `350`
3. Input: `P = 1500, R = 4, T = 3` | Output: `180`
4. Input: `P = 1200, R = 7.5, T = 4` | Output: `360`
5. Input: `P = 0, R = 6, T = 2` | Output: `0`
6. Input: `P = 5000, R = 0, T = 5` | Output: `0`
7. Input: `P = 10000, R = 2, T = 1` | Output: `200`
8. Input: `P = 3000, R = 4.5, T = 10` | Output: `1350`
9. Input: `P = 500, R = 3.2, T = 6` | Output: `96`
10. Input: `P = 7500, R = 5.5, T = 3` | Output: `1237.5`

3. Maximum of Three Numbers
Question: Write a program to find the maximum of three numbers.

Test Cases:
1. Input: `a = 5, b = 10, c = 3` | Output: `10`
2. Input: `a = 20, b = 15, c = 25` | Output: `25`
3. Input: `a = -5, b = -10, c = -3` | Output: `-3`
4. Input: `a = 0, b = 0, c = 0` | Output: `0`
5. Input: `a = 100, b = 50, c = 150` | Output: `150`
6. Input: `a = 7, b = 7, c = 7` | Output: `7`
7. Input: `a = -7, b = 0, c = 7` | Output: `7`
8. Input: `a = 25, b = 25, c = 20` | Output: `25`
9. Input: `a = 8, b = 10, c = 10` | Output: `10`
10. Input: `a = -1, b = -1, c = -1` | Output: `-1`

4. Odd or Even
Question: Write a program to check if a number is odd or even.

Test Cases:
1. Input: `n = 2` | Output: `Even`
2. Input: `n = 3` | Output: `Odd`
3. Input: `n = 0` | Output: `Even`
4. Input: `n = -5` | Output: `Odd`
5. Input: `n = -8` | Output: `Even`
6. Input: `n = 1` | Output: `Odd`
7. Input: `n = 16` | Output: `Even`
8. Input: `n = 13` | Output: `Odd`
9. Input: `n = -12` | Output: `Even`
10. Input: `n = 100` | Output: `Even`

5. Grade Card
Question: Write a program to assign grades based on marks.

Test Cases:
1. Input: `marks = 85` | Output: `Grade A`
2. Input: `marks = 70` | Output: `Grade B`
3. Input: `marks = 55` | Output: `Grade C`
4. Input: `marks = 40` | Output: `Grade D`
5. Input: `marks = 30` | Output: `Grade F`
6. Input: `marks = 90` | Output: `Grade A`
7. Input: `marks = 75` | Output: `Grade B`
8. Input: `marks = 50` | Output: `Grade C`
9. Input: `marks = 65` | Output: `Grade B`
10. Input: `marks = 80` | Output: `Grade A`

6. Reverse Digits of a Number
Question: Write a program to reverse the digits of a number.

Test Cases:
1. Input: `num = 123` | Output: `321`
2. Input: `num = 456` | Output: `654`
3. Input: `num = 789` | Output: `987`
4. Input: `num = 100` | Output: `1`
5. Input: `num = 0` | Output: `0`
6. Input: `num = 908` | Output: `809`
7. Input: `num = 77` | Output: `77`
8. Input: `num = 9` | Output: `9`
9. Input: `num = 4321` | Output: `1234`
10. Input: `num = 876` | Output: `678`

7. Replace One Digit with Another
Question: Write a program to replace one digit with another in a number.

Test Cases:
1. Input: `num = 746, d1 = 7, d2 = 8` | Output: `846`
2. Input: `num = 123, d1 = 2, d2 = 5` | Output: `153`
3. Input: `num = 987, d1 = 9, d2 = 1` | Output: `187`
4. Input: `num = 555, d1 = 5, d2 = 3` | Output: `333`
5. Input: `num = 404, d1 = 4, d2 = 9` | Output: `909`
6. Input: `num = 800, d1 = 0, d2 = 2` | Output: `822`
7. Input: `num = 777, d1 = 7, d2 = 0` | Output: `000`
8. Input: `num = 123456, d1 = 4, d2 = 9` | Output: `129956`
9. Input: `num = 101010, d1 = 1, d2 = 2` | Output: `202020`
10. Input: `num = 246, d1 = 4, d2 = 7` | Output: `276`

8. Armstrong Numbers
Question: Write a program to check if a number is an Armstrong number.

Test Cases:
1. Input: `num = 153` | Output: `Yes`
2. Input: `num = 370` | Output: `Yes`
3. Input: `num = 9474` | Output: `Yes`
4. Input: `num = 9475` | Output: `No`
5. Input: `num = 0` | Output: `Yes`
6. Input: `num = 1` | Output: `Yes`
7. Input: `num = 10` | Output: `No`
8. Input: `num = 407` | Output: `Yes`
9. Input: `num = 1634` | Output: `Yes`
10. Input: `num = 9476` | Output: `No`

9. N-th Fibonacci Number
Question: Write a program to find the N-th Fibonacci number.

Test Cases:
1. Input: `n = 1` | Output: `0`
2. Input: `n = 2` | Output: `1`
3. Input: `n = 3` | Output: `1`
4. Input: `n = 4` | Output: `

2`
5. Input: `n = 5` | Output: `3`
6. Input: `n = 6` | Output: `5`
7. Input: `n = 7` | Output: `8`
8. Input: `n = 8` | Output: `13`
9. Input: `n = 9` | Output: `21`
10. Input: `n = 10` | Output: `34`

10. Check if a Number is Fibonacci
Question: Write a program to check if a number is a Fibonacci number.

Test Cases:
1. Input: `num = 5` | Output: `Yes`
2. Input: `num = 7` | Output: `No`
3. Input: `num = 8` | Output: `Yes`
4. Input: `num = 9` | Output: `No`
5. Input: `num = 13` | Output: `Yes`
6. Input: `num = 21` | Output: `Yes`
7. Input: `num = 34` | Output: `Yes`
8. Input: `num = 55` | Output: `Yes`
9. Input: `num = 144` | Output: `Yes`
10. Input: `num = 150` | Output: `No`

Array related questions:

1. Program to Find the Largest Element in an Array
Question: Write a program to find the largest element in a given array.

Test Cases:
1. Input: `arr[] = {20, 10, 20, 4, 100}` | Output: `100`
2. Input: `arr[] = {-1, -2, -3, -4, -5}` | Output: `-1`
3. Input: `arr[] = {0, 0, 0, 0}` | Output: `0`
4. Input: `arr[] = {1, 1, 1, 1, 1}` | Output: `1`
5. Input: `arr[] = {99, 88, 77, 66, 100}` | Output: `100`
6. Input: `arr[] = {7, 7, 7, 7, 7}` | Output: `7`
7. Input: `arr[] = {1, 2, 3, 4, 5}` | Output: `5`
8. Input: `arr[] = {100, 200, 300, 400}` | Output: `400`
9. Input: `arr[] = {-10, -20, -30, -40}` | Output: `-10`
10. Input: `arr[] = {2147483647, -2147483648, 0}` | Output: `2147483647`

2. Program to Find the Smallest Element in an Array
Question: Write a program to find the smallest element in a given array.

Test Cases:
1. Input: `arr[] = {20, 10, 20, 4, 100}` | Output: `4`
2. Input: `arr[] = {-1, -2, -3, -4, -5}` | Output: `-5`
3. Input: `arr[] = {0, 0, 0, 0}` | Output: `0`
4. Input: `arr[] = {1, 1, 1, 1, 1}` | Output: `1`
5. Input: `arr[] = {99, 88, 77, 66, 100}` | Output: `66`
6. Input: `arr[] = {7, 7, 7, 7, 7}` | Output: `7`
7. Input: `arr[] = {1, 2, 3, 4, 5}` | Output: `1`
8. Input: `arr[] = {100, 200, 300, 400}` | Output: `100`
9. Input: `arr[] = {-10, -20, -30, -40}` | Output: `-40`
10. Input: `arr[] = {2147483647, -2147483648, 0}` | Output: `-2147483648`

3. Swap the First and Last Elements of an Array
Question: Write a program to swap the first and last elements of an array.

Test Cases:
1. Input: `arr[] = {20, 30, 40}` | Output: `{40, 30, 20}`
2. Input: `arr[] = {1, 2, 3, 4, 5}` | Output: `{5, 2, 3, 4, 1}`
3. Input: `arr[] = {10, 9, 8, 7}` | Output: `{7, 9, 8, 10}`
4. Input: `arr[] = {-1, -2, -3, -4}` | Output: `{-4, -2, -3, -1}`
5. Input: `arr[] = {100}` | Output: `{100}`
6. Input: `arr[] = {1, 1, 1, 1}` | Output: `{1, 1, 1, 1}`
7. Input: `arr[] = {50, 60, 70, 80}` | Output: `{80, 60, 70, 50}`
8. Input: `arr[] = {5, 6, 7, 8, 9}` | Output: `{9, 6, 7, 8, 5}`
9. Input: `arr[] = {0, 0, 0, 0}` | Output: `{0, 0, 0, 0}`
10. Input: `arr[] = {123, 456, 789}` | Output: `{789, 456, 123}`

4. Reverse an Array Using a Loop (In-place)
Question: Write a program to reverse an array in place using a loop.

Test Cases:
1. Input: `arr[] = {1, 2, 3, 4, 5, 6}` | Output: `{6, 5, 4, 3, 2, 1}`
2. Input: `arr[] = {9, 8, 7, 6, 5}` | Output: `{5, 6, 7, 8, 9}`
3. Input: `arr[] = {10, 20, 30, 40}` | Output: `{40, 30, 20, 10}`
4. Input: `arr[] = {-1, -2, -3, -4}` | Output: `{-4, -3, -2, -1}`
5. Input: `arr[] = {1}` | Output: `{1}`
6. Input: `arr[] = {7, 7, 7, 7}` | Output: `{7, 7, 7, 7}`
7. Input: `arr[] = {50, 60, 70, 80}` | Output: `{80, 70, 60, 50}`
8. Input: `arr[] = {2, 4, 6, 8}` | Output: `{8, 6, 4, 2}`
9. Input: `arr[] = {100, 200, 300}` | Output: `{300, 200, 100}`
10. Input: `arr[] = {0, 0, 0, 0}` | Output: `{0, 0, 0, 0}`

5. Reverse an Array in the Given Range
Question: Write a program to reverse an array within a given range `[start, end]`.

Test Cases:
1. Input: `arr[] = {1, 2, 3, 4, 5}, start = 1, end = 3` | Output: `{1, 4, 3, 2, 5}`
2. Input: `arr[] = {10, 20, 30, 40, 50}, start = 0, end = 4` | Output: `{50, 40, 30, 20, 10}`
3. Input: `arr[] = {7, 8, 9, 10, 11}, start = 2, end = 4` | Output: `{7, 8, 11, 10, 9}`
4. Input: `arr[] = {-5, -6, -7, -8}, start = 0, end = 2` | Output: `{-7, -6, -5, -8}`
5. Input: `arr[] = {5, 4, 3, 2, 1}, start = 1, end = 3` | Output: `{5, 2, 3, 4, 1}`
6. Input: `arr[] = {6, 7, 8, 9}, start = 1, end = 2` | Output: `{6, 8, 7, 9}`
7. Input: `arr[] = {10, 20, 30, 40, 50}, start = 1, end = 3` | Output: `{10, 40, 30, 20, 50}`
8. Input: `arr[] = {100, 200, 300}, start = 0, end = 1` | Output: `{200, 100, 300}`
9. Input: `arr[] = {1, 2, 3, 4, 5}, start = 2, end = 4` | Output: `{1, 2, 5, 4, 3}`
10. Input: `arr[] = {7, 7, 7, 7}, start = 0, end = 3` | Output: `{7, 7, 7, 7}`

6. Rotate an Array
Question: Write a program to rotate an array to the right by `k` steps.

Test Cases:
1. Input: `arr[] = {1, 2, 3, 4, 5}, k = 2` | Output: `{4, 5, 1, 2, 3}`
2. Input: `arr[] = {7, 8, 9, 10}, k = 1` | Output: `{

10, 7, 8, 9}`
3. Input: `arr[] = {3, 4, 5, 6}, k = 3` | Output: `{4, 5, 6, 3}`
4. Input: `arr[] = {10, 20, 30, 40}, k = 4` | Output: `{10, 20, 30, 40}`
5. Input: `arr[] = {100, 200, 300}, k = 2` | Output: `{200, 300, 100}`
6. Input: `arr[] = {-1, -2, -3, -4}, k = 1` | Output: `{-4, -1, -2, -3}`
7. Input: `arr[] = {50, 60, 70}, k = 3` | Output: `{50, 60, 70}`
8. Input: `arr[] = {0, 0, 0}, k = 1` | Output: `{0, 0, 0}`
9. Input: `arr[] = {5, 10, 15, 20, 25}, k = 3` | Output: `{15, 20, 25, 5, 10}`

10. Input: `arr[] = {8, 9, 10}, k = 1` | Output: `{10, 8, 9}`

7. Linear Search
Question: Write a program to implement linear search in an array.

Test Cases:
1. Input: `arr[] = {1, 2, 3, 4, 5}, key = 3` | Output: `2`
2. Input: `arr[] = {10, 20, 30, 40}, key = 25` | Output: `-1`
3. Input: `arr[] = {7, 8, 9}, key = 7` | Output: `0`
4. Input: `arr[] = {-5, -6, -7}, key = -7` | Output: `2`
5. Input: `arr[] = {100, 200, 300}, key = 300` | Output: `2`
6. Input: `arr[] = {1, 1, 1, 1}, key = 1` | Output: `0`
7. Input: `arr[] = {50, 60, 70}, key = 60` | Output: `1`
8. Input: `arr[] = {3, 6, 9}, key = 9` | Output: `2`
9. Input: `arr[] = {10, 20, 30, 40}, key = 10` | Output: `0`
10. Input: `arr[] = {7, 8, 9}, key = 10` | Output: `-1`

8. Find First and Last Positions of an Element in an Array
Question: Write a program to find the first and last positions of a given element in an array.

Test Cases:
1. Input: `arr[] = {1, 2, 2, 2, 3}, key = 2` | Output: `1, 3`
2. Input: `arr[] = {5, 6, 7, 8, 9}, key = 7` | Output: `2, 2`
3. Input: `arr[] = {10, 20, 30, 30, 30}, key = 30` | Output: `2, 4`
4. Input: `arr[] = {-5, -4, -4, -3}, key = -4` | Output: `1, 2`
5. Input: `arr[] = {100, 100, 100}, key = 100` | Output: `0, 2`
6. Input: `arr[] = {1, 2, 3, 4, 5}, key = 6` | Output: `-1, -1`
7. Input: `arr[] = {10, 20, 20, 20, 30}, key = 20` | Output: `1, 3`
8. Input: `arr[] = {7, 8, 8, 8, 9}, key = 8` | Output: `1, 3`
9. Input: `arr[] = {50, 60, 70, 70}, key = 70` | Output: `2, 3`
10. Input: `arr[] = {3, 4, 5, 6, 7}, key = 5` | Output: `2, 2`

9. Program to Find the Second Largest Element in an Array
Question: Write a program to find the second largest element in a given array.

Test Cases:
1. Input: `arr[] = {1, 2, 3, 4, 5}` | Output: `4`
2. Input: `arr[] = {10, 20, 30, 40}` | Output: `30`
3. Input: `arr[] = {7, 7, 7, 7}` | Output: `7`
4. Input: `arr[] = {100, 50, 20, 10}` | Output: `50`
5. Input: `arr[] = {-1, -2, -3, -4}` | Output: `-2`
6. Input: `arr[] = {5, 10, 15, 20}` | Output: `15`
7. Input: `arr[] = {9, 8, 7, 6}` | Output: `8`
8. Input: `arr[] = {50, 40, 30, 20}` | Output: `40`
9. Input: `arr[] = {1, 2, 2, 3, 3}` | Output: `3`
10. Input: `arr[] = {99, 100, 101, 102}` | Output: `101`

10. Print All Indices of a Given Value in an Array
Question: Write a program to print all indices of a given value in an array.

Test Cases:
1. Input: `arr[] = {1, 2, 2, 3, 2}, key = 2` | Output: `1, 2, 4`
2. Input: `arr[] = {5, 6, 7, 7, 7}, key = 7` | Output: `2, 3, 4`
3. Input: `arr[] = {10, 10, 10, 20}, key = 10` | Output: `0, 1, 2`
4. Input: `arr[] = {-1, -1, -2, -1}, key = -1` | Output: `0, 1, 3`
5. Input: `arr[] = {3, 4, 4, 4, 5}, key = 4` | Output: `1, 2, 3`
6. Input: `arr[] = {7, 8, 9, 9}, key = 9` | Output: `2, 3`
7. Input: `arr[] = {50, 50, 60, 60}, key = 50` | Output: `0, 1`
8. Input: `arr[] = {1, 1, 1, 1}, key = 1` | Output: `0, 1, 2, 3`
9. Input: `arr[] = {100, 200, 200, 300}, key = 200` | Output: `1, 2`
10. Input: `arr[] = {3, 3, 3, 3}, key = 3` | Output: `0, 1, 2, 3`

String-related questions

1. Reverse a String
Question: Write a program to reverse the characters in a given string.

Test Cases:
1. Input: `"hello"` | Output: `"olleh"`
2. Input: `"world"` | Output: `"dlrow"`
3. Input: `"Java"` | Output: `"avaJ"`
4. Input: `"OpenAI"` | Output: `"IAnepO"`
5. Input: `"racecar"` | Output: `"racecar"`
6. Input: `"12345"` | Output: `"54321"`
7. Input: `" "` | Output: `" "`
8. Input: `"a"` | Output: `"a"`
9. Input: `"!@#$%"` | Output: `"%$#@!"`
10. Input: `"madam"` | Output: `"madam"`

2. Check for Palindrome
Question: Write a program to determine if a string reads the same backward as forward.

Test Cases:
1. Input: `"madam"` | Output: `True`
2. Input: `"hello"` | Output: `False`
3. Input: `"racecar"` | Output: `True`
4. Input: `"palindrome"` | Output: `False`
5. Input: `"12321"` | Output: `True`
6. Input: `"OpenAI"` | Output: `False`
7. Input: `"noon"` | Output: `True`
8. Input: `"a"` | Output: `True`
9. Input: `" "` | Output: `True`
10. Input: `"level"` | Output: `True`

3. Find First Non-Repeating Character
Question: Write a program to identify the first character that does not repeat in a string.

Test Cases:
1. Input: `"swiss"` | Output: `"w"`
2. Input: `"hello"` | Output: `"h"`
3. Input: `"racecar"` | Output: `"e"`
4. Input: `"aabbcc"` | Output: `None`
5. Input: `"Java"` | Output: `"J"`
6. Input: `"aabbc"` | Output: `"c"`
7. Input: `"abcdef"` | Output: `"a"`
8. Input: `"popcorn"` | Output: `"p"`
9. Input: `"banana"` | Output: `"b"`
10. Input: `"success"` | Output: `"u"`

4. Count the Occurrence of Each Character
Question: Write a program to count how many times each character appears in a string.

Test Cases:
1. Input: `"hello"` | Output: `{'h': 1, 'e': 1, 'l': 2, 'o': 1}`
2. Input: `"mississippi"` | Output: `{'m': 1, 'i': 4, 's': 4, 'p': 2}`
3. Input: `"Java"` | Output: `{'J': 1, 'a': 2, 'v': 1}`
4. Input: `"abracadabra"` | Output: `{'a': 5, 'b': 2, 'r': 2, 'c': 1, 'd': 1}`
5. Input: `"racecar"` | Output: `{'r': 2, 'a': 2, 'c': 2, 'e': 1}`
6. Input: `"12345"` | Output: `{'1': 1, '2': 1, '3': 1, '4': 1, '5': 1}`
7. Input: `"OpenAI"` | Output: `{'O': 1, 'p': 1, 'e': 1, 'n': 1, 'A': 1, 'I': 1}`
8. Input: `"abcdabcd"` | Output: `{'a': 2, 'b': 2, 'c': 2, 'd': 2}`
9. Input: `"aabbcc"` | Output: `{'a': 2, 'b': 2, 'c': 2}`
10. Input: `"banana"` | Output: `{'b': 1, 'a': 3, 'n': 2}`

5. Anagram Check
Question: Write a program to verify if two strings contain the same characters in a different order.

Test Cases:
1. Input: `"listen", "silent"` | Output: `True`
2. Input: `"triangle", "integral"` | Output: `True`
3. Input: `"apple", "pale"` | Output: `False`
4. Input: `"rat", "tar"` | Output: `True`
5. Input: `"dusty", "study"` | Output: `True`
6. Input: `"night", "thing"` | Output: `True`
7. Input: `"evil", "vile"` | Output: `True`
8. Input: `"fluster", "restful"` | Output: `True`
9. Input: `"hello", "world"` | Output: `False`
10. Input: `"loop", "pool"` | Output: `True`

6. String Rotation
Question: Write a program to check if one string is a rotated version of another.

Test Cases:
1. Input: `"abcd", "dabc"` | Output: `True`
2. Input: `"hello", "lohel"` | Output: `True`
3. Input: `"rotation", "tationro"` | Output: `True`
4. Input: `"apple", "leapp"` | Output: `True`
5. Input: `"OpenAI", "AIpenO"` | Output: `True`
6. Input: `"java", "avaj"` | Output: `True`
7. Input: `"world", "dlrow"` | Output: `False`
8. Input: `"example", "pleexam"` | Output: `True`
9. Input: `"abcdef", "efabcd"` | Output: `True`
10. Input: `"string", "gnirts"` | Output: `False`

7. Remove Duplicates
Question: Write a program to eliminate duplicate characters from a string.

Test Cases:
1. Input: `"hello"` | Output: `"helo"`
2. Input: `"mississippi"` | Output: `"misp"`
3. Input: `"Java"` | Output: `"Jav"`
4. Input: `"abracadabra"` | Output: `"abrcd"`
5. Input: `"racecar"` | Output: `"race"`
6. Input: `"banana"` | Output: `"ban"`
7. Input: `"OpenAI"` | Output: `"OpenAI"`
8. Input: `"abcabc"` | Output: `"abc"`
9. Input: `"112233"` | Output: `"123"`
10. Input: `"success"` | Output: `"suce"`

8. Longest Common Prefix
Question: Write a program to find the longest common starting substring among an array of strings.

Test Cases:
1. Input: `["flower", "flow", "flight"]` | Output: `"fl"`
2. Input: `["dog", "racecar", "car"]` | Output: `""`
3. Input: `["interspecies", "interstellar", "interstate"]` | Output: `"inters"`
4. Input: `["throne", "throne"]` | Output: `"throne"`
5. Input: `["prefix", "prelude", "prevent"]` | Output: `"pre"`
6. Input: `["apple", "ape", "april"]` | Output: `"ap"`
7. Input: `["java", "javascript", "javelin"]` | Output: `"jav"`
8. Input: `["cat", "car", "carbon"]` | Output: `"ca"`
9. Input: `["hello", "hell", "heaven"]` | Output: `"he"`
10. Input: `["banana", "band", "banner"]` | Output: `"ban"`

9. String Compression
Question: Write a program to compress a string using the counts of repeated characters.

Test Cases:
1. Input: `"aabcccccaaa"` | Output: `"a2b1c5a3"`
2. Input: `"abb"` | Output: `"a1b2"`
3. Input: `"abcd"` | Output: `"a1b1c1d1"`
4. Input: `"aaabbbccc"` | Output: `"a3b3c3"`
5. Input: `"xxxxxxyyyyyy"` | Output: `"x6y6"`
6. Input: `"a"` | Output: `"a1"`
7. Input: `"bbbaaa"` | Output: `"b3a3"`
8. Input: `"opppppppp"` | Output: `"o1p7"`
9. Input: `"mississippi"` | Output: `"m1i1s2i1s2i1p2i1"`
10. Input: `"111222333"` | Output: `"13122333"`

10. String to Integer (atoi)
Question: Write a program to convert a string to an integer, handling edge cases.

Test Cases:
1. Input: `"42"` | Output: `42`
2. Input: `"

  -42"` | Output: `-42`
3. Input: `"4193 with words"` | Output: `4193`
4. Input: `"words and 987"` | Output: `0`
5. Input: `"-91283472332"` | Output: `-2147483648` (Handle Integer Underflow)
6. Input: `"2147483648"` | Output: `2147483647` (Handle Integer Overflow)
7. Input: `"0032"` | Output: `32`
8. Input: `"+1"` | Output: `1`
9. Input: `"   12345   "` | Output: `12345`
10. Input: `"   "` | Output: `0`

11. Check Substring
Question: Write a program to determine if one string is a substring of another.

Test Cases:
1. Input: `"hello", "ell"` | Output: `True`
2. Input: `"world", "or"` | Output: `True`
3. Input: `"OpenAI", "pen"` | Output: `True`
4. Input: `"Java", "python"` | Output: `False`
5. Input: `"abcdef", "def"` | Output: `True`
6. Input: `"abcdef", "gh"` | Output: `False`
7. Input: `"test", "testing"` | Output: `False`
8. Input: `"rotation", "tat"` | Output: `True`
9. Input: `"string", "tri"` | Output: `True`
10. Input: `"abcdefgh", "hgf"` | Output: `False`

12. Valid Parentheses
Question: Write a program to check if a string of parentheses is balanced.

Test Cases:
1. Input: `"()"` | Output: `True`
2. Input: `"()[]{}"` | Output: `True`
3. Input: `"(]"` | Output: `False`
4. Input: `"([)]"` | Output: `False`
5. Input: `"{" | Output: `False`
6. Input: `"({[]})"` | Output: `True`
7. Input: `"((()))"` | Output: `True`
8. Input: `"{{{{"` | Output: `False`
9. Input: `"{}[]()"` | Output: `True`

10. Input: `"({})"` | Output: `False`

13. Permutations of a String
Question: Write a program to generate all permutations of a given string.

Test Cases:
1. Input: `"abc"` | Output: `["abc", "acb", "bac", "bca", "cab", "cba"]`
2. Input: `"123"` | Output: `["123", "132", "213", "231", "312", "321"]`
3. Input: `"ab"` | Output: `["ab", "ba"]`
4. Input: `"a"` | Output: `["a"]`
5. Input: `"xy"` | Output: `["xy", "yx"]`
6. Input: `"abcd"` | Output: `["abcd", "abdc", "acbd", "acdb", "adbc", "adcb", "bacd", "badc", "bcad", "bcda", "bdac", "bdca", "cabd", "cadb", "cbad", "cbda", "cdab", "cdba", "dabc", "dacb", "dbac", "dbca", "dcab", "dcba"]`
7. Input: `"no"` | Output: `["no", "on"]`
8. Input: `"xyz"` | Output: `["xyz", "xzy", "yxz", "yzx", "zxy", "zyx"]`
9. Input: `"aa"` | Output: `["aa", "aa"]`
10. Input: `"cat"` | Output: `["cat", "cta", "act", "atc", "tca", "tac"]`

14. Count and Say Sequence
Question: Write a program to generate the nth term in the count-and-say sequence.

Test Cases:
1. Input: `1` | Output: `"1"`
2. Input: `2` | Output: `"11"`
3. Input: `3` | Output: `"21"`
4. Input: `4` | Output: `"1211"`
5. Input: `5` | Output: `"111221"`
6. Input: `6` | Output: `"312211"`
7. Input: `7` | Output: `"13112221"`
8. Input: `8` | Output: `"1113213211"`
9. Input: `9` | Output: `"31131211131221"`
10. Input: `10` | Output: `"13211311123113112211"`

15. Longest Palindromic Substring
Question: Write a program to find the longest palindromic substring in a given string.

Test Cases:
1. Input: `"babad"` | Output: `"bab"` (or `"aba"`)
2. Input: `"cbbd"` | Output: `"bb"`
3. Input: `"racecar"` | Output: `"racecar"`
4. Input: `"a"` | Output: `"a"`
5. Input: `"forgeeksskeegfor"` | Output: `"geeksskeeg"`
6. Input: `"madam"` | Output: `"madam"`
7. Input: `"banana"` | Output: `"anana"`
8. Input: `"abcd"` | Output: `"a"` (or `"b"`, `"c"`, `"d"`)

9. Input: `"noon"` | Output: `"noon"`
10. Input: `"abracadabra"` | Output: `"aca"`

Pattern Based Questions

PATTERN

1. N=5

1

1 2

1 2 3

1 2 3 4

1 2 3 4 5

2. N=5

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

3. N=5

1

2 3

4 5 6

7 8 9 10

11 12 13 14 15

4. N=5

1

0 1

1 0 1

0 1 0 1

1 0 1 0 1

5. N=7

****

****

****

****

****

****

****

6. N=5

*

**

***

****

*****

7. N=6

******

*****

****

***

**

*

8. N=5

```
    *
   **
  ***
 ****
*****
```

9. N=5

```
        1
      2 0 2
    3 0 0 0 3
  4 0 0 0 0 0 4
5 0 0 0 0 0 0 0 5
```

10. N=5

```
*             *
* *           * *
* * *       * * *
* * * *   * * * *
* * * * * * * * *
```

11. N=5

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

12. N=5

```
*           *

**         **

***       ***

**** ****

*********

**** ****

***       ***

**         **

*           *
```

13. N=5

14. N=5

15. N=6

```
      ******

     ******

    ******

   ******

  ******

******
```

Function related Question

/*A Boston number is a composite number, the sum of whose digits is the sum of the digits of its prime factors obtained as a result of prime factorization (excluding 1). The first few such numbers are 4,22 ,27 ,58 ,85 ,94 and 121 . For example, 378 = 2 × 3 × 3 × 3 × 7 is a Boston number since 3 + 7 + 8 = 2 + 3 + 3 + 3 + 7. Write a program to check whether a given integer is a Boston number.

Input Format

There will be only one line of input:N , the number which needs to be checked.

Constraints

1 < N < 2,147,483,647 (max value of an integer of the size of 4 bytes)

Output Format

1 if the number is a Boston number. 0 if the number is a not Boston number.

Sample Input

378

Sample Output

1

Explanation

Self Explanatory

*/