Complete the function $solveMeFirst$ to compute the sum of two integers.

Example

$a = 7$

$b = 3$

Return 10.

Function Description

Complete the $solveMeFirst$ function with the following parameters:

- int a: the first value
- int b: the second value

Returns

- int: the sum of a and b

Constraints

$1 \leq a, b \leq 1000$

Sample Input

```
a = 2
b = 3
```

Sample Output

```
5
```

Explanation

Change Theme Language C

```
1  #include <stdio.h>
2
3  int main(void) {
4      int a, b;
5      if (scanf("%d %d", &a, &b) == 2) {
6          printf("%d\n", a + b);
7      }
8      return 0;
9  }
10
11
12
```

Upload Code as File Test against custom input Run Code

Problem Submissions Leaderboard Discussions

Given an array of integers, find the sum of its elements.

For example, if the array $ar = [1, 2, 3]$. $1 + 2 + 3 = 6$, so return 6.

Function Description

Complete the *simpleArraySum* function with the following parameter(s):

- $ar[n]$: an array of integers

Returns

- *int*: the sum of the array elements

Input Format

The first line contains an integer, n, denoting the size of the array.

The second line contains n space-separated integers representing the array's elements.

Constraints

$0 < n, ar[i] \leq 1000$

Sample Input

```
STDIN          Function
-----          --------
6              ar[] size n = 6
1 2 3 4 10 11  ar = [1, 2, 3, 4, 10, 11]
```

Sample Output

```
31
```

Change Theme

```
1  #include <stdio.h>
2
3  int main() {
4      int n, i, sum = 0;
5      scanf("%d", &n);
6
7      int ar[n];
8
9      for(i = 0; i < n; i++)
10         scanf("%d", &ar[i]);
11
12     for(i = 0; i < n; i++)
13         sum += ar[i];
14
15     printf("%d", sum);
16
17     return 0;
18 }
19
```

Upload Code as File Test against custom input

Alice and Bob each created one problem for HackerRank. A reviewer rates the two challenges, awarding points on a scale from 1 to 100 for three categories: problem clarity, originality, and difficulty.

The rating for Alice's challenge is the triplet a = (a[0], a[1], a[2]), and the rating for Bob's challenge is the triplet b = (b[0], b[1], b[2]).

The task is to calculate their comparison points by comparing each category:

- If a[i] > b[i], then Alice is awarded 1 point.
- If a[i] < b[i], then Bob is awarded 1 point.
- If a[i] = b[i], then neither person receives a point.

Example

a = [1, 2, 3]

b = [3, 2, 1]

- For elements *0*, Bob is awarded a point because a[0] < b[0].
- For the equal elements a[1] and b[1], no points are earned.
- Finally, for elements 2, a[2] > b[2] so Alice receives a point.

The return array is [1, 1] with Alice's score first and Bob's second.

Function Description

Complete the function compareTriplets with the following parameter(s):

- int a[3]: Alice's challenge rating
- int b[3]: Bob's challenge rating

Returns

- int[2]: the first element is Alice's score and the second is Bob's score

Change Theme Language C

```c
#include <stdio.h>

int main() {
    int a[3], b[3];
    int alice = 0, bob = 0;

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


    for(int i = 0; i < 3; i++)
        scanf("%d", &b[i]);

    for(int i = 0; i < 3; i++) {
        if(a[i] > b[i])
            alice++;
        else if(a[i] < b[i])
            bob++;
    }

    // Print results
    printf("%d %d", alice, bob);

    return 0;
}
```

Upload Code as File Test against custom input Run Code

Type here to search

In this challenge. you need to calculate and print the sum of elements in an array.

considering that some integers may be very large.

Function Description

Complete the $aVeryBigSum$ function with the following parameter(s):

- $int\ ar[n]$: an array of integers

Return

- $long$: the sum of the array elements

Input Format

The first line of the input consists of an integer n.

The next line contains n space-separated integers contained in the array.

Output Format

Return the integer sum of the elements in the array.

Constraints

$1 \leq n \leq 10$

$0 \leq ar[i] \leq 10^{10}$

Sample Input

```
STDIN                                                    Function
-----                                                    --------
5                                                        arr[] size n = 5
1000000001 1000000002 1000000003 1000000004 1000000005   arr[...]
```

Output

Change Theme Language C

```c
#include <stdio.h>

long aVeryBigSum(int n, long arr[]) {
    long sum = 0;
    for (int i = 0; i < n; i++) {
        sum += arr[i];
    }
    return sum;
}

int main() {
    int n;
    scanf("%d", &n);
    long arr[n];
    for (int i = 0; i < n; i++) {
        scanf("%ld", &arr[i]);
    }
    printf("%ld\n", aVeryBigSum(n, arr));
    return 0;
}
```

Upload Code as File Test against custom input

Submissions Leaderboard Discussions

Type here to search

iven a square matrix, calculate the absolute difference between the sums of its iagonals.

or example, the square matrix arr is shown below:

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

- The left-to-right diagonal = $1 + 5 + 9 = 15$.
- The right-to-left diagonal = $3 + 5 + 9 = 17$.

Their absolute difference is $|15 - 17| = 2$.

Function description

Complete the *diagonalDifference* function with the following parameter:

- $int\ arr[n][m]$: a 2-D array of integers

Return

- int: the absolute difference in sums along the diagonals

Input Format

The first line contains a single integer, n, the number of rows and columns in the square matrix arr.

Each of the next n lines describes a row, $arr[i]$, and consists of n space-separated integers $arr[i][j]$.

Constraints

- $-100 \leq arr[i][j] \leq 100$

Problem Submissions Leaderboard Discussions

Given an array of integers, calculate the ratios of its elements that are ***positive***, ***negative***, and ***zero***. Print the decimal value of each fraction on a new line with 6 places after the decimal.

Note: This challenge introduces precision problems. The test cases are scaled to six decimal places, though answers with absolute error of up to 10^{-4} are acceptable.

Example

$arr = [1, 1, 0, -1, -1]$

There are $n = 5$ elements: two positive, two negative and one zero. Their ratios are $\frac{2}{5} = 0.400000$, $\frac{2}{5} = 0.400000$ and $\frac{1}{5} = 0.200000$. Results are printed as:

```
0.400000
0.400000
0.200000
```

Function Description

Complete the *plusMinus* function with the following parameter(s):

- *int arr[n]*: an array of integers

Print

Print the ratios of positive, negative and zero values in the array. Each value should be printed on a separate line with 6 digits after the decimal. The function should not return a value.

Input Format

The first line contains an integer, n, the size of the array.

Change Theme Language C

```c
#include <stdio.h>

int main() {
    int n, i;
    scanf("%d", &n);

    int arr[n];
    for(i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    int pos = 0, neg = 0, zero = 0;

    for(i = 0; i < n; i++) {
        if(arr[i] > 0)
            pos++;
        else if(arr[i] < 0)
            neg++;
        else
            zero++;
    }

    printf("%.6f\n", (float)pos / n);
    printf("%.6f\n", (float)neg / n);
    printf("%.6f\n", (float)zero / n);
```

Upload Code as File Test against custom input

Type here to search

Staircase detail

This is a staircase of size $n = 4$:

```
   #
  ##
 ###
####
```

Its base and height are both equal to n. It is drawn using # symbols and spaces. **The last line is not preceded by any spaces.**

Write a program that prints a staircase of size n.

Function Description

Complete the *staircase* function with the following parameter(s):

- *int* n: an integer

Print

Print a staircase as described above. No value should be returned.

Note: The last line is not preceded by spaces. All lines are right-aligned.

Input Format

A single integer, n, denoting the size of the staircase.

Constraints

$0 < n \leq 100$.

Sample Input

Change Theme Language C

```c
#include <stdio.h>

int main() {
    int n, row, space, hash;

    scanf("%d", &n);


    for (row = 1; row <= n; row++) {


        for (space = 1; space <= n - row; space++
            printf(" ");
        }


        for (hash = 1; hash <= row; hash++) {
            printf("#");
        }
        printf("\n");
    }

    // Step 6: End of program
    return 0;
}
```

Upload Code as File Test against custom input

Type here to search

Given five positive integers, find the minimum and maximum values that can be calculated by summing exactly four of the five integers. Then print the respective minimum and maximum values as a single line of two space-separated long integers.

Example

$arr = [1, 3, 5, 7, 9]$

The minimum sum is $1 + 3 + 5 + 7 = 16$ and the maximum sum is $3 + 5 + 7 + 9 = 24$. The function prints

```
16 24
```

Function Description

Complete the $miniMaxSum$ function with the following parameter(s):

- $arr[5]$: an array of 5 integers

Print

Print two space-separated integers on one line: the minimum sum and the maximum sum of 4 of 5 elements.No value should be returned.

Note For some languages, like C, C++, and Java, the sums may require that you use a long integer due to their size.

Input Format

A single line of five space-separated integers.

Constraints

$1 \leq arr[i] \leq 10^9$

Change Theme Language C

```c
#include <stdio.h>

int main() {
    long int arr[5], sum = 0;
    long int min, max;
    int i;

    for(i = 0; i < 5; i++) {
        scanf("%ld", &arr[i]);
        sum += arr[i];
    }

    min = arr[0];
    max = arr[0];

    for(i = 1; i < 5; i++) {
        if(arr[i] < min)
            min = arr[i];
        if(arr[i] > max)
            max = arr[i];
    }

    printf("%ld %ld", sum - max, sum - min);

    return 0;
}
```

Upload Code as File Test against custom input

Problem Submissions Leaderboard Discussions

Type here to search Earnings up

n charge of the cake for a child's birthday. It will have one candle for each year of their total age. They will only
o blow out the tallest of the candles. Your task is to count how many candles are the tallest.

$= [4, 4, 1, 3]$

t candles are 4 units high. There are 2 candles with this height, so the function should return 2.

Description

the function *birthdayCakeCandles* with the following parameter(s):

ndles[n]: the candle heights

e number of candles that are tallest

mat

e contains a single integer, n, the size of $candles[]$.

d line contains n space-separated integers, where each integer i describes the height of $candles[i]$.

ts

$\leq 10^5$

$ndles[i] \leq 10^7$

put 0

tput 0

n 0

ghts are $[3, 2, 1, 3]$. The tallest candles are 3 units, and there are 2 of them.

Change The

```c
#include <stdio.h>

int main() {
    int n, i;

    scanf("%d", &n);

    int candles[n];
    for(i = 0; i < n; i++) {
        scanf("%d", &candles[i]);
    }

    int maxHeight = candles[0];
    for(i = 1; i < n; i++) {
        if(candles[i] > maxHeight) {
            maxHeight = candles[i];
        }
    }

    int count = 0;
    for(i = 0; i < n; i++) {
        if(candles[i] == maxHeight) {
            count++;
        }
    }

    // Step 6: Print the number of tallest candles
    printf("%d", count);

    // Step 7: End of program
    return 0;
}

```

Upload Code as File

Test against custom input

here to search

https://www.hackerrank.com/challenges/time-conversion/problem?isFullScreen=true

erRank Prepare Algorithms Warmup Time Conversion

Given a time in 12-hour AM/PM format, convert it to military (24-hour) time.

Note: - 12:00:00AM on a 12-hour clock is 00:00:00 on a 24-hour clock.

12:00:00PM on a 12-hour clock is 12:00:00 on a 24-hour clock.

xample

s = '12:01:00PM'

Return '12:01:00'.

s = '12:01:00AM'

Return '00:01:00'.

nction Description

mplete the *timeConversion* function with the following parameter(s):

string s: a time in 12 hour format

urns

string: the time in 24 hour format

ut Format

ngle string s that represents a time in 12-hour clock format (i.e.: hh:mm:ssAM or mm:ssPM).

straints

ll input times are valid

ple Input 0

7:05:45PM

Change Theme Language C

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

int main() {
    char time12[11];
    scanf("%s", time12);

    int hour, minute, second;
    char period[3]; //


    sscanf(time12, "%2d:%2d:%2d%2s", &hour, &minute, &second, perio

    if (strcmp(period, "AM") == 0) {
        if (hour == 12) hour = 0; // Midnight case
    } else if (strcmp(period, "PM") == 0) {
        if (hour != 12) hour += 12; // Afternoon case
    }

    printf("%02d:%02d:%02d\n", hour, minute, second);

    return 0;
}
```

Upload Code as File Test against custom input Run Code

pe here to search

23°C Mostly cloudy

DELL

https://www.hackerrank.com/challenges/grading/problem?isFullScreen=true

Prepare Algorithms Implementation Grading Students

erLand University has the following grading policy:

ery student receives a *grade* in the inclusive range from 0 to 100.

y *grade* less than 40 is a failing grade.

s a professor at the university and likes to round each student's *grade* according to rules:

he difference between the *grade* and the next multiple of 5 is less than 3, round *de* up to the next multiple of 5.

e value of *grade* is less than 38, no rounding occurs as the result will still be a ng grade.

les

de = 84 round to 85 (85 - 84 is less than 3)

de = 29 do not round (result is less than 38)

de = 57 do not round (60 - 57 is 3 or higher)

e initial value of *grade* for each of Sam's n students, write code to automate the g process.

n Description

e the function *gradingStudents* with the following parameter(s):

rades[*n*]: the grades before rounding

]: the grades after rounding

rmat

ine contains a single integer, n, the number of students.

Change Theme Language C

```c
#include <stdio.h>

int main() {
    int n;
    scanf("%d", &n);
    int grade;

    for (int i = 0; i < n; i++) {
        scanf("%d", &grade);

        if (grade >= 38) {  // Only round if passing
            int nextMultiple = ((grade / 5) + 1) * 5;
            if (nextMultiple - grade < 3)
                grade = nextMultiple;
        }

        printf("%d\n", grade);
    }

    return 0;
}
```

Upload Code as File Test against custom input

ere to search

23°C Mostly cloudy

https://www.hackerrank.com/challenges/apple-and-orange/problem?isFullScreen=true

nk Prepare Algorithms Implementation Apple and Orange

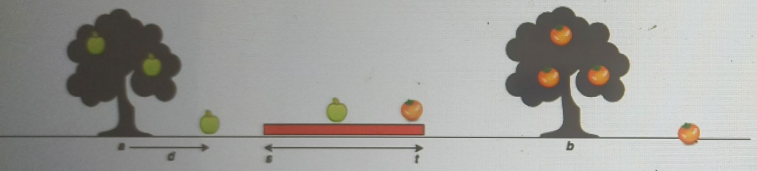
house has an apple tree and an orange tree that yield an abundance of fruit. Using the information given below,
ine the number of apples and oranges that land on Sam's house.

diagram below:

red region denotes the house, where s is the start point, and t is the endpoint. The apple tree is to the left of
house, and the orange tree is to its right.

ume the trees are located on a single point, where the apple tree is at point a, and the orange tree is at point b.

en a fruit falls from its tree, it lands d units of distance from its tree of origin along the x-axis. *A negative value
means the fruit fell d units to the tree's left, and a positive value of d means it falls d units to the tree's right. *



e value of d for m apples and n oranges, determine how many apples and oranges will fall on Sam's house
he inclusive range $[s, t]$)?

mple. Sam's house is between $s = 7$ and $t = 10$. The apple tree is located at $a = 4$ and the orange at $b = 12$.
re $m = 3$ apples and $n = 3$ oranges. Apples are thrown $apples = [2, 3, -4]$ units distance from a, and
$s = [3, -2, -4]$ units distance. Adding each apple distance to the position of the tree, they land at
$4+3, 4+-4] = [6, 7, 0]$. Oranges land at $[12+3, 12+-2, 12+-4] = [15, 10, 8]$. One apple and two
land in the inclusive range $7 - 10$ so we print

n Description

te the countApplesAndOranges function in the editor below. It should print the number of apples and oranges
d on Sam's house, each on a separate line.

pplesAndOranges has the following parameter(s):

Change Theme La

```
#include <stdio.h>

int main() {
    int s, t;
    int a, b;
    int m, n;

    scanf("%d %d", &s, &t);
    scanf("%d %d", &a, &b);
    scanf("%d %d", &m, &n);

    int apples[m], oranges[n];
    for (int i = 0; i < m; i++) scanf("%d", &apples[i]);
    for (int i = 0; i < n; i++) scanf("%d", &oranges[i]);

    int appleCount = 0, orangeCount = 0;

    for (int i = 0; i < m; i++) {
        int pos = a + apples[i];
        if (pos >= s && pos <= t) appleCount++;
    }

    for (int i = 0; i < n; i++) {
        int pos = b + oranges[i];
        if (pos >= s && pos <= t) orangeCount++;
    }

    printf("%d\n%d\n", appleCount, orangeCount);

    return 0;
}
```

Upload Code as File Test against custom input

ype here to search

Change Theme Language C

```c
#include <stdio.h>

int main() {
    int x1, v1, x2, v2;
    scanf("%d %d %d %d", &x1, &v1, &x2, &v2);

    if (v1 == v2) {

        if (x1 == x2)
            printf("YES\n");
        else
            printf("NO\n");
    } else {

        int diff = x2 - x1;
        int speedDiff = v1 - v2;

        if (speedDiff != 0 && diff % speedDiff == 0 && (diff / speedDiff) >=
            printf("YES\n");
        else
            printf("NO\n");
    }

    return 0;
}
```

Line: 27 Col:

Upload Code as File Test against custom input Run Code Submit Code

Forming a Magic Square | Hacker | hr-testcases-us-east-1.s3.amazon

https://www.hackerrank.com/challenges/magic-square-forming/problem?isFullScreen=true

67%

HackerRank | Prepare | Algorithms | Implementation | Forming a Magic Square

Change Theme

We define a magic square to be an $n \times n$ matrix of distinct positive integers from 1 to n^2 where the sum of any row, column, or diagonal of length n is always equal to the same number: the magic constant.

You will be given a 3×3 matrix s of integers in the inclusive range $[1, 9]$. We can convert any digit a to any other digit b in the range $[1, 9]$ at cost of $|a - b|$. Given s, convert it into a magic square at minimal cost. Print this cost on a new line.

Note: The resulting magic square must contain distinct integers in the inclusive range $[1, 9]$.

Example

$s = [[5, 3, 4], [1, 5, 8], [6, 4, 2]]$

The matrix looks like this:

```
5 3 4
1 5 8
6 4 2
```

We can convert it to the following magic square:

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

This took three replacements at a cost of $|5 - 8| + |8 - 9| + |4 - 7| = 7$.

Function Description

Complete the formingMagicSquare function in the editor below.

formingMagicSquare has the following parameter(s):

- int s[3][3]: a 3×3 array of integers

Returns

- int: the minimal total cost of converting the input square to a magic square

Input Format

Each of the 3 lines contains three space-separated integers of row $s[i]$.

Constraints

- $s[i][j] \in [1, 9]$

Sample Input 0

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

```c
#include <stdio.h>
#include <stdlib.h>

int min(int a, int b) {
    return a < b ? a : b;
}

int formingMagicSquare(int s[3][3]) {
    // All possible 3x3 magic squares
    int magicSquares[8][3][3] = {
        {{8,1,6},{3,5,7},{4,9,2}},
        {{6,1,8},{7,5,3},{2,9,4}},
        {{4,9,2},{3,5,7},{8,1,6}},
        {{2,9,4},{7,5,3},{6,1,8}},
        {{8,3,4},{1,5,9},{6,7,2}},
        {{4,3,8},{9,5,1},{2,7,6}},
        {{6,7,2},{1,5,9},{8,3,4}},
        {{2,7,6},{9,5,1},{4,3,8}}
    };

    int minCost = 1000000; // Large number
    for (int k = 0; k < 8; k++) {
        int cost = 0;
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
                cost += abs(s[i][j] - magicSquares[k][i][j]);
        minCost = min(minCost, cost);
    }
    return minCost;
}

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

    printf("%d\n", formingMagicSquare(s));
    return 0;
}
```

Upload Code as File | Test against custom input

Type here to search

23°C Mostly

Picking Numbers | HackerRank

hr-testcases-us-east-1.s3.amazon

https://www.hackerrank.com/challenges/picking-numbers/problem?isFullScreen=true

HackerRank Prepare Algorithms Implementation Picking Numbers

90%

Given an array of integers, find the longest subarray where the absolute difference between any two elements is less than or equal to 1.

Example

$a = [1, 1, 2, 2, 4, 4, 5, 5, 5]$

There are two subarrays meeting the criterion: $[1, 1, 2, 2]$ and $[4, 4, 5, 5, 5]$. The maximum length subarray has 5 elements.

Function Description

Complete the pickingNumbers function in the editor below.

pickingNumbers has the following parameter(s):

- int a[n]: an array of integers

Returns

- int: the length of the longest subarray that meets the criterion

Input Format

The first line contains a single integer n, the size of the array a.

The second line contains n space-separated integers, each an $a[i]$.

Constraints

- $2 \leq n \leq 100$
- $0 < a[i] < 100$
- The answer will be ≥ 2.

Sample Input 0

```
6
4 6 5 3 3 1
```

Change Theme Language C

```c
#include <stdio.h>

int pickingNumbers(int a[], int n) {
    int freq[101] = {0}; // Since numbers are in range 0..100

    for (int i = 0; i < n; i++)
        freq[a[i]]++;

    int maxLen = 0;
    for (int i = 0; i < 100; i++) {
        int len = freq[i] + freq[i + 1];
        if (len > maxLen)
            maxLen = len;
    }

    return maxLen;
}

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

    printf("%d\n", pickingNumbers(a, n));
    return 0;
}
```

Upload Code as File Test against custom input

Type here to search

23°C Mostly cloudy

pare Algorithms Implementation Climbing the Leaderboard

wants to climb to the top of the leaderboard and track their ranking. The game uses Dense

ard works like this:

ighest score is ranked number **1** on the leaderboard.

ual scores receive the same ranking number, and the next player(s) receive the immediately

mber.

80]

ave ranks **1**, **2**, **2**, and **3**, respectively. If the player's scores are **70**, **80** and **105**, their rankings

, 3rd and 1st. Return [4, 3, 1].

eaderboard function in the editor below.

s the following parameter(s):

derboard scores

yer's scores

nk after each new score

integer n, the number of players on the leaderboard.

space-separated integers $ranked[i]$, the leaderboard scores in decreasing order.

n integer, m, the number games the player plays.

space-separated integers $player[j]$, the game scores.

0^9 for $0 \leq i < n$

^{9} for $0 \leq j < m$

oard, $ranked$, is in descending order.

Change Theme Language C

```c
#include <stdio.h>

int main() {
    int n, m;
    scanf("%d", &n);
    int ranked[n];
    for (int i = 0; i < n; i++)
        scanf("%d", &ranked[i]);

    scanf("%d", &m);
    int player[m];
    for (int i = 0; i < m; i++)
        scanf("%d", &player[i]);


    int unique[n], uLen = 0;
    unique[uLen++] = ranked[0];
    for (int i = 1; i < n; i++) {
        if (ranked[i] != ranked[i - 1])
            unique[uLen++] = ranked[i];
    }


    int i = uLen - 1;
    for (int j = 0; j < m; j++) {
        while (i >= 0 && player[j] >= unique[i])
            i--;
        // Rank is i+2 (dense ranking)
        printf("%d\n", i + 2);
    }

    return 0;
}

```

Upload Code as File Test against custom input

search

23°C Mostly cloudy

A video player plays a game in which the character competes in a hurdle race. Hurdles are of varying heights, and the characters have a maximum height they can jump. There is a magic potion they can take that will increase their maximum jump height by 1 unit for each dose. How many doses of the potion must the character take to be able to jump all of the hurdles. If the character can already clear all of the hurdles, return 0.

Example

$height = [1, 2, 3, 3, 2]$

$k = 1$

The character can jump 1 unit high initially and must take $3 - 1 = 2$ doses of potion to be able to jump all of the hurdles.

Function Description

Complete the hurdleRace function in the editor below.

hurdleRace has the following parameter(s):

- int k: the height the character can jump naturally
- int height[n]: the heights of each hurdle

Returns

- int: the minimum number of doses required, always 0 or more

Input Format

The first line contains two space-separated integers n and k, the number of hurdles and the maximum height the character can jump naturally.

The second line contains n space-separated integers $height[i]$ where $0 \leq i < n$.

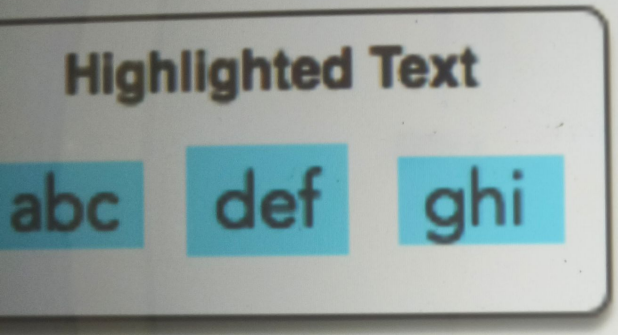
Constraints

Change Theme Language C

```c
#include <stdio.h>

int hurdleRace(int k, int height[], int n) {
    int maxHurdle = 0;
    for (int i = 0; i < n; i++)
        if (height[i] > maxHurdle)
            maxHurdle = height[i];

    int doses = maxHurdle - k;
    return doses > 0 ? doses : 0;
}

int main() {
    int n, k;
    scanf("%d %d", &n, &k);

    int height[n];
    for (int i = 0; i < n; i++)
        scanf("%d", &height[i]);

    printf("%d\n", hurdleRace(k, height, n));

    return 0;
}
```

Upload Code as File Test against custom input

ntiguous block of text is selected in a PDF viewer, the selection is highlighted
rectangle. In this PDF viewer, each word is highlighted independently. For



st of 26 character heights aligned by index to their letters. For example, 'a' is
nd 'z' is at index 25. There will also be a string. Using the letter heights given,
he area of the rectangle highlight in mm^2 assuming all letters are $1mm$ wide.

$1, 3, 1, 4, 1, 3, 2, 5, 5, 5, 5, 1, 1, 5, 5, 1, 5, 2, 5, 5, 5, 5, 5, 5]$ $word = 'torn'$

s are $t = 2, o = 1, r = 1$ and $n = 1$. The tallest letter is 2 high and there are 4
hightlighted area will be $2 * 4 = 8mm^2$ so the answer is 8.

Description

the designerPdfViewer function in the editor below.

Change Theme Language C

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

int designerPdfViewer(int h[26], char word[]) {
    int maxHeight = 0;
    int len = strlen(word);

    for (int i = 0; i < len; i++) {
        int index = word[i] - 'a';
        if (h[index] > maxHeight)
            maxHeight = h[index];
    }

    return maxHeight * len;
}

int main() {
    int h[26];
    for (int i = 0; i < 26; i++)
        scanf("%d", &h[i]);

    char word[101]; // word length <= 100
    scanf("%s", word);

    printf("%d\n", designerPdfViewer(h, word));
```

Upload Code as File Test against custom input

https://www.hackerrank.com/challenges/utopian-tree/problem?isFullScreen=true

nk Prepare Algorithms Implementation Utopian Tree

pian Tree goes through 2 cycles of growth every year. Each spring, it doubles in
Each summer, its height increases by 1 meter.

an Tree sapling with a height of 1 meter is planted at the onset of spring. How tall
ree be after n growth cycles?

mple, if the number of growth cycles is $n = 5$, the calculations are as follows:

od	Height
	1
	2
	3
	6
	7
	14

Description

e the utopianTree function in the editor below.

ee has the following parameter(s):

he number of growth cycles to simulate

e height of the tree after the given number of cycles

rmat

ine contains an integer, t, the number of test cases.

ent lines each contain an integer, n, the number of cycles for that test case.

nts

Change Theme Language C

```c
#include <stdio.h>

int utopianTree(int n) {
    int height = 1;
    for (int i = 1; i <= n; i++) {
        if (i % 2 == 1) // Spring
            height *= 2;
        else // Summer
            height += 1;
    }
    return height;
}

int main() {
    int t;
    scanf("%d", &t);

    for (int i = 0; i < t; i++) {
        int n;
        scanf("%d", &n);
        printf("%d\n", utopianTree(n));
    }

    return 0;
}
```

Upload Code as File Test against custom input

ere to search

23°C Mostly cl

ww.hackerrank.com/challenges/angry-professor/problem?isFullScreen=true

Prepare Algorithms Implementation Angry Professor

natics professor has a class of students. Frustrated with their lack of discipline, the
to cancel class if fewer than some number of students are present when class
s go from on time ($arrivalTime \leq 0$) to arrived late ($arrivalTime > 0$).
me of each student and a threshhold number of attendees, determine if the class is

, 2]
s arrived on. The last **2** were late. The threshold is **3** students, so class will go on.

e arrival times ($a[i] \leq 0$) indicate the student arrived early or on time; positive
> 0) indicate the student arrived $a[i]$ minutes late.

tion

gryProfessor function in the editor below. It must return YES if class is cancelled, or

as the following parameter(s):
shold number of students
rival times of the n students

YES or NO

nput contains t, the number of test cases.
consists of two lines.

Change Theme Language C

```c
#include <stdio.h>

const char* angryProfessor(int k, int a[], int n) {
    int onTime = 0;
    for (int i = 0; i < n; i++)
        if (a[i] <= 0)
            onTime++;

    return onTime >= k ? "NO" : "YES";
}

int main() {
    int t;
    scanf("%d", &t);

    for (int i = 0; i < t; i++) {
        int n, k;
        scanf("%d %d", &n, &k);

        int a[n];
        for (int j = 0; j < n; j++)
            scanf("%d", &a[j]);

        printf("%s\n", angryProfessor(k, a, n));
    }

    return 0;
}
```

Upload Code as File Test against custom input

to search

23°C Mostly cl

Lily likes to play games with integers. She has created a new game where she determines the difference between a number and its reverse. For instance, given the number 12, its reverse is 21. Their difference is 9. The number 120 reversed is 21, and their difference is 99.

She decides to apply her game to decision making. She will look at a numbered range of days and will only go to a movie on a beautiful day.

Given a range of numbered days, $[i \ldots j]$ and a number k, determine the number of days in the range that are beautiful. Beautiful numbers are defined as numbers where $|i - reverse(i)|$ is evenly divisible by k. If a day's value is a beautiful number, it is a beautiful day. Return the number of beautiful days in the range.

Function Description

Complete the beautifulDays function in the editor below.

beautifulDays has the following parameter(s):

- int i: the starting day number
- int j: the ending day number
- int k: the divisor

Returns

- int: the number of beautiful days in the range

Input Format

A single line of three space-separated integers describing the respective values of i, j, and k.

Constraints

- $1 \leq i \leq j \leq 2 \times 10^6$
- $1 \leq k \leq 2 \times 10^9$

Sample Input

```c
#include <stdio.h>
#include <stdlib.h>

int reverseNumber(int n) {
    int rev = 0;
    while (n > 0) {
        rev = rev * 10 + n % 10;
        n /= 10;
    }
    return rev;
}

int beautifulDays(int i, int j, int k) {
    int count = 0;
    for (int day = i; day <= j; day++) {
        int rev = reverseNumber(day);
        int diff = abs(day - rev);
        if (diff % k == 0)
            count++;
    }
    return count;
}

int main() {
    int i, j, k;
    scanf("%d %d %d", &i, &j, &k);
    printf("%d\n", beautifulDays(i, j, k));
    return 0;
}
```

Upload Code as File | Test against custom input

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

int viralAdvertising(int n) {
    int shared = 5;
    int cumulative = 0;

    for (int day = 1; day <= n; day++) {
        int liked = shared / 2; // floor division
        cumulative += liked;
        shared = liked * 3;
    }

    return cumulative;
}

int main() {
    int n;
    scanf("%d", &n);
    printf("%d\n", viralAdvertising(n));
    return 0;
}
```

nd a number of treats to pass out to them. Their jailer decides the
o seat the prisoners around a circular table in sequentially
will be drawn from a hat. Beginning with the prisoner in that chair,
prisoner sequentially around the table until all have been

ugh. The last piece of candy looks like all the others, but it tastes
occupied by the prisoner who will receive that candy.

ndy and distribution starts at chair 2. The prisoners arrange
. Prisoners receive candy at positions $2, 3, 4, 1, 2, 3$. The
mber 3.

on in the editor below. It should return an integer representing
arn.

ameter(s):

ssing out sweets from

r to warn

number of test cases.

Change Theme Language C

```c
#include <stdio.h>

int saveThePrisoner(int n, int m, int s) {
    return ((s - 1 + m - 1) % n) + 1;
}

int main() {
    int t;
    scanf("%d", &t);  // number of test cases

    while(t--) {
        int n, m, s;
        scanf("%d %d %d", &n, &m, &s);
        int result = saveThePrisoner(n, m, s);
        printf("%d\n", result);
    }

    return 0;
}
```

Upload Code as File Test against custom input

hr-testcases-us-east-1.s3.amazon

com/challenges/circular-array-rotation/problem?isFullScreen=true

orithms Implementation Circular Array Rotation

$[5, 3, 4] \rightarrow [4, 5, 3]$

based indices 1 and 2 as indicated in the *queries* array.

nction in the editor below.

parameter(s):

requested in m

ed integers, n, k, and q, the number of elements in the integer

er of queries.

arated integers, where each integer i describes array element

s a single integer, $queries[i]$, an index of an element in a to

Change Theme Language C

```c
#include <stdio.h>

void circularArrayRotation(int a[], int n, int k, int queries[], int q, int result
    k = k % n; // handle rotations greater than array size
    for(int i = 0; i < q; i++) {
        int index = (queries[i] - k + n) % n;  // find original index
        result[i] = a[index];
    }
}

int main() {
    int n, k, q;
    scanf("%d %d %d", &n, &k, &q);

    int a[n];
    for(int i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }

    int queries[q], result[q];
    for(int i = 0; i < q; i++) {
        scanf("%d", &queries[i]);
    }

    circularArrayRotation(a, n, k, queries, q, result);

    for(int i = 0; i < q; i++) {
        printf("%d\n", result[i]);
    }

}
```

Line:

Run Code

Change Theme Language C

```c
#include <stdio.h>

void circularArrayRotation(int a[], int n, int k, int queries[], int q, int result[]) {
    k = k % n; // handle rotations greater than array size
    for(int i = 0; i < q; i++) {
        int index = (queries[i] - k + n) % n;  // find original index
        result[i] = a[index];
    }
}

int main() {
    int n, k, q;
    scanf("%d %d %d", &n, &k, &q);

    int a[n];
    for(int i = 0; i < n; i++) {
        scanf("%d", &a[i]);
    }

    int queries[q], result[q];
    for(int i = 0; i < q; i++) {
        scanf("%d", &queries[i]);
    }

    circularArrayRotation(a, n, k, queries, q, result);

    for(int i = 0; i < q; i++) {
        printf("%d\n", result[i]);
    }

    return 0;
}
```

$(x) \leq n$. For

keep a

Change Theme Language

```c
#include <stdio.h>

void permutationEquation(int p[], int n, int result[]) {
    for(int x = 1; x <= n; x++) {
        int pos1, pos2;
        // Find position of x in p
        for(int i = 0; i < n; i++) {
            if(p[i] == x) {
                pos1 = i;
                break;
            }
        }
        // Find position of pos1+1 in p
        for(int i = 0; i < n; i++) {
            if(p[i] == pos1 + 1) {
                pos2 = i;
                break;
            }
        }
        result[x - 1] = pos2 + 1;
    }
}

int main() {
    int n;
    scanf("%d", &n);
    int p[n], result[n];

    for(int i = 0; i < n; i++) {
        scanf("%d", &p[i]);
    }

    permutationEquation(p, n, result);

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

Upload Code as File

Test against custom input

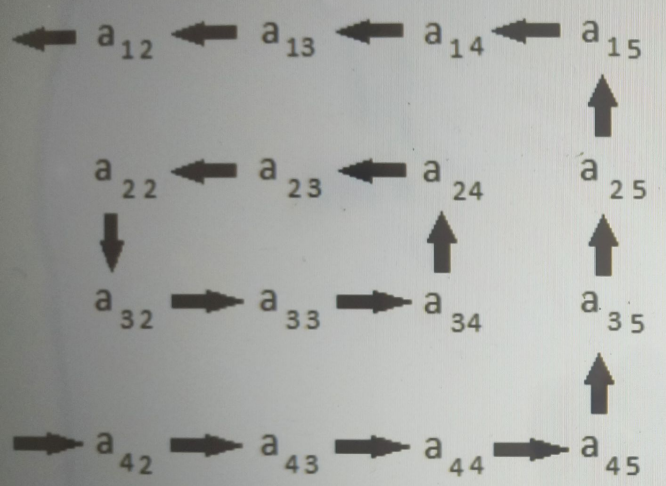
```c
#include <stdio.h>

int jumpingOnClouds(int c[], int n, int k) {
    int energy = 100;
    int i = 0;

    do {
        i = (i + k) % n;
        energy--;
        if(c[i] == 1) {
            energy -= 2;
        }
    } while(i != 0);

    return energy;
}

int main() {
    int n, k;
    scanf("%d %d", &n, &k);

    int c[n];
    for(int i = 0; i < n; i++) {
        scanf("%d", &c[i]);
    }

    int result = jumpingOnClouds(c, n, k);
    printf("%d\n", result);

    return 0;
}
```

Algorithms Implementation Matrix Layer Rotation

of dimension $m \times n$ and a positive integer r. You have to rotate the matrix r times and print the resultant matrix.

clockwise direction.

s represented by the following figure. Note that in one rotation, you have to shift elements by one step only.



Matrix Rotation

minimum of m and n will be even.

Start matrix by 2:

First				Second			
2	3	4	5	3	4	5	6
1	2	3	6	2	3	4	7
12	1	4	7	1	2	1	8
11	10	9	8	12	11	10	9

Change Theme Lan

```c
#include <stdio.h>

int main() {
    int m, n, r;
    scanf("%d %d %d", &m, &n, &r);

    int matrix[m][n];
    for (int i = 0; i < m; i++)
        for (int j = 0; j < n; j++)
            scanf("%d", &matrix[i][j]);

    int layers = (m < n ? m : n) / 2;

    for (int layer = 0; layer < layers; layer++) {
        int top = layer;
        int left = layer;
        int bottom = m - 1 - layer;
        int right = n - 1 - layer;

        int len = 2 * (bottom - top + right - left);
        int temp[len];
        int idx = 0;

        for (int j = left; j <= right; j++)
            temp[idx++] = matrix[top][j];
        for (int i = top + 1; i <= bottom - 1; i++)
            temp[idx++] = matrix[i][right];
        for (int j = right; j >= left; j--)
            temp[idx++] = matrix[bottom][j];
        for (int i = bottom - 1; i >= top + 1; i--)
            temp[idx++] = matrix[i][left];

        int rot = r % len;
        int rotated[len];
        for (int i = 0; i < len; i++)
            rotated[i] = temp[(i + rot) % len];

        idx = 0;
        for (int j = left; j <= right; j++)
            matrix[top][j] = rotated[idx++];
        for (int i = top + 1; i <= bottom - 1; i++)
            matrix[i][right] = rotated[idx++];
        for (int j = right; j >= left; j--)
            matrix[bottom][j] = rotated[idx++];
```

Upload Code as File Test against custom input

e to search

23°C Mostly cloud