A25 CS101 Lab Quiz 1 - Monday
8th September 2025 - 09:30 hrs to 11:00 hrs
3 Compulsory Questions - 15 Marks (5, 5, 5)

Instructions:

- Keep your ID card on the table for ready reference. If your ID card isn't with you, you won't be allowed to appear for the quiz/exam.
- Keep your phones, tablets, notes, bags, books, etc., near the instructor's platform.
- No clarifications will be provided for any question by anyone (TAs/Instructor).
- **Please note that your answers should NOT include any programming concept that hasn't been covered in the class. Such solutions if found will NOT be graded.**
- Marks will be given for each hidden test case that passes.
- TAs would be around to help you with respect to the logistics like compiling, submitting, etc. They will not help you debug the error or resolve your issues related to syntax/logical errors in your program.

- तैयार संदर्भ के लिए अपना ID card कार्ड table पर रखें। यदि आपका ID card आपके पास नहीं है, तो आपको quiz/परीक्षा में बैठने की अनुमति नहीं दी जाएगी।
- अपने phone, tablet, notes, bag, किताबें आदि प्रशिक्षक के मंच के पास रखें।
- किसी भी प्रश्न के लिए किसी भी (TA/Instructor) द्वारा कोई स्पष्टीकरण नहीं दिया जाएगा।
- **कृपया ध्यान दें कि आपके उत्तरों में कोई भी programming concept शामिल नहीं होनी चाहिए जिसे कक्षा में पहढ़या नहीं गया हो। यदि ऐसे समाधान पाए जाते हैं तो उन्हें grade नहीं किया जाएगा।**
- Marks केवल hidden test cases जो pass होते है, उन्हें दिए जाएंगे
- TAs program को compile, और submit सबमिट करने आदि जैसे logistics के संबंध में आपकी मदद करने के लिए मौजूद रहेंगे। वे आपको errors को डीबग करने या आपके program में syntax / logical errors से संबंधित कोई मदद नहीं करेंगे।

Q1. Counting Record-Breaking Runs

In the 100-meter Olympic finals, a runner's performance is measured by their finish time (in seconds). Over the years, multiple gold medalists have recorded their times. A performance is considered a **record-breaking run** if its finish time is **strictly less** than all previous times.

Given the times recorded by the gold medalists over the years, your task is to determine how many record-breaking runs occurred. The first recorded time is always considered a record-breaking run.

ओलंपिक की 100 मीटर फाइनल दौड़ में किसी धावक के प्रदर्शन को उसके समाप्ति समय (सेकंड में) से मापा जाता है। साल दर साल, कई स्वर्ण पदक विजेताओं ने अपने समय दर्ज किए हैं। किसी प्रदर्शन को रिकॉर्ड तोड़ दौड़ माना जाता है यदि उसका समाप्ति समय सभी पिछले समयों से कम हो।

आपको पिछले कुछ सालों के स्वर्ण पदक विजेताओं द्वारा दर्ज किए गए समय दिए गए हैं. आपका काम यह पता लगाना है कि कुल कितनी बार रिकॉर्ड तोड़ दौड़ें हुईं। पहला दर्ज किया गया समय हमेशा एक रिकॉर्ड तोड़ दौड़ माना जाता है।

Input Format

- The first line contains an integer **n** - the number of recorded times.
- The second line contains **n** real numbers (floating-point) separated by spaces - the times recorded by the gold medalists in chronological order.
- पहली पंक्ति में एक पूर्णांक **n** होगा - जो कि रिकॉर्ड किए गए समयों की संख्या है।
- दूसरी पंक्ति में **n** वास्तविक संख्याएँ (floating-point numbers) होंगी जो कि space से अलग-अलग की गई होंगी. ये स्वर्ण पदक विजेताओं द्वारा कालक्रम में रिकॉर्ड किए गए समय हैं।

Output Format

- Output a single integer - the total count of record-breaking runs.
- एक पूर्णांक Output करें - जो कि रिकॉर्ड तोड़ दौड़ों (record-breaking runs) की कुल संख्या हो।

Assumptions on Input

- 1 <= n <= 10^5
- Each recorded time is a positive real number between 9.0 and 15.0 (realistic 100m sprint times).
- Times are given in chronological order.
- The first time is always counted as a record-breaking run.
- 1 <= n <= 10^5
- प्रत्येक रिकॉर्ड किया गया समय 9.0 और 15.0 के बीच एक धनात्मक वास्तविक संख्या होगी (जो 100 मीटर स्प्रिंट के समय हैं)।
- समय कालक्रम (chronological order) में दिए गए हैं।
- पहला समय हमेशा रिकॉर्ड तोड़ दौड़ माना जाएगा।

Visible Testcases

Input	Output
5 9.95 9.93 9.98 9.90 9.89	4
4 10.0 10.1 10.2 9.9	2
3 9.85 9.85 9.80	2

Hidden Testcases

Input	Output
1 9.99	1
10 9.99 10.0 9.98 9.97 9.95 9.96 9.94 9.93 9.92 9.91	8
4 10.0 10.0 10.0 10.0	1
5 9.90 9.85 9.80 9.75 9.70	5
6 10.0 9.99 10.98 9.97 12.96 9.95	4

Solution :

C/C++

```
#include <simplecpp>

using namespace std;

int main() {
    int n;
    cin >> n;

    double time, bestTime = 1000.0;
    int recordCount = 0;

    for (int i = 0; i < n; i++) {
        cin >> time;
        if (time < bestTime) {
```

```
            recordCount++;
            bestTime = time;
        }
    }

    cout << recordCount << "\n";
    return 0;
}
```

Q2. Diagonal Triangle Fill

You've been tasked to write a program for creating a special numbering pattern for a triangular grid of side length **n**.

Imagine a triangle made up of rows, where the first row has **n** elements, the second row has **n-1**, the third row has **n-2**, and so on, until the last row has 1 element.

Your job is to fill this triangular grid with integers starting from 01, following a unique **diagonal numbering** pattern:

- Start from the **top-left element** of the triangle with 01.
- Fill the numbers **diagonally down, moving to the right**.
- Each diagonal starts at the top row and goes down to the bottom-left of the triangle, staying within the valid triangle bounds.

HINT - In each row, the differences of two consecutive entries are increasing by 1 with each step. The same is true for each column.

आपको एक प्रोग्राम लिखना है जिसका काम संख्याओं को एक त्रिभुजाकर ग्रिड में एक विशेष पैटर्न में प्रस्तुत करना है.
त्रिभुज में n पंक्तियाँ होंगी - पहली पंक्ति में n संख्याएँ, दूसरी पंक्ति में n-1, तीसरी पंक्ति में n-2, और इसी प्रकार, अंतिम पंक्ति में 1 संख्या होनी चाहिए।
त्रिभुजाकार ग्रिड की शुरुआत 01 से करते हुए, एक विशेष विकर्ण आधारित पैटर्न में आगे बढ़ना है.

- त्रिभुज की शुरुआत उसके ऊपरी-बाएँ कोने से होगी, जहाँ संख्या 01 होगी।
- हर विकर्ण में नीचे जाते हुए और एक विकर्ण से अगले दाईं ओर वाले विकर्ण में बढ़ते हुए संख्याएँ भरते चलें।
- हर विकर्ण ऊपरी पंक्ति से शुरू होकर त्रिभुज के निचले-बाएँ भाग तक जाता है, त्रिभुज की सीमाओं के भीतर रहते हुए।

सुझाव - प्रत्येक पंक्ति में, दो नज़दीकी संख्याओं का अंतर 1 से बढ़ता जा रहा है। यही बात प्रत्येक स्तंभ के लिए भी सत्य है।

Example

For n = 6, the output will be as follows (colors are added here just to show the diagonals, colors are not part of the actual output).

उदाहरण

n = 6 के लिए, आउटपुट नीचे दिया गया है (यहाँ रंग केवल विकर्ण दिखाने के लिए डाले गए हैं, रंग वास्तविक आउटपुट का हिस्सा नहीं हैं)।

01	02	04	07	11	16
03	05	08	12	17	
06	09	13	18		
10	14	19			
15	20				
21					

Input

- An integer n, the side length of the triangle.
- एक पूर्णांक n, जो कि त्रिभुज की भुजा की लंबाई होगी।

Output

- Print **n** lines.
- The i-th line contains **n - i + 1** integers corresponding to the i-th row of the triangle.
- n पंक्तियाँ प्रिंट करें।
- i-वीं पंक्ति में त्रिभुज की i-वीं पंक्ति के अनुरूप n - i + 1 संख्याएँ हैं।

Assumptions on Input

- 1 <= n <= 13

Note :

- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:
  - cout << "Enter a number:",
  - cout << "The computed answer is", etc.
- cout should be used to print only the computed final output. In addition, do not print unnecessary spaces unless specified in the program.
- **If any hard coding is found, or if any test case passes by merely writing a cout statement and without any logic, then the marks for that test case will NOT be awarded.**
- सामान्य संदेशों को प्रिंट करने के लिए कोई भी C++ स्टेटमेंट न लिखें। उदाहरण के लिए, आपके प्रोग्राम में निम्नलिखित मौजूद नहीं होने चाहिए:
  - cout << "Enter a number:",
  - cout << "The computed answer is", etc.
- cout का उपयोग केवल अंतिम आउटपुट को प्रिंट करने के लिए किया जाना चाहिए। प्रश्न में निर्दिष्ट स्पेस के अलावा, कोई भी अनावश्यक स्पेस न प्रिंट करें।
- **यदि कोई हार्ड कोडिंग पाई जाती है, या यदि कोई टेस्ट केस केवल एक cout स्टेटमेंट लिखकर और बिना किसी तर्क के पास हो जाता है, तो उस टेस्ट केस के लिए अंक नहीं दिए जाएँगे।**

Visible Testcases

Input	Output
9	01 02 04 07 11 16 22 29 37 03 05 08 12 17 23 30 38 06 09 13 18 24 31 39 10 14 19 25 32 40 15 20 26 33 41 21 27 34 42 28 35 43 36 44 45
7	01 02 04 07 11 16 22 03 05 08 12 17 23 06 09 13 18 24 10 14 19 25 15 20 26 21 27 28
6	01 02 04 07 11 16 03 05 08 12 17 06 09 13 18 10 14 19 15 20 21

Hidden Testcases

Input	Output
1	01
10	01 02 04 07 11 16 22 29 37 46 03 05 08 12 17 23 30 38 47 06 09 13 18 24 31 39 48 10 14 19 25 32 40 49 15 20 26 33 41 50 21 27 34 42 51 28 35 43 52 36 44 53 45 54 55
7	01 02 04 07 11 16 22 03 05 08 12 17 23 06 09 13 18 24 10 14 19 25 15 20 26 21 27 28
13	01 02 04 07 11 16 22 29 37 46 56 67 79 03 05 08 12 17 23 30 38 47 57 68 80 06 09 13 18 24 31 39 48 58 69 81 10 14 19 25 32 40 49 59 70 82

	15 20 26 33 41 50 60 71 83 21 27 34 42 51 61 72 84 28 35 43 52 62 73 85 36 44 53 63 74 86 45 54 64 75 87 55 65 76 88 66 77 89 78 90 91
3	01 02 04 03 05 06

Solution :

```
C/C++
#include<simplecpp>

main_program{
    int n;
    cin >> n;

    for (int row = 0; row < n; row++) {
        for (int col = 0; col < n - row; col++) {
            int diagonal_number = row + col;

            // Calculate the starting number for this diagonal
            int start = (diagonal_number * (diagonal_number + 1)) / 2 + 1;

            int num = start + row;
            if (num < 10) cout << '0' << num;
            else cout << num;

            // cout << setfill('0') << setw(2) << num;

            // To not add spaces at the end
            if (col < n - row - 1) cout << " ";
        }
        cout << endl;
    }
    return 0;
}
```

Q3. Calendar Day

In this question you need to write only two functions in the file **`implementation.cpp`** as described below. A main program (**`main.cpp`**) is already provided and is not editable.

इस प्रश्न में आपको केवल दो फ़ंक्शन एक फ़ाइल **implementation.cpp** में लिखने हैं, जैसा कि नीचे बताया गया है। एक main program (**main.cpp**) पहले से दिया गया है और उसमें कोई बदलाव नहीं किया जा सकता।

Function 1: `noOfDays: int noOfDays(int m)`

Write a function with name **`noOfDays`** that takes a single integer between 1 and 12 (inclusive of both) and returns the number of days in that month **for the year 2025.**

Feel free to use the calendar on your computer to verify the number of days in respective months if you do not remember them.

For example,

`noOfDays`(7) must return 31.

`noOfDays`(2) must return 28.

When this function is called:

- Argument 1 (m) is guaranteed to be an integer between 1 to 12.

एक फ़ंक्शन **noOfDays** नाम का लिखिए जो 1 से 12 के बीच (दोनों को शामिल करते हुए) एक पूर्णांक (integer) लेता है, और उस महीने के दिनों की संख्या (वर्ष **2025** में) संख्या return करता है।

यदि आपको याद नहीं कि महीनों में कितने दिन होते है, तो आप अपने कंप्यूटर के कैलेंडर में देख सकते हैं.

उदाहरण के लिए:

- **noOfDays(7)** को **31** return करना चाहिए।
- **noOfDays(2)** को **28** return करना चाहिए।

इसकी गारंटी है कि जब यह फ़ंक्शन कॉल किया जाएगा, तो उसका argument **m** हमेशा 1 से 12 के बीच का पूर्णांक ही होगा।

Function 2: `oneWeekLater: int oneWeekLater(int d, int m)`

Write another function **`oneWeekLater`** which **MUST USE** the function **`noOfDays.`**

This function takes 2 integers - the first integer represents the day and the second integer represents the month.

The function should return a single integer - the day (between 1 to 31) which occurs 7 days after the provided day and month (basically a week later).

For example,

`oneWeekLater`(20, 8) must return 27

since 27 August occurs 7 days after 20 August.

`oneWeekLater`(24, 2) must return 3.

Since 3 March occurs 7 days after 24 Feb.

When this function is called:

- Argument 1 (d) is guaranteed to be an integer between 1 to 31.
- Argument 2 (m) is guaranteed to be between 1 to 12.

एक और फ़ंक्शन **oneWeekLater** लिखिए, जिसे कि ऊपर बताए गए **noOfDays** फ़ंक्शन का उपयोग करना ही चाहिए।

यह फ़ंक्शन दो पूर्णांक (integer) लेता है – पहला पूर्णांक दिन (day) को दर्शाता है और दूसरा पूर्णांक महीने (month) को।

यह फ़ंक्शन एक पूर्णांक रिटर्न करेगा – वह तारीख (1 से 31 के बीच) जो दिए गए दिन और महीने से **7 दिन बाद** आती है (यानी एक सप्ताह बाद का दिन)।

उदाहरण के लिए:

- oneWeekLater(20, 8) को 27 रिटर्न करना चाहिए, क्योंकि 20 अगस्त के 7 दिन बाद 27 अगस्त आता है।
- oneWeekLater(24, 2) को 3 रिटर्न करना चाहिए, क्योंकि 24 फ़रवरी के 7 दिन बाद 3 मार्च आता है।

जब यह फ़ंक्शन कॉल किया जाएगा:

- Argument 1 (d) हमेशा 1 से 31 के बीच का पूर्णांक होगा।
- Argument 2 (m) हमेशा 1 से 12 के बीच का पूर्णांक होगा।

Main program: Provided

As mentioned above, the main program (`main.cpp`) is already written and provided to you. It accepts two integers (`day` and `month`) from the user. The functions, `noOfDays` and `oneWeekLater` are called from the main function with appropriate arguments.

जैसा कि ऊपर बताया गया है, main program (**main.cpp**) पहले से दिया हुआ है। यह उपयोगकर्ता से दो पूर्णांक (दिन और महीना) लेता है। फ़ंक्शन **noOfDays** और **oneWeekLater** को main function से उचित arguments के साथ कॉल किया जाता है।

Note:

- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:
  - cout << "Enter a number:",
  - cout << "The computed answer is", etc.
- cout should be used to print only the computed final output. In addition, do not print unnecessary spaces unless specified in the program.
- **If any hard coding is found, or if any test case passes by merely writing a cout statement and without any logic, then the marks for that test case will NOT be awarded.**
- सामान्य संदेशों को प्रिंट करने के लिए कोई भी C++ स्टेटमेंट न लिखें। उदाहरण के लिए, आपके प्रोग्राम में निम्नलिखित मौजूद नहीं होने चाहिए:
  - cout << "Enter a number:",
  - cout << "The computed answer is", etc.
- cout का उपयोग केवल अंतिम आउटपुट को प्रिंट करने के लिए किया जाना चाहिए। प्रश्न में बताई गई स्पेस के अलावा, कोई भी अनावश्यक स्पेस न प्रिंट करें।
- **यदि कोई हार्ड कोडिंग पाई जाती है, या यदि कोई टेस्ट केस केवल एक cout स्टेटमेंट लिखकर और बिना किसी तर्क के पास हो जाता है, तो उस टेस्ट केस के लिए अंक नहीं दिए जाएँगे।**

Visible Test Cases

Input	Output
20 8	31 27
27 8	31 3
22 2	28 1

Hidden Test Cases

Input	Output
16 11	30 23
24 7	31 31
24 9	30 1
25 12	31 1
21 2	28 28

Code:

implementation.cpp to be provided by student

```
C/C++
#include <simplecpp>
int noOfDays(int m)
{
    switch (m)
    {
    case 1:
        return 31;
    case 2:
        return 28;
    case 3:
        return 31;
    case 4:
        return 30;
    case 5:
        return 31;
    case 6:
        return 30;
    case 7:
        return 31;
    case 8:
        return 31;
    case 9:
        return 30;
    case 10:
        return 31;
    case 11:
        return 30;
    case 12:
        return 31;
    }
    return 0;
}
int oneWeekLater(int d, int m)
{
    int days = noOfDays(m);
    int next = d + 7;
    if (next <= days)
        return next;
    return next - days;
```

```
}
```

main.cpp to be provided by instructor

```cpp
#include <simplecpp>

int noOfDays(int m);
int oneWeekLater(int d, int m);

main_program
{
    int day, month;
    cin >> day >> month;
    cout << noOfDays(month) << " " << oneWeekLater(day, month) << endl;
}
```

A25 CS101 Lab Quiz 1 - Tuesday
9th September 2025 - 09:30 hrs to 11:00 hrs
3 Compulsory Questions - 15 Marks (5, 5, 5)

Instructions:

- Keep your ID card on the table for ready reference. If your ID card isn't with you, you won't be allowed to appear for the quiz/exam.
- Keep your phones, tablets, notes, bags, books, etc., near the instructor's platform.
- No clarifications will be provided for any question by anyone (TAs/Instructor).
- **Please note that your answers should NOT include any programming concept that hasn't been covered in the class. Such solutions if found will NOT be graded.**
- Marks will be given for each hidden test case that passes.
- TAs would be around to help you with respect to the logistics like compiling, submitting, etc. They will not help you debug the error or resolve your issues related to syntax/logical errors in your program.

- तैयार संदर्भ के लिए अपना ID card कार्ड table पर रखें। यदि आपका ID card आपके पास नहीं है, तो आपको quiz/परीक्षा में बैठने की अनुमति नहीं दी जाएगी।
- अपने phone, tablet, notes, bag, किताबें आदि प्रशिक्षक के मंच के पास रखें।
- किसी भी प्रश्न के लिए किसी भी (TA/Instructor) द्वारा कोई स्पष्टीकरण नहीं दिया जाएगा।
- **कृपया ध्यान दें कि आपके उत्तरों में कोई भी ऐसा programming concept शामिल नहीं होना चाहिए जिसे कक्षा में पढ़ाया नहीं गया हो। यदि ऐसे समाधान पाए जाते हैं तो उन्हें grade नहीं किया जाएगा।**
- Marks केवल hidden test cases जो pass होते है, उन्हें दिए जाएंगे
- TAs program को compile, और submit सबमिट करने आदि जैसे logistics के संबंध में आपकी मदद करने के लिए मौजूद रहेंगे। वे आपको errors को डीबग करने या आपके program में syntax / logical errors से संबंधित कोई मदद नहीं करेंगे।

Q1. Vowel Cluster Count

You're helping build a **text analyzer** tool that processes words and gives useful information about them. One feature you're working on is detecting **vowel clusters** - groups of vowels that appear one after another in a word.

Your task is to write a program that takes a word and counts how many **vowel clusters** it contains.

What is a vowel cluster?
A vowel cluster is a group of one or more vowels (a, e, i, o, u) that appear next to each other without any consonants in between.
For example:

- ae is one vowel cluster
- a is one vowel cluster
- abc has just one vowel cluster: a

For example, in the word **beautiful**, the vowels e, a, and u appear together, forming one vowel cluster. Later in the word, i and u appear alone, forming their own clusters. So in total, the word has **3 vowel clusters**.

आप एक ऐसा टेक्स्ट एनालाइज़र टूल बना रहे हैं जो शब्दों को प्रोसेस करके उनके बारे में उपयोगी जानकारी देता है। जिस फीचर पर आप काम कर रहे हैं, वह है **"vowel clusters"** (स्वर समूह) का पता लगाना – यानी एक शब्द में लगातार एक के बाद एक आने वाले स्वरों की पहचान करना।

आपको एक ऐसा प्रोग्राम लिखना है जो किसी शब्द को इनपुट के रूप में ले और उसमें **vowel clusters** (स्वर समूहों) की संख्या बताए।

vowel cluster (स्वर समूह) **क्या होता है?**
vowel cluster (स्वर समूह) एक या एक से अधिक स्वरों (a, e, i, o, u) का समूह होता है, जो बिना किसी व्यंजन (consonant) के बीच में आए, लगातार साथ-साथ आते हैं।

उदाहरण:

- **ae** एक vowel cluster है।
- **a** भी एक vowel cluster है।
- **abc** में केवल एक vowel cluster है: **a**

उदाहरण के लिए:

शब्द **beautiful** में, पहले **e, a, u** एक साथ आते हैं – यह एक vowel cluster है। बाद में शब्द में **i** और **u** अलग-अलग आते हैं, जो अपने-अपने vowel cluster बनाते हैं। तो कुल मिलाकर beautiful शब्द में **3 vowel cluster** हैं।

Input Format

- An integer n, the length of the word.
- A word of n lowercase English letters.
- एक पूर्णांक n, जो शब्द की लंबाई को दर्शाता है।
- n अंग्रेज़ी अक्षरों (small) से बना एक शब्द।

Output Format

- An integer: the number of vowel clusters in the word.
- एक पूर्णांक: शब्द में मौजूद vowel clusters (स्वर समूहों) की संख्या।

Assumptions on Input

- 1 <= n <= 10^5

Note

- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:
  - cout << "Enter a number:",
  - cout << "The computed answer is", etc.
- cout should be used to print only the computed final output. In addition, do not print unnecessary spaces unless specified in the program.
- **If any hard coding is found, or if any test case passes by merely writing a cout statement and without any logic, then the marks for that test case will NOT be awarded.**
- सामान्य संदेशों को प्रिंट करने के लिए कोई भी C++ स्टेटमेंट न लिखें। उदाहरण के लिए, आपके प्रोग्राम में निम्नलिखित मौजूद नहीं होने चाहिए:
  - cout << "Enter a number:",
  - cout << "The computed answer is", etc.
- cout का उपयोग केवल अंतिम आउटपुट को प्रिंट करने के लिए किया जाना चाहिए। प्रश्न में निर्दिष्ट स्पेस के अलावा, कोई भी अनावश्यक स्पेस न प्रिंट करें।
- **यदि कोई हार्ड कोडिंग पाई जाती है, या यदि कोई टेस्ट केस केवल एक cout स्टेटमेंट लिखकर और बिना किसी तर्क के पास हो जाता है, तो उस टेस्ट केस के लिए अंक नहीं दिए जाएँगे।**

Visible Testcases

Input	Output
9 beautiful	3
7 greater	2
3 bcd	0

Hidden Testcases

Input	Output
1 b	0
10 aaeeiioouu	1
7 abecidu	4
15 abcdeiouxyzuaei	3
8 abcdeifg	2

Solution :

```cpp
C/C++
#include<simplecpp>
main_program{
    int n;
    cin >> n;

    char c;

    int clusterCount = 0;
    bool inCluster = false;

    for (int i = 0; i < n; i++) {
        cin >> c;
        if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {
            if (inCluster == false) {
                clusterCount++;
                inCluster = true;
            }
        } else {
            inCluster = false;
        }
    }
    cout << clusterCount << endl;
    return 0;
}
```

Q2. Concentric L’s

Write a program to accept an integer n and print an n x n square grid filled with alternating concentric L-shaped patterns using the characters 'o' and 'x'.

The pattern always starts with 'o'.
Next row and column are filled with 'x' but that forms an L shape flipped on the vertical axis
Then again with 'o', alternating until the grid is completely filled.

एक प्रोग्राम लिखिए जो एक पूर्णांक n इनपुट के रूप में ले और n x n का एक वर्गाकार ग्रिड (square grid) प्रिंट करे, जो कि संकेन्द्रित (concentric) L-आकार के पैटर्न से भरा हो।ये L-आकार के पैटर्न अदल बदल कर 'o' और 'x' अक्षरों से बने होना चाहिए.

पैटर्न की शुरुआत हमेशा **'o'** से होती है। अगली पंक्ति और स्तंभ 'x' से भरे जाते हैं, जो एक उल्टा L आकार बनाते हैं (ऊर्ध्वाधर अक्ष (vertical axis) पर पलटा हुआ)। इसके बाद फिर से 'o' का उपयोग होता है, और यह क्रम तब तक चलता है जब तक पूरा ग्रिड भर नहीं जाता।

Example

Input:
4

Output:
```
oxox
xxox
ooox
xxxx
```

```
Input
7

Output
oxoxoxo
xxoxoxo
oooxoxo
xxxxoxo
oooooxo
xxxxxxo
ooooooo
```

Input format:

- Integer n
- एक पूर्णांक **n**

Output format:

- Print the n x n pattern of concentric Ls using 'o' and 'x'.
- 'o' और 'x' का से बना n x n आकार का संकेन्द्रित L-आकार वाला पैटर्न प्रिंट करें।

Note

- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:
    - cout << "Enter a number:",
    - cout << "The computed answer is", etc.

- cout should be used to print only the computed final output. In addition, do not print unnecessary spaces unless specified in the program.
- **If any hard coding is found, or if any test case passes by merely writing a cout statement and without any logic, then the marks for that test case will NOT be awarded.**
- सामान्य संदेशों को प्रिंट करने के लिए कोई भी C++ स्टेटमेंट न लिखें। उदाहरण के लिए, आपके प्रोग्राम में निम्नलिखित मौजूद नहीं होने चाहिए:
    - cout << "Enter a number:",
    - cout << "The computed answer is", etc.
- cout का उपयोग केवल अंतिम आउटपुट को प्रिंट करने के लिए किया जाना चाहिए। प्रश्न में निर्दिष्ट स्पेस के अलावा, कोई भी अनावश्यक स्पेस न प्रिंट करें।
- **यदि कोई हार्ड कोडिंग पाई जाती है, या यदि कोई टेस्ट केस केवल एक cout स्टेटमेंट लिखकर और बिना किसी तर्क के पास हो जाता है, तो उस टेस्ट केस के लिए अंक नहीं दिए जाएँगे।**

Visible Test Cases

2	ox xx
5	oxoxo xxoxo oooxo xxxxo ooooo
7	oxoxoxo xxoxoxo oooxoxo xxxxoxo oooooxo xxxxxxo ooooooo

Hidden Test Cases

1	o
4	oxox xxox ooox xxxx
3	oxo xxo ooo
6	oxoxox xxoxox oooxox xxxxox ooooox xxxxxx

10	oxoxoxoxox xxoxoxoxox oooxoxoxox xxxxoxoxox oooooxoxox xxxxxxoxox oooooooxox xxxxxxxxox ooooooooox xxxxxxxxxx

Solution :

C/C++

```
#include <simplecpp>
main_program {
    int n;
    cin >> n;

    // Loop through each cell of the grid
    for (int r = 0; r < n; r++) {
        for (int c = 0; c < n; c++) {
            // Distance from bottom boundary
            int distBottom = n - 1 - r;
            // Distance from right boundary
            int distRight  = n - 1 - c;

            // Layer number = smaller of the two distances
            int layer;
            if (distBottom < distRight)
                layer = distBottom;
            else
                layer = distRight;

            // For even n, outer layer parity flips so adjust by +1
            int effectiveLayer = (n % 2 == 0) ? (layer + 1) : layer;

            // Even layer = 'o', Odd layer = 'x'
            if (effectiveLayer % 2 == 0)
                cout << 'o';
            else
                cout << 'x';
        }
```

```
        cout << "\n";
    }

    return 0;
}
```

Q3. Playing Cards

In this question you need to write only two functions in the file **`implementation.cpp`** as described below. A main program (**`main.cpp`**) is already provided and is not editable.

इस प्रश्न में आपको केवल दो फ़ंक्शन एक फ़ाइल **implementation.cpp** में लिखने हैं, जैसा कि नीचे बताया गया है। एक main program (**main.cpp**) पहले से दिया गया है और उसमें कोई बदलाव नहीं किया जा सकता।

Function 1 `intValue: int intValue(char v)`

Write a function with the name **`intValue`** that takes a single character as an argument from the given set

1, 2, 3, 4, 5, 6, 7, 8, 9, J, Q, K, A

And returns an integer mapping them to these numbers, respectively

10, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14.

Essentially, it creates the following mapping from characters to integers as shown in the table.

Characters	Integer
1	10
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
J	11
Q	12
K	13
A	14

For example,

`intValue`(J) must return 11.

`intValue`(5) must return 5.

When this function is called, argument values are guaranteed to be from the “characters” column of the table

एक फ़ंक्शन लिखिए जिसका नाम **`intValue`** हो, जो एक english अक्षर (character) को argument के रूप में लेता है। यह अक्षर निम्नलिखित सेट में से कोई एक होगा:

1, 2, 3, 4, 5, 6, 7, 8, 9, J, Q, K, A

यह फ़ंक्शन इन अक्षरों को नीचे दिए गए पूर्णांकों (integer) से मैप करेगा और उस पूर्णांक को returns करेगा। मैपिंग के लिए ऊपर दी गई सारणी देखें।

10, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14.

उदाहरण**:**

- `intValue('J')` को 11 return करना चाहिए।
- `intValue('5')` को 5 return करना चाहिए।

जब यह फ़ंक्शन कॉल किया जाएगा, तो इनपुट हमेशा उपरोक्त दिए गए "characters" कॉलम में से ही होगा।

Function 2 `winner: int winner(char s1, char v1, char s2, char v2, char s3, char v3)`

There are 3 players playing a card game. Each player plays one card that belongs to one of the four suits mentioned below. The player who has the highest value card of the same suit as player 1 will win.

Write a function **`winner`** that **MUST USE** the function **`intValue`**.

- 6 arguments: 3 cards (suit + value) for the 3 players
- **Suits**: S (Spades), H (Hearts), D (Diamonds), C (Clubs). Arguments s1, s2, and s3 are guaranteed to be a valid suit i.e. S, H, D, or C
- **Values**: Arguments v1, v2, v3 are guaranteed to have valid inputs for the intValue function (i.e. characters from the table - 1, 2, J, K).
- **The cards are guaranteed to be distinct (you will not encounter a set of 2 cards with the same suit and value).**
- This function must return a **single integer (between 1 to 3) representing the winning player, that is, the player who has the highest valued card of the same suit as player 1.**

Examples

Function Call	Should Return	Explanation
winner(S, 5, H, 2, S, 3)	1	The first and third cards have the same suit i.e. ‘Spades’. intValue(5) > intValue(3), hence, player 1 wins
winner(D, 9, C, K, D, Q)	3	Since D is the suit of the first and third cards and intValue(Q) > intValue(9), player 3 wins
winner(H, 2, H, 9, H, 1)	3	since intValue(1) > intValue(9) > intValue(2) and all the cards are from suit H, player 3 wins.
winner(S, 2, D, A, C, 1)	1	since first card is the only card from the first suit (which is S), player 1 wins.

तीन खिलाड़ी एक ताश का खेल खेल रहे हैं। प्रत्येक खिलाड़ी एक ताश का पत्ता (card) खेलता है, जो नीचे दिए गए चार सूट्स (suits) में से किसी एक का होता है। विजेता वह खिलाड़ी होगा जिसके पास वही सूट हो जो खिलाड़ी 1 के पास है, और उस सूट में सबसे अधिक मान (value) वाला कार्ड हो।

एक फ़ंक्शन लिखिए जिसका नाम winner हो और जो intValue फ़ंक्शन का उपयोग करे।

- इनपुट (कुल 6 arguments): तीन खिलाड़ियों के तीन cards. हर कार्ड का एक सूट और एक वैल्यू.
- Suits (सूट्स): 'S' (Spades हुकूम), 'H' (Hearts पान), 'D' (Diamonds ईंट), 'C' (Clubs चिड़ी) Arguments s1, s2, और s3 हमेशा इन चार में से कोई एक सूट होंगे.
- Values (मान): v1, v2, और v3 ऐसे अक्षर होंगे जो intValue फ़ंक्शन के लिए वैध argument हों — जैसे '1', '2', 'J', 'K' आदि।
- सभी कार्ड आपस में भिन्न (distinct) होंगे — यानी कोई दो कार्ड एक ही सूट और वैल्यू नहीं रखेंगे।
- यह फ़ंक्शन एक पूर्णांक (1 से 3 के बीच) लौटाएगा, जो विजेता खिलाड़ी का नंबर होगा —अर्थात वह खिलाड़ी जिसके पास वही सूट है जो खिलाड़ी 1 के पास है और उस सूट का सबसे अधिक वैल्यू वाला कार्ड है।

उदाहरण

Function Call	Should Return	Explanation
winner(S, 5, H, 2, S, 3)	1	पहला और तीसरा कार्ड एक ही सूट (spades) के हैं. और intValue(5) > intValue(3), इसलिए खिलाड़ी 1 विजेता होगा.
winner(D, 9, C, K, D, Q)	3	चूँकि पहले और तीसरे कार्ड का सूट D है और intValue(Q) > intValue(9), इसलिए खिलाड़ी 3 विजेता होगा.
winner(H, 2, H, 9, H, 1)	3	चूँकि सारे कार्ड एक ही सूट D के हैं और intValue(1) > intValue(9) > intValue(2), इसलिए खिलाड़ी 3 विजेता होगा.
winner(S, 2, D, A, C, 1)	1	चूँकि पहले कार्ड के अलावा और कोई कार्ड उस सूट S का नहीं है, इसलिए खिलाड़ी 1 विजेता होगा.

Main program: Provided

The main program accepts characters as input (as a stream of characters). For example S9D7SQ. The functions, `intValue` and `winner`, are called with appropriate arguments and the results are printed.

main program कैरेक्टर्स की एक स्ट्रीम को इनपुट के रूप में लेता है — जैसे कि: S9D7SQ. इसके बाद, प्रोग्राम में intValue और winner फ़ंक्शनों को उपयुक्त arguments के साथ कॉल किया जाता है और उनका परिणाम (Result) प्रिंट किया जाता है।

Note:

- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:
  - cout << "Enter a number:",
  - cout << "The computed answer is", etc.
- cout should be used to print only the computed final output. In addition, do not print unnecessary spaces unless specified in the program.
- **If any hard coding is found, or if any test case passes by merely writing a cout statement and without any logic, then the marks for that test case will NOT be awarded.**

- सामान्य संदेशों को प्रिंट करने के लिए कोई भी C++ स्टेटमेंट न लिखें। उदाहरण के लिए, आपके प्रोग्राम में निम्नलिखित मौजूद नहीं होने चाहिए:
  - cout << "Enter a number:",
  - cout << "The computed answer is", etc.
- cout का उपयोग केवल अंतिम आउटपुट को प्रिंट करने के लिए किया जाना चाहिए। प्रश्न में बताई गई स्पेस के अलावा, कोई भी अनावश्यक स्पेस न प्रिंट करें।
- **यदि कोई हार्ड कोडिंग पाई जाती है, या यदि कोई टेस्ट केस केवल एक cout स्टेटमेंट लिखकर और बिना किसी तर्क के पास हो जाता है, तो उस टेस्ट केस के लिए अंक नहीं दिए जाएँगे।**

Visible Test Cases

Input	Output
S9H7SQ	9 7 12 3
H1H9H7	10 9 7 1
DQDKSA	12 13 14 2

Hidden Test Cases

Input	Output
C2SAH1	2 14 10 1
C9C1D1	9 10 10 2
DAS2D9	14 2 9 1
H3H8H1	3 8 10 3
SAHKDQ	14 13 12 1

Solution:

implementation.cpp (to be written by the student)

```
C/C++
#include <simplecpp>
int intValue(char c)
{
    if (c == '1')
        return 10;
    if (c >= '2' && c <= '9')
        return c - '0';
    if (c == 'J')
        return 11;
    if (c == 'Q')
        return 12;
    if (c == 'K')
        return 13;
    return 14; // A
}

int winner(char s1, char v1, char s2, char v2, char s3, char v3)
{
    int a = intValue(v1), b = intValue(v2), c = intValue(v3);

    if (s2 != s1 && s3 != s1)
        return 1;
    else if (s2 == s1 && s3 != s1)
        return (a > b ? 1 : 2);
```

```
    else if (s2 != s1 && s3 == s1)
        return (a > c ? 1 : 3);
    else
    { // all same suit
        if (a >= b && a >= c)
            return 1;
        else if (b >= a && b >= c)
            return 2;
        else
            return 3;
    }
}
```

main.cpp (to be provided by the instructor)

```
C/C++
#include <simplecpp>

int intValue(char v);
int winner(char s1, char v1, char s2, char v2, char s3, char v3);

main_program
{
    char v1, s1, v2, s2, v3, s3;
    cin >> s1 >> v1 >> s2 >> v2 >> s3 >> v3;
    cout << intValue(v1) << " " << intValue(v2) << " " << intValue(v3) << " ";
    cout << winner(s1, v1, s2, v2, s3, v3) << endl;
    return 0;
}
```

A25 CS101 Lab Quiz 1 - Thursday
11th September 2025 - 09:30 hrs to 11:00 hrs
3 Compulsory Questions - 15 Marks (5, 5, 5)

Instructions:

- Keep your ID card on the table for ready reference. If your ID card isn't with you, you won't be allowed to appear for the quiz/exam.
- Keep your phones, tablets, notes, bags, books, etc., near the instructor's platform.
- No clarifications will be provided for any question by anyone (TAs/Instructor).
- **Please note that your answers should NOT include any programming concept that hasn't been covered in the class. Such solutions if found will NOT be graded.**
- Marks will be given for each hidden test case that passes.
- TAs would be around to help you with respect to the logistics like compiling, submitting, etc. They will not help you debug the error or resolve your issues related to syntax/logical errors in your program.

- TAs द्वारा जाँच के लिए अपना ID card कार्ड table पर रखें। यदि आपका ID card आपके पास नहीं है, तो आपको quiz/परीक्षा में बैठने की अनुमति नहीं दी जाएगी।
- अपने phone, tablet, notes, bag, किताबें आदि प्रशिक्षक के मंच के पास रखें।
- किसी भी प्रश्न के लिए किसी भी (TA/Instructor) द्वारा कोई स्पष्टीकरण नहीं दिया जाएगा।
- **कृपया ध्यान दें कि आपके उत्तरों में कोई भी ऐसा programming concept शामिल नहीं होना चाहिए जिसे कक्षा में पढ़ाया नहीं गया हो। यदि ऐसे समाधान पाए जाते हैं तो उन्हें grade नहीं किया जाएगा।**
- Marks केवल जो hidden test cases pass होते है, उनके लिए दिए जाएंगे
- TAs program को compile, और submit सबमिट करने आदि जैसे logistics के संबंध में आपकी मदद करने के लिए मौजूद रहेंगे। वे आपको errors को डीबग करने या आपके program में syntax / logical errors से संबंधित कोई मदद नहीं करेंगे।

Q1. Turtle Directions

This question is about doing some computation for a given sequence of turtle commands. You do not need to actually draw anything using turtleSim(). Just need to output a sequence of 0/1 as described below.
इस सवाल में आपको कुछ turtle commands दी जाएंगी. और उसके आधार पर कुछ computation करने को कहा जाएगा. आपको वास्तव में turtleSim() का उपयोग करके कुछ भी ड्रॉ (draw) नहीं करना है। आपको केवल नीचे दिए गए अनुसार 0/1 की एक शृंखला आउटपुट करनी है।

Write a program for the following :
You are given an integer **n** and a sequence of **n** turtle commands from {F, L}.

1. F denotes move forward by 100 units.
2. L denotes turn left by 120 degrees.

After executing each command print if turtle is in the starting position/coordinates or not. Print 1 if yes, print 0 if not the starting coordinates (**without any space**).
Note : The angle/direction does not matter, only the coordinates do.

Hint: For any command F, Turtle will be moving in one of three directions (0 degrees, 120 degrees and 240 degrees). Turtle will reach the original position if and only if the total number of forward steps in each of three directions are all equal.

आपको एक पूर्णांक n और n turtle commands की एक शृंखला दी जाएगी, जहाँ प्रत्येक कमांड सेट {F, L} में से होगी.

1. F का अर्थ है — turtle को 100 यूनिट आगे बढ़ाना।
2. L का अर्थ है — turtle को 120 डिग्री बाईं ओर मोड़ना।

हर कमांड को एक-एक करके निष्पादित (execute) करने के बाद आपको यह देखना है कि क्या turtle फिर से अपनी शुरुआती स्थिति (starting position / original coordinates) पर लौट आया है या नहीं? अगर हाँ, तो 1 प्रिंट करें, अगर नहीं, तो 0 प्रिंट करें (**कोई space बिना**).
ध्यान दें: केवल स्थिति (coordinates) मायने रखती है, दिशा (angle/direction) नहीं।
सुझाव (Hint): टर्टल केवल तीन दिशाओं में आगे बढ़ सकता है: 0 डिग्री, 120 डिग्री, और 240 डिग्री। turtle शुरुआती स्थिति पर तभी लौटेगा, जब इन तीनों दिशाओं में कुल कदम संख्या बराबर हो।

Example:
Input:
4
L F L L
Output:
1000

Explanation :
The first L command changes the direction but the turtle is still at the start → so we print 1.
The next commands move it away from the start -> print 0.
पहला L कमांड केवल दिशा बदलता है, लेकिन turtle अब भी प्रारंभिक स्थान (starting position) पर ही है → इसलिए हम 1 प्रिंट करते हैं।
इसके बाद आने वाले कमांड्स turtle को प्रारंभिक स्थान से दूर ले जाते हैं → इसलिए हम 0 प्रिंट करते हैं।

Example 2:
Input:
6
F L F L F L

Output:
000011

Explanation :
In the first 4 commands, the turtle is not at the start → so we print 0.
After the 5th command, the turtle comes back to the start → we print 1.
The last move is just a turn - changes only the direction the turtle is facing, position doesn't change → still at the start, so we print 1.

पहले चार कमांड्स के बाद turtle प्रारंभिक स्थान (starting position) पर नहीं होता → इसलिए हम 0 प्रिंट करते हैं।
पाँचवें कमांड के बाद turtle फिर से प्रारंभिक स्थान पर लौट आता है → इसलिए हम 1 प्रिंट करते हैं।
आखिरी कमांड सिर्फ एक मोड़ (turn) है — यह केवल turtle की दिशा बदलता है, उसकी स्थिति नहीं बदलती → turtle अब भी प्रारंभिक स्थान पर है, इसलिए हम 1 प्रिंट करते हैं।

Input format:

- Integer n : size of command sequence
- n characters separated by space, denoting the commands (F or L)
- पूर्णांक n : commands की संख्या
- n अक्षर, जो स्पेस द्वारा अलग किए गए हैं, और प्रत्येक अक्षर एक कमांड को दर्शाता है — F (आगे बढ़ो) या L (बाएँ मुड़ो)।

Output format:

- After each command : print 1 for yes - the turtle is in the original position, or print 0 for no - the turtle is NOT in the original position, **without any space**.
- हर कमांड के निष्पादन (execution) के बाद: यदि turtle प्रारंभिक स्थान (original position) पर है, तो 1 प्रिंट करें, और यदि turtle प्रारंभिक स्थान पर नहीं है, तो 0 प्रिंट करें, **कोई space बिना.**

Note

- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:
  - cout << "Enter a number:",
  - cout << "The computed answer is", etc.
- cout should be used to print only the computed final output. In addition, do not print unnecessary spaces unless specified in the program.

- **If any hard coding is found, or if any test case passes by merely writing a cout statement and without any logic, then the marks for that test case will NOT be awarded.**
- सामान्य संदेशों को प्रिंट करने के लिए कोई भी C++ स्टेटमेंट न लिखें। उदाहरण के लिए, आपके प्रोग्राम में निम्नलिखित मौजूद नहीं होने चाहिए:
  - cout << "Enter a number:",
  - cout << "The computed answer is", etc.
- cout का उपयोग केवल अंतिम आउटपुट को प्रिंट करने के लिए किया जाना चाहिए। प्रश्न में बताई गई स्पेस के अलावा, कोई भी अनावश्यक स्पेस न प्रिंट करें।
- **यदि कोई हार्ड कोडिंग पाई जाती है, या यदि कोई टेस्ट केस केवल एक cout स्टेटमेंट लिखकर और बिना किसी तर्क के पास हो जाता है, तो उस टेस्ट केस के लिए अंक नहीं दिए जाएँगे।**

Visible Test Cases

1 F	0
4 L F L L	1000
7 F L F L F L F	0000110

Hidden Test Cases

3 L L L	111
12 F L F L F L F L F L F L	000011000011
9 F L L F L L F L L	000000111
3 F L F	000

5 L L F L L	11000

Solution:

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;   // number of commands

    int zero = 0, onetwenty = 0, twoforty = 0;
    int angle = 0;

    for (int i = 0; i < n; i++) {
        char dir;
        cin >> dir;

        if (dir == 'L') {
            angle += 120;
            angle %= 360;
        }
        else if (dir == 'F') {
            if (angle == 0) zero += 100;
            else if (angle == 120) onetwenty += 100;
            else if (angle == 240) twoforty += 100;
        }

        // check after each command
        if (zero == onetwenty && onetwenty == twoforty)
            cout << 1;
        else
            cout << 0;
    }

    return 0;
}
```

[redacted]

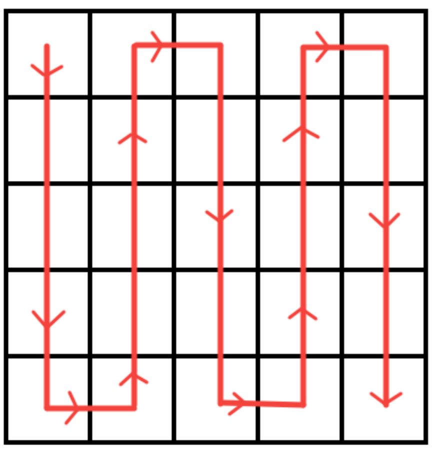
Q2. Snake Patterns

Write a program to accept an integer **'n'** representing the size of an **n×n** grid. Your task is to print the pattern as shown below, such that the grid is filled with integers from **1 to n^2** in a snake-like pattern **column-wise** as follows:

- For **first column**, fill numbers from **top to bottom**.
- For **second column**, fill numbers **bottom to top**.
- Repeat the above rules the for rest of the columns alternatively.

एक प्रोग्राम लिखिए जिसका इनपुट एक पूर्णांक n हो और जो एक n × n ग्रिड में संख्याएँ प्रिंट करे। आपको ग्रिड में 1 से लेकर n^2 तक की संख्याएँ भरना है कि जो कि नीचे दर्शाए गए साँप जैसे पैटर्न में बढ़ रही हों।

- पहले कॉलम में संख्याएँ ऊपर से नीचे भरें।
- दूसरे कॉलम में संख्याएँ नीचे से ऊपर भरें।
- इसी क्रम को बाकी के कॉलम्स के लिए बारी-बारी से दोहराएँ।



Input Format

- A single integer n (1 <= n <= 9).
- एक पूर्णांक संख्या n (1 <= n <= 9).

Output Format

- Print the n×n grid.
- Each number must be printed as a two-digit number (leading zeros if needed).
- Numbers in the same row are separated by a space.
- Print each row in a new line.
- n × n का ग्रिड प्रिंट करें।
- हर संख्या दो अंकों में प्रिंट होनी चाहिए (यदि आवश्यक हो तो शून्य जोड़ें, जैसे 01, 02, ..., 10 आदि)।
- एक पंक्ति (row) में स्थित अगल बगल की संख्याओं के बीच में स्पेस दें।
- प्रत्येक पंक्ति को नई लाइन में प्रिंट करें।

Assumptions on Input

- 1 <= n <= 9
- n is an integer.

Visible Testcases

Input	Output
4	01 08 09 16 02 07 10 15 03 06 11 14 04 05 12 13
2	01 04 02 03
3	01 06 07 02 05 08 03 04 09

Hidden Testcases

Input	Output
6	01 12 13 24 25 36 02 11 14 23 26 35 03 10 15 22 27 34 04 09 16 21 28 33 05 08 17 20 29 32 06 07 18 19 30 31
8	01 16 17 32 33 48 49 64 02 15 18 31 34 47 50 63 03 14 19 30 35 46 51 62 04 13 20 29 36 45 52 61 05 12 21 28 37 44 53 60 06 11 22 27 38 43 54 59 07 10 23 26 39 42 55 58 08 09 24 25 40 41 56 57
9	01 18 19 36 37 54 55 72 73 02 17 20 35 38 53 56 71 74 03 16 21 34 39 52 57 70 75 04 15 22 33 40 51 58 69 76 05 14 23 32 41 50 59 68 77 06 13 24 31 42 49 60 67 78

	07 12 25 30 43 48 61 66 79 08 11 26 29 44 47 62 65 80 09 10 27 28 45 46 63 64 81
1	01
7	01 14 15 28 29 42 43 02 13 16 27 30 41 44 03 12 17 26 31 40 45 04 11 18 25 32 39 46 05 10 19 24 33 38 47 06 09 20 23 34 37 48 07 08 21 22 35 36 49

Solution :

C/C++

```
#include <simplecpp>

main_program {
    int n;
    cin >> n;

    for (int row = 0; row < n; ++row) {
        for (int col = 0; col < n; ++col) {
            int num;

            if (col % 2 == 0) {
                num = col * n + row + 1;
            } else {
                num = (col + 1) * n - row;
            }

            if (num < 10)
                cout << '0';
            cout << num;

            if (col != n - 1)
                cout << " ";
        }
        cout << endl;
    }
```

```
}
```

Q3. Acute Triangle Check

In this question you need to write only two functions in the file **`implementation.cpp`** as described below. A main program (**`main.cpp`**) is already provided and is not editable.

इस प्रश्न में आपको केवल दो फ़ंक्शन एक फ़ाइल **implementation.cpp** में लिखने हैं, जैसा कि नीचे बताया गया है। एक main program (**main.cpp**) पहले से दिया गया है और उसमें कोई बदलाव नहीं किया जा सकता।

In geometry, a triangle is called **acute-angled** if **all three of its internal angles are less than 90 degrees**. Given three 2D points representing the vertices of a triangle, your task is to determine whether the triangle is acute-angled.

Your task is to implement the following two functions:

1. bool isAcute(int x1, int y1, int x2, int y2) - Given two 2D vectors (x1, y1) and (x2, y2), return true if the angle between them is acute, else return false. i.e., whether the angle between them is less than 90 degrees. This can be done using the dot product. Given two 2D vectors A and B, the angle between them is acute if and only if their dot product is positive.
2. bool isAcuteTriangle(int x1, int y1, int x2, int y2, int x3, int y3) - this function must use the **isAcute function.** Given three 2D points representing the vertices of a triangle:
    a. Point A: (x1, y1)
    b. Point B: (x2, y2)
    c. Point C: (x3, y3)

    Return true is all the three internal angles of the triangle are acute, i.e. the triangle is an acute-angled triangle. Otherwise return false. You can assume that the given three points always form a triangle (that is, they are not colinear).

Note: You must use the isAcute function when implementing isAcuteTriangle.

You will be provided with a main function which does the following:

1. It takes number of testcases as input.
2. For each of testcase, it takes six integers as input : x1, y1, x2, y2, x3, y3. These represent the coordinates for vertices of a triangle.
3. It then calls the function isAcuteTriangle with these six coordinates and prints the result.

ज्यामिति (geometry) में, एक त्रिभुज को न्यूनकोण त्रिभुज (acute-angled triangle) कहा जाता है यदि उसकी सभी आंतरिक कोण (internal angles) 90 डिग्री से कम हों।

आपका काम निम्नलिखित दो फ़ंक्शन्स को implement करना है:

1. bool isAcute(int x1, int y1, int x2, int y2)

दिए गए दो 2D वेक्टर (सदिश): (x1, y1) और (x2, y2) के बीच कोण (angle) न्यूनकोण (acute) है या नहीं, यह निर्धारित करें। यदि कोण 90 डिग्री से कम है, तो true return करें, अन्यथा false। यह डॉट प्रोडक्ट (dot product) की मदद से किया जा सकता है: यदि दो वेक्टर (सदिश) A और B का डॉट प्रोडक्ट धनात्मक (positive) है, तो उनके बीच का कोण न्यूनकोण (acute) होता है।

2. bool isAcuteTriangle(int x1, int y1, int x2, int y2, int x3, int y3) - आपको तीन 2D बिंदु (points) दिए गए हैं जो एक त्रिभुज के शीर्ष बिंदु (vertices) हैं

1. बिंदु A: (x1, y1)
2. बिंदु B: (x2, y2)
3. बिंदु C: (x3, y3)

आपको यह पता करना है कि इस त्रिभुज के सभी तीन कोण न्यूनकोण (acute) हैं या नहीं। यदि तीनों कोण न्यूनकोण हों, तो true return करें; अन्यथा false। आप यह मान सकते हैं कि दिए गए तीनों बिंदु हमेशा एक वैध त्रिभुज बनाते हैं (यानि, वे संरेख नहीं हैं)।

नोट: isAcuteTriangle फ़ंक्शन को लागू करते समय, आपको isAcute फ़ंक्शन का उपयोग करना अनिवार्य है।

मुख्य प्रोग्राम (main function): आपको एक मुख्य प्रोग्राम दिया जाएगा (जो पहले से लिखा हुआ है) —

1. इसमें सबसे पहले टेस्ट केसों की संख्या इनपुट के रूप में ली जाती है।
2. हर टेस्ट केस में छह पूर्णांक इनपुट के रूप में लिए जाते हैं: x1, y1, x2, y2, x3, y3 — जो त्रिभुज के तीन शीर्षों के निर्देशांक (coordinates) होते हैं।
3. इसके बाद यह प्रोग्राम isAcuteTriangle फ़ंक्शन को कॉल करता है और उसका परिणाम प्रिंट करता है।

Assumptions on Input

- Each integer will be in the range [-1000000, 1000000].
- Value of n will be in the range [1, 100]
- प्रत्येक पूर्णांक (integer) का मान [-1000000, 1000000] की सीमा (range) में होगा।
- n का मान [1, 100] की सीमा में होगा।

Note :

- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:

    - cout << "Enter a number:",
    - cout << "The computed answer is", etc.
- cout should be used to print only the computed final output. In addition, do not print unnecessary spaces unless specified in the program.
- **If any hard coding is found, or if any test case passes by merely writing a cout statement and without any logic, then the marks for that test case will NOT be awarded.**

- सामान्य संदेशों को प्रिंट करने के लिए कोई भी C++ स्टेटमेंट न लिखें। उदाहरण के लिए, आपके प्रोग्राम में निम्नलिखित मौजूद नहीं होने चाहिए:
    - cout << "Enter a number:",
    - cout << "The computed answer is", etc.
- cout का उपयोग केवल अंतिम आउटपुट को प्रिंट करने के लिए किया जाना चाहिए। प्रश्न में बताई गई स्पेस के अलावा, कोई भी अनावश्यक स्पेस न प्रिंट करें।
- **यदि कोई हार्ड कोडिंग पाई जाती है, या यदि कोई टेस्ट केस केवल एक cout स्टेटमेंट लिखकर और बिना किसी तर्क के पास हो जाता है, तो उस टेस्ट केस के लिए अंक नहीं दिए जाएँगे।**

Visible Testcases

Input	Output
1 0 0 3 0 0 4	false
2 0 0 3 0 1 2 0 0 1 0 0 1	true false
3 10 10 14 11 12 15 2 3 6 3 4 7 0 0 3 0 0 4	true true false

Hidden Testcases

Input	Output
3 -1000 -1000 -500 -1200 -700 -800 0 0 5 0 0 5 -2 1 3 2 1 -3	true false true
2 0 0 10 1 5 0 0 0 5 0 -1 2	false false
1 1 1 4 2 2 4	true
4 0 0 3 0 0 4 -1 -1 3 -1 -1 4 0 0 10 1 1 10 -3 -2 2 5 3 1	false false true false
5 -10 -10 20 -10 5 40 1000 1000 1005 1002 1001 1004 1 2 4 6 -3 3	true true false true false

0 0 10 1 2 8 -2 1 0 0 3 1	

Solution :

triangle.cpp (To be written by students)

```
C/C++
#include<simplecpp>

bool isAcute(int x1, int y1, int x2, int y2) {
    // Calculate dot product of vectors (x1, y1) and (x2, y2)
    int dotProduct = x1 * x2 + y1 * y2;

    // Angle is acute if dot product is positive
    return dotProduct > 0;
}

bool isAcuteTriangle(int x1, int y1, int x2, int y2, int x3, int y3) {
    // To check if all angles are acute, we need to check the angle at each
vertex

    // Check angle at vertex A (x1, y1)
    // Vectors from A to B and A to C
    int AB_x = x2 - x1, AB_y = y2 - y1;
    int AC_x = x3 - x1, AC_y = y3 - y1;
    bool angleA_acute = isAcute(AB_x, AB_y, AC_x, AC_y);

    // Check angle at vertex B (x2, y2)
    // Vectors from B to A and B to C
    int BA_x = x1 - x2, BA_y = y1 - y2;
    int BC_x = x3 - x2, BC_y = y3 - y2;
    bool angleB_acute = isAcute(BA_x, BA_y, BC_x, BC_y);

    // Check angle at vertex C (x3, y3)
    // Vectors from C to A and C to B
    int CA_x = x1 - x3, CA_y = y1 - y3;
    int CB_x = x2 - x3, CB_y = y2 - y3;
```

```
    bool angleC_acute = isAcute(CA_x, CA_y, CB_x, CB_y);

    // Triangle is acute if all three angles are acute
    return angleA_acute && angleB_acute && angleC_acute;
}
```

main.cpp (To be provided by instructor)

```
C/C++
#include<simplecpp>

// Function to check whether the angle between two vectors is acute or not.
bool isAcute(int x1, int y1, int x2, int y2);

// Function to check whether given triangle is a acute-angle triangle or not.
bool isAcuteTriangle(int x1, int y1, int x2, int y2, int x3, int y3);

main_program{
    int no_of_testcases;
    cin >> no_of_testcases;
    while(no_of_testcases--) {
          int x1, y1, x2, y2, x3, y3;
          cin >> x1 >> y1 >> x2 >> y2 >> x3 >> y3;
          bool result = isAcuteTriangle(x1, y1, x2, y2, x3, y3);
          cout << (result ? "true" : "false") << endl;
    }

    return 0;
}
```

A25 CS101 Lab Quiz 1 - Friday
12th September 2025 - 09:30 hrs to 11:00 hrs
3 Compulsory Questions - 15 Marks (5, 5, 5)

Instructions:

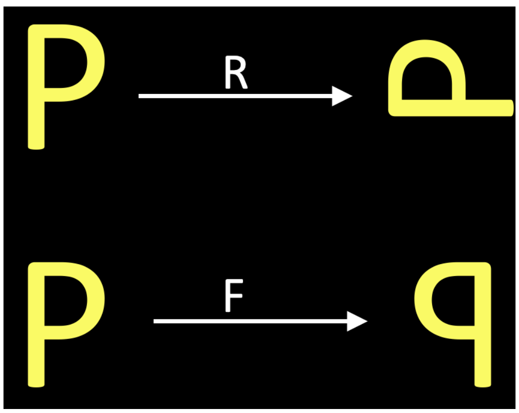
- Keep your ID card on the table for ready reference. If your ID card isn't with you, you won't be allowed to appear for the quiz/exam.
- Keep your phones, tablets, notes, bags, books, etc., near the instructor's platform.
- No clarifications will be provided for any question by anyone (TAs/Instructor).
- **Please note that your answers should NOT include any programming concept that hasn't been covered in the class. Such solutions if found will NOT be graded.**
- Marks will be given for each hidden test case that passes.
- TAs would be around to help you with respect to the logistics like compiling, submitting, etc. They will not help you debug the error or resolve your issues related to syntax/logical errors in your program.

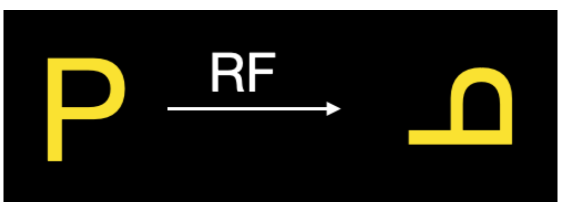
- तैयार संदर्भ के लिए अपना ID card कार्ड table पर रखें। यदि आपका ID card आपके पास नहीं है, तो आपको quiz/परीक्षा में बैठने की अनुमति नहीं दी जाएगी।
- अपने phone, tablet, notes, bag, किताबें आदि प्रशिक्षक के मंच के पास रखें।
- किसी भी प्रश्न के लिए किसी भी (TA/Instructor) द्वारा कोई स्पष्टीकरण नहीं दिया जाएगा।
- **कृपया ध्यान दें कि आपके उत्तरों में कोई भी ऐसा programming concept शामिल नहीं होना चाहिए जिसे कक्षा में पढ़ाया नहीं गया हो। यदि ऐसे समाधान पाए जाते हैं तो उन्हें grade नहीं किया जाएगा।**
- Marks केवल hidden test cases जो pass होते है, उन्हें दिए जाएंगे
- TAs program को compile, और submit सबमिट करने आदि जैसे logistics के संबंध में आपकी मदद करने के लिए मौजूद रहेंगे। वे आपको errors को डीबग करने या आपके program में syntax / logical errors से संबंधित कोई मदद नहीं करेंगे।

Q1. Flips and Rotations

An image editing app has two buttons: rotate (R) and flip (F). R rotates the image anticlockwise by 90 degrees. F flips the image along the vertical axis. Below, the first image shows the two operations. Next image shows the effect of operation R followed by F.

एक इमेज एडिटिंग ऐप में दो बटन हैं: रोटेट (R) और फ़्लिप (F). R इमेज को 90 डिग्री एंटी-क्लॉकवाइज़ (वामावर्त) घुमाता है। F इमेज को दाएँ से बाएँ दर्पण प्रतिबिंब की तरह पलट देता है। नीचे पहली इमेज में दोनों ऑपरेशनों का परिणाम दिखाया गया है। इसके बाद वाली इमेज में दिखाया गया है कि पहले R और उसके बाद F ऑपरेशन करने का क्या परिणाम होता है।





Given a positive integer n, and a sequence of n operations R/F, you have to find out at which steps the image will be in its original orientation.

Write a program that first takes input number n and then a sequence of n operations and outputs 1 or 0 after each operation. After any operation, the output should be 1 if the image is in its original orientation and 0 if the image's orientation is different from the original orientation.

Hint: Operation sequence FF or RRRR will bring the image in its original orientation. The orientations obtained by the operation sequences RFF and FRF will not be the same despite having the same number of flips and rotations. FR will result in the same orientation as RRRF.

आपको एक धनात्मक पूर्णांक n और n ऑपरेशनों (R/F) की एक श्रृंखला (sequence) दी जाएगा। आपको यह पता लगाना है कि किस-किस स्टेप (step) के बाद इमेज अपनी मूल स्थिति (original orientation) में है।

एक प्रोग्राम लिखिए जो पहले इनपुट में संख्या n को ले और फिर n ऑपरेशनों (R या F) की एक श्रृंखला इनपुट के रूप में ले। प्रोग्राम को हर ऑपरेशन के बाद एक संख्या (1 या 0) आउटपुट करना है: यदि इमेज मूल स्थिति में है → 1 प्रिंट करें। यदि इमेज मूल स्थिति से भिन्न है → 0 प्रिंट करें।
संकेत (Hint): कुछ ऑपरेशन श्रृंखलाएँ इमेज को फिर से मूल स्थिति में ले आती हैं: जैसे: FF, या RRRR। लेकिन कुछ श्रृंखलाएँ, जैसे: RFF और FRF, ऐसी हो सकती हैं जिनमें फ्लिप और रोटेशन की संख्या समान हो, लेकिन उनके परिणाम अलग हों. FR का परिणाम RRRF के समान होता है।

For example, for:

- 4 R R R R: 0001
- 2 R F: 00
- 8 R F R R R F R R: 00000001
- 8 R F R F R R F F: 00010000

Explanation: in the last example, the image will not be in the original orientation till the first 3 operations. After the fourth operation, it will come to its original orientation. After 5th operation onwards, it will never be in the original orientation.
व्याख्या: अंतिम उदाहरण में, पहले तीन ऑपरेशन तक इमेज अपनी मूल स्थिति में नहीं आएगी। चौथे ऑपरेशन के बाद, यह अपनी मूल स्थिति में आ जाएगी। पाँचवें ऑपरेशन से आगे, यह कभी भी अपनी मूल स्थिति में नहीं आएगी।

Input Format

- An integer n and a sequence of n operations either R or F.
- एक पूर्णांक n और n ऑपरेशनों की एक श्रृंखला, जहाँ प्रत्येक ऑपरेशन R (Rotate) या F (Flip) होता है।

Output Format

- A sequence of 1's and 0's (without any spaces or line change)
- 1's और 0's की एक श्रृंखला (sequence)। (बिना किसी स्पेस या लाइन चेंज के)

Assumptions on Input

- 1 <= n <= 100
- Sequence of operations is of n length and only contains R or F.

Visible Testcases

Input	Output
4 RRRR	0001
4 FRFR	0001
6 RRRFRF	000000

Hidden Testcases

Input	Output
3 FRF	000
8 RRFRRRFR	00000001

3 FFR	010
4 FRRF	0000
8 RFRRRFRR	00000001

Solution :

```
C/C++
#include<simplecpp>
main_program{
    int n;
    cin>>n;
    //can be 0 1 2 3
    int rotation=0;
    //can be 0 1
    int flip=0;
    while(n--){
        char c;
        cin>>c;
        //cout<<c<<'\n';
        if(c=='R'){
            if(flip==0){
                rotation+=1;
            }
            else{
                rotation-=1;
            }
        }
        if(c=='F'){
            flip+=1;
        }
        flip%=2;
        rotation%=4;

       if(flip==0 && rotation==0){
             cout<<1;
       }
       else{
             cout<<0;
       }
    }
    cout<<'\n';
}
```

Q2. Shrinking O's

Write a program to accept an integer n and print the required pattern for given ‘n’ according to the following examples.

एक प्रोग्राम लिखिए जो एक पूर्णांक n इनपुट ले, और नीचे दिए गए उदाहरणों के अनुसार दिए गए n के लिए वांछित पैटर्न (pattern) प्रिंट करे।

For example, for:

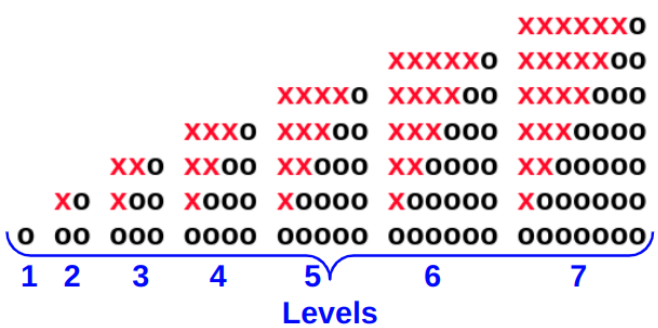
n=5:

```
                    xxxxo
               xxxo xxxoo
          xxo  xxoo xxooo
     xo   xoo  xooo xoooo
o    oo   ooo  oooo ooooo
```

n=3:

```
         xxo
     xo  xoo
o    oo  ooo
```

n=7:



Colors are shown only for explanation, colors are not part of the output.

रंग केवल समझाने के लिए दिखाए गए हैं. रंग आऊटपुट का हिस्सा नहीं हैं.

Explanation

- There are **n** levels in a pattern
- The **`i’th`** level contains **`i`** number of rows and **`i`** number of columns
- Each level contains a pattern consisting of **`x`** and **`o`**
- The first level contains only o
- The next level contains an x in the first row and first column, and the remaining pattern in this level is filled with o.

- The number of x in the first row increases with each level, and the number of x in a particular level decreases in every row until there are none in the last row of the level
- There is exactly one space between each level on a particular row

- पैटर्न में कुल n स्तर (levels) हैं।
- प्रत्येक i वें स्तर (level i) में i पंक्तियाँ (rows) और i स्तंभ (columns) हैं।
- हर स्तर में x और o का एक विशेष पैटर्न होता है:
- पहले स्तर (level 1) में केवल o होते हैं।
- दूसरे स्तर में पहली पंक्ति और पहला स्तंभ x से भरे होते हैं, और बाकी स्थानों पर o होते हैं।
- प्रत्येक अगले स्तर में पहली पंक्ति के x की संख्या बढ़ती जाती है। एक स्तर की हर अगली पंक्ति में x की संख्या घटती जाती है। हर स्तर की आखिरी पंक्ति में कोई x नहीं होता, केवल o होते हैं।
- प्रत्येक पंक्ति में अलग-अलग स्तरों को एक स्पेस से अलग किया गया है।

Input Format

- A single integer n (0 < n < 25).
- एक पूर्णांक n (0 < n < 25).

Output Format

- Print the required pattern.
- दिया गया पैटर्न प्रिंट करें।

Assumptions on Input

- 0 < n < 25

Visible Testcases

<table>
<tr><td>Input</td><td>Output</td></tr>
<tr><td>7</td><td><pre>
                           xxxxxxo
                    xxxxxo xxxxxoo
              xxxxo xxxxoo xxxxooo
         xxxo xxxoo xxxooo xxxoooo
     xxo xxoo xxooo xxoooo xxooooo
  xo xoo xooo xoooo xooooo xoooooo
o oo ooo oooo ooooo oooooo ooooooo
</pre></td></tr>
<tr><td>3</td><td><pre>
     xxo
  xo xoo
o oo ooo
</pre></td></tr>
<tr><td>5</td><td><pre>
              xxxxo
         xxxo xxxoo
     xxo xxoo xxooo
  xo xoo xooo xoooo
o oo ooo oooo ooooo
</pre></td></tr>
</table>

Hidden Testcases

Input	Output
1	o

15	
	xxxxxxxxxxxxxxo
	
	xxxxxxxxxxxxxo xxxxxxxxxxxxxoo
	
	xxxxxxxxxxxxo xxxxxxxxxxxxoo xxxxxxxxxxxxooo
	
	xxxxxxxxxxxo xxxxxxxxxxxoo xxxxxxxxxxxooo xxxxxxxxxxxoooo
	
	xxxxxxxxxxo xxxxxxxxxxoo xxxxxxxxxxooo xxxxxxxxxxoooo xxxxxxxxxxooooo
	                                                      xxxxxxxxxo
	xxxxxxxxxoo xxxxxxxxxooo xxxxxxxxxoooo xxxxxxxxxooooo xxxxxxxxxoooooo
	                                            xxxxxxxxo xxxxxxxxoo
	xxxxxxxxooo xxxxxxxxoooo xxxxxxxxooooo xxxxxxxxoooooo xxxxxxxxooooooo
	                                   xxxxxxxo xxxxxxxoo xxxxxxxooo
	xxxxxxxoooo xxxxxxxooooo xxxxxxxoooooo xxxxxxxooooooo xxxxxxxoooooooo
	                           xxxxxxo xxxxxxoo xxxxxxooo xxxxxxoooo
	xxxxxxooooo xxxxxxoooooo xxxxxxooooooo xxxxxxoooooooo xxxxxxooooooooo
	                    xxxxxo xxxxxoo xxxxxooo xxxxxoooo xxxxxooooo
	xxxxxoooooo xxxxxooooooo xxxxxoooooooo xxxxxooooooooo xxxxxoooooooooo
	              xxxxo xxxxoo xxxxooo xxxxoooo xxxxooooo xxxxoooooo
	xxxxooooooo xxxxoooooooo xxxxooooooooo xxxxoooooooooo xxxxooooooooooo
	         xxxo xxxoo xxxooo xxxoooo xxxooooo xxxoooooo xxxooooooo
	xxxoooooooo xxxooooooooo xxxoooooooooo xxxooooooooooo xxxoooooooooooo
	     xxo xxoo xxooo xxoooo xxooooo xxoooooo xxooooooo xxoooooooo
	xxooooooooo xxoooooooooo xxooooooooooo xxoooooooooooo xxooooooooooooo
	  xo xoo xooo xoooo xooooo xoooooo xooooooo xoooooooo xooooooooo
	xoooooooooo xooooooooooo xoooooooooooo xooooooooooooo xoooooooooooooo
	o oo ooo oooo ooooo oooooo ooooooo oooooooo ooooooooo oooooooooo
	ooooooooooo oooooooooooo ooooooooooooo oooooooooooooo ooooooooooooooo

8	xxxxxxxo
	                           xxxxxxo xxxxxxoo
	                    xxxxxo xxxxxoo xxxxxooo
	              xxxxo xxxxoo xxxxooo xxxxoooo
	         xxxo xxxoo xxxooo xxxoooo xxxooooo
	     xxo xxoo xxooo xxoooo xxooooo xxoooooo
	  xo xoo xooo xoooo xooooo xoooooo xooooooo
	o oo ooo oooo ooooo oooooo ooooooo oooooooo

6	xxxxxo
	              xxxxo xxxxoo
	         xxxo xxxoo xxxooo
	     xxo xxoo xxooo xxoooo
	  xo xoo xooo xoooo xooooo
	o oo ooo oooo ooooo oooooo

2	xo o oo

Solution :

```
C/C++
#include<simplecpp>
main_program{
    int n;
    cin>>n;
    int x=n-1;
    for(int i=n; i>0; i--){
        //printing spaces
        int spaces=((x+1)*x)/2+x;
        while(spaces--){
            cout<<' ';
        }
        //printing row
        for(int j=1;j<=n-x;j++){
            //printing elements
            for(int k=0;k<x+j;k++){
                if(k<x)
                cout<<'x';
                else
                cout<<'o';
            }
            cout<<' ';
        }
        x-=1;
        cout<<'\n';
    }
}
```

Q3. Knight Movement

In this question you need to write only two functions in the file **`knight.cpp`** as described below. A main program (**`main.cpp`**) is already provided and is not editable.

इस प्रश्न में आपको केवल दो फ़ंक्शन एक फ़ाइल **knight.cpp** में लिखने हैं, जैसा कि नीचे बताया गया है। एक main program (**main.cpp**) पहले से दिया गया है और उसमें कोई बदलाव नहीं किया जा सकता।

Chess is played on a board divided into an 8x8 grid of 64 equal-sized squares.

Chessboard positions are written as coordinates, starting from (1,1) to (1,8) in the first row, and continuing up to (8,1) to (8,8) in the last row.

Players move playing pieces named pawns, rooks, knights, bishops, a queen, and a king, each moving in its own unique way.

A knight in chess moves in an “L” shape:

1. two squares in one direction (horizontal or vertical), then
2. one square perpendicular to that.

For example in a Chess board :
From position (4,4), a knight can move to:
(6,5), (6,3), (2,5), (2,3), (5,6), (5,2), (3,6), (3,2) (highlighted in pink color, see the image)

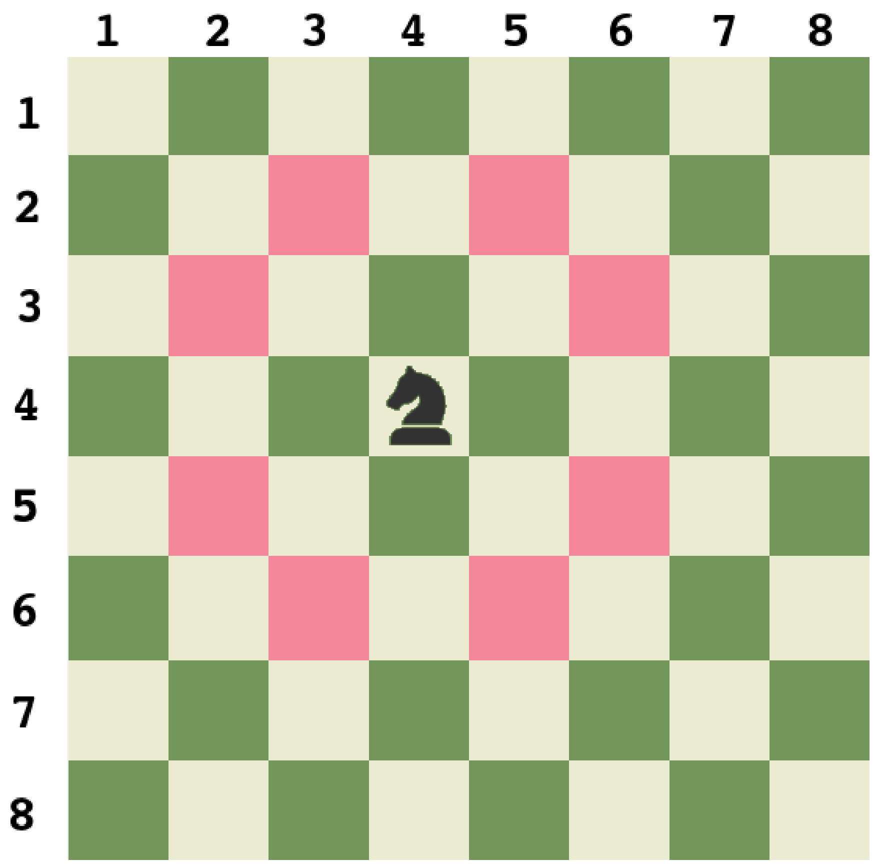
शतरंज (Chess) बोर्ड 8×8 के ग्रिड में बँटा होता है, जिसमें कुल 64 समान आकार के खाने (squares) होते हैं।
शतरंज की स्थितियाँ (positions) निर्देशांक (coordinates) के रूप में लिखी जा सकती हैं — पहली पंक्ति में (1,1) से (1,8) तक, और अगली पंक्तियों के निर्देशांक इसी तरह तय करते हुए, आखिरी पंक्ति में (8,1) से (8,8) तक।

खिलाड़ी अपने मोहरों (pieces) को चलते हैं, जिनके नाम होते हैं: प्यादा (pawn), हाथी (rook), घोड़ा (knight), ऊँट (bishop), रानी (queen) और राजा (king) — हर मोहरा अपने एक विशेष तरीके से चलता है।

शतरंज में घोड़ा (Knight) 'L' आकार में चलता है:

1. पहले किसी दिशा में दो खाने (squares) चलता है — (क्षैतिज या ऊर्ध्वाधर),
2. फिर उस दिशा के लंबवत (perpendicular) में एक खाना चलता है।

उदाहरण के लिए:
अगर घोड़ा खाने (4,4) में है, तो वह निम्नलिखित खानों पर जा सकता है:
(6,5), (6,3), (2,5), (2,3), (5,6), (5,2), (3,6), (3,2)
(इन सभी को चित्र में गुलाबी रंग (pink) से दर्शाया गया है)



Main.cpp is already coded for you:

- User inputs the number of testcases : **t**. For each testcase the following runs :
  - It reads four integers, denoting the starting position (x1, y1) and final position (x2, y2) from the user,
  - Validates the input coordinates to check if they are not falling outside the 8x8 chessboard.
  - Calls the knightMove() function to check if the knight can move from the starting position to the final position in one move; if false, it calls the knight2Moves() function to check if the movement is possible in exactly two moves.
  - Prints the appropriate message.

Your task is to write the **knight.cpp file** and complete the functionality of the program. You are supposed to do the following :

1. Write a boolean function "**bool knightMove(int x1, int y1, int x2, int y2)**" that takes two distinct chessboard positions and checks whether the knight can move from the first position (x1, y1) to the second position (x2, y2) in exactly one move. Return true if the move is possible, and false if not.

   You can assume that each argument is a number between 1 and 8 and (x1, y1) is not the same as (x2, y2).

2. Write another boolean function "**bool knight2Moves(int x1, int y1, int x2, int y2)**" which **MUST USE** the function **knightMove.** It checks whether the knight can move from the first position (x1, y1) to the second (x2, y2) in exactly two moves. Returns true if the position is reachable in exactly two moves, false if not.
   You can assume that each argument is a number between 1 and 8.

आपके लिए main program पहले से Main.cpp में लिखा हुआ है:

- पहले यूजर टेस्ट केसों की संख्या t इनपुट करता है। फिर हर टेस्ट केस के लिए निम्नलिखित काम किए जाते हैं।
  - चार पूर्णांक पढ़े जाते हैं, जो कि आरंभिक स्थिति (x1, y1) और अंतिम स्थिति (x2, y2) को दर्शाते हैं।
  - इनपुट निर्देशांकों को जांचा जाता है कि वे 8x8 के शतरंज बोर्ड से बाहर तो नहीं हैं।
  - इसके बाद knightMove() फ़ंक्शन को कॉल किया जाता है ताकि यह पता चले कि क्या घोड़ा (knight) सिर्फ एक चाल में (x1, y1) से (x2, y2) जा सकता है।
  - यदि ऐसा संभव नहीं होता, तो knight2Moves() फ़ंक्शन को कॉल किया जाता है ताकि पता चले कि यह movement ठीक दो चालों में संभव है या नहीं।
  - फिर एक उपयुक्त संदेश (appropriate message) प्रिंट किया जाता है।

आपका काम एक knight.cpp फ़ाइल लिखना और उसमें दो फंन्क्शन्स को implement करना है।

1. एक Boolean function लिखिए: bool knightMove(int x1, int y1, int x2, int y2) । यह फ़ंक्शन शतरंज बोर्ड की दो अलग-अलग स्थितियाँ लेता है, और यह देखता है कि क्या knight (घोड़ा) पहली स्थिति (x1, y1) से दूसरी स्थिति (x2, y2) तक सिर्फ एक चाल में जा सकता है। यदि जा सकता है, तो true return करें; अन्यथा false।
   आप मान सकते हैं कि सभी arguments 1 से 8 के बीच हैं, और (x1, y1) ≠ (x2, y2) है।

2. एक दूसरा Boolean function लिखिए: bool knight2Moves(int x1, int y1, int x2, int y2) जिसमें कि **`knightMove`** फ़ंक्शन का उपयोग होना ही चाहिए। यह फ़ंक्शन देखता है कि क्या knight (घोड़ा) (x1, y1) से (x2, y2) तक ठीक दो चालों में पहुँच सकता है। यदि हाँ, तो true return करता है; अन्यथा false।
   आप मान सकते हैं कि सभी arguments 1 से 8 के बीच होंगे।

Note

- Do not write any C++ statements for printing general messages. For example, the following should NOT be present in your program:
  - cout << "Enter a number:",
  - cout << "The computed answer is", etc.
- cout should be used to print only the computed final output. In addition, do not print unnecessary spaces unless specified in the program.
- **If any hard coding is found, or if any test case passes by merely writing a cout statement and without any logic, then the marks for that test case will NOT be awarded.**
- सामान्य संदेशों को प्रिंट करने के लिए कोई भी C++ स्टेटमेंट न लिखें। उदाहरण के लिए, आपके प्रोग्राम में निम्नलिखित मौजूद नहीं होने चाहिए:
  - cout << "Enter a number:",
  - cout << "The computed answer is", etc.
- cout का उपयोग केवल अंतिम आउटपुट को प्रिंट करने के लिए किया जाना चाहिए। प्रश्न में बताई गई स्पेस के अलावा, कोई भी अनावश्यक स्पेस न प्रिंट करें।
- **यदि कोई हार्ड कोडिंग पाई जाती है, या यदि कोई टेस्ट केस केवल एक cout स्टेटमेंट लिखकर और बिना किसी तर्क के पास हो जाता है, तो उस टेस्ट केस के लिए अंक नहीं दिए जाएँगे।**

Visible Test Cases

1 4 4 5 6	Knight can reach the target in 1 move.
2 3 3 5 3 3 3 5 2	Knight can reach the target in 2 moves. Knight can reach the target in 1 move.

3 3 3 3 2 3 3 4 1 4 1 6 5	Knight cannot reach the target in 1 or 2 moves. Knight can reach the target in 1 move. Knight can reach the target in 2 moves.

Hidden Test Cases

2 1 7 1 6 1 7 7 7	Knight cannot reach the target in 1 or 2 moves. Knight cannot reach the target in 1 or 2 moves.
1 1 7 3 6	Knight can reach the target in 1 move.
3 1 1 1 2 1 1 2 1 2 1 1 2	Knight cannot reach the target in 1 or 2 moves. Knight cannot reach the target in 1 or 2 moves. Knight can reach the target in 2 moves.
1 1 2 5 4	Knight can reach the target in 2 moves.
5 1 1 5 1 5 6 3 2 3 2 1 4 4 4 5 6 7 3 4 1	Knight can reach the target in 2 moves. Knight can reach the target in 2 moves. Knight cannot reach the target in 1 or 2 moves. Knight can reach the target in 1 move. Knight cannot reach the target in 1 or 2 moves.

Solution (2 files):

main.cpp (to be provided by the instructor)

```
#include <simplecpp>


bool knightMove(int x1, int y1, int x2, int y2);
bool knight2Moves(int x1, int y1, int x2, int y2);
main_program {
    //number of testcases
    int testcases;
    cin>>testcases;

    while(testcases--){
        int x1, y1, x2, y2;


        //Enter starting position (x1 , y1)
```

```
        cin >> x1 >> y1;


        //Enter target position (x2 , y2)
        cin >> x2 >> y2;


        // Validate input
        if (x1 < 1 || x1 > 8 || y1 < 1 || y1 > 8 ||
            x2 < 1 || x2 > 8 || y2 < 1 || y2 > 8) {
            cout << "Invalid input: all coordinates must be between 1 and 8.\n";
            return 1;
        }


        if (knightMove(x1, y1, x2, y2)) {
            cout << "Knight can reach the target in 1 move.\n";
        }
        else if (knight2Moves(x1, y1, x2, y2)) {
            cout << "Knight can reach the target in 2 moves.\n";
        }
        else {
            cout << "Knight cannot reach the target in 1 or 2 moves.\n";
        }

    }


    return 0;
}
```

knight.cpp (to be written by the student)

```
// Knight move explanation:
// A knight moves in an "L" shape:
```

```cpp
// - two squares in one direction (horizontal or vertical)
//   followed by one square perpendicular.
// That means the possible moves from (x1,y1) are:
// (x1 ± 2, y1 ± 1) and (x1 ± 1, y1 ± 2).
#include <simplecpp>


bool knightMove(int x1, int y1, int x2, int y2) {
    int dx = abs(x1 - x2);
    int dy = abs(y1 - y2);

    // valid knight move: (2,1) or (1,2)
    return (dx == 2 && dy == 1) || (dx == 1 && dy == 2);
}

bool knight2Moves(int x1, int y1, int x2, int y2) {
    // Check all intermediate squares
    for (int i = 1; i <= 8; i++) {
        for (int j = 1; j <= 8; j++) {
            if (knightMove(x1, y1, i, j) && knightMove(i, j, x2, y2)) {
                return true;
            }
        }
    }
    return false;
}
```