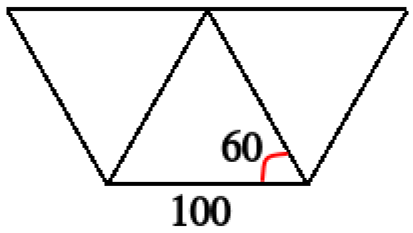
CS101 Lab Quiz 1 - Batch D
16 Feb 2024 - 20:45 hrs to 22:15 hrs
4 Compulsory Questions - 40 Marks

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.
- Rough sheets will be provided to you.
- Create a folder on your `Desktop` and name it `submission_YourRollNumber`
  E.g.: If my roll number is `23k1234` then my folder name is **`submission_23k1234`**
- Create all four programs in the newly created folder.
- Name the program files as mentioned in this pdf only.
- No clarifications will be provided for any question by anyone (TAs/Instructor). When in doubt, make suitable assumptions, state them clearly as comments in your program file itself, and proceed to solve the problem.
- Please note that your answers should NOT include any programming concept that hasn't been covered in the class. You are not allowed to use arrays, functions, strings, or any advanced concepts of C/C++. Such solutions will NOT be graded.
- Marks will be given for each hidden test case that passes.

Filename: LQ01_D_Q1.cpp **(10 marks)**

Write a program to create the following figure. Ensure that the entire figure remains visible within the screen area. The length of the sides of each triangle is 100 and the angles are 60. These are denoted in the figure itself. Do not draw the length or angles; they are just for illustration purposes.

Rubrics:
- Its ok if the size of the length is not exact. It should look similar to the one below



Solution Program:
```
#include<simplecpp>
main_program {
        turtleSim();

        repeat(3) {
            forward(100);
            left(120);
        }

        forward(100);

        left(60);
        forward(100);

        left(120);
        forward(200);

        left(120);
        forward(100);

        wait(2);
}
```

Filename: LQ01_D_Q2.cpp **(8 marks)**

Write a C++ program to accept two floating point numbers from the user, a and b, and evaluate the expression and print the result: $\mathbf{(a^5 – 3a^4b + 4a^2b^3 – 5ab^4 + b^5) / (a – b)}$

Input Format:
The input consists of two floating point values, a and b

Output Format:
Print the computed value of the expression

Constraints (Assume)
-10 <= a, b <=10
a is not equal to b
(a-b) is never 0

***Note**:*

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

b. **cout << "The computed answer is"**, etc.

In addition, **do not** print unnecessary spaces unless specified in the program.

- You just have to print the result

Practice Test Cases

Input (a,b)	Output
2 1	-9
-2 3	-261.8
3 1	11
2.5 1.5	-49.4375

Evaluate Test case

Input	**Output**	**Marks**
-1 1	-3	2
7 9	37861	2
-2 -1	-9	2
-3 5	-1942.75	2

Solution Program :

```
#include <simplecpp>
main_program {
    float a, b;
    cin>>a>>b;
    float result;

    result = ( (a*a*a*a)
             - (3*a*a*a*b)
             + (4*a*a*b*b*b)
             - (5*a*b*b*b*b)
             + (b*b*b*b*b) )
            / (a - b);

    cout <<  result;

    return 0;
}
```

Filename: LQ01_D_Q3.cpp **(10 marks)**
Write a C++ program that takes four integers as input and determines if they are in ascending order, descending order, or any arbitrary order.

Instructions:

1. The program should read four integers (a, b, c, and d) from the user. **All numbers are unique.**
2. Based on the values of a, b, c, and d, the program should determine if the numbers are in ascending order, descending order, or arbitrary (i.e. if the numbers do not follow either ascending or descending order).

Input Format:
The input consists of four integers, a, b, c, and d, each separated by space.

Output Format:
Print exactly one of the following words based on the order of the input numbers:

- ascending
- descending
- arbitrary

Constraints (Assume)
-1000 <= a, b, c, d <= 1000

Note:

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

  In addition, **do not** print unnecessary spaces unless specified in the program.

Practice Test Cases

Input (a,b,c,d)	Output
1 2 3 4	ascending
4 3 2 1	descending
1 3 2 4	arbitrary
-10 -5 0 3	ascending
100 50 0 -1000	descending
-100 89 21 -500	arbitrary

Evaluate Test case

Input	**Output**	2 marks per test case Do not cut marks if there are spelling mistakes.
1 2 5 8	ascending	
42 -12 0 9	arbitrary	
6 3 2 -1	descending	
9 8 2 6	arbitrary	
-10 -8 -4 -1	ascending	

Solution Program :

```
#include <simplecpp>

main_program {
    int a, b, c, d;
    cin >> a >> b >> c >> d;

    // Check for ascending order
    if (a < b && b < c && c < d) {
        cout << "ascending";
    }
    // Check for descending order
    else if (a > b && b > c && c > d) {
        cout << "descending";
    }
    // If neither ascending nor descending, then it's arbitrary
    else {
        cout << "arbitrary";
    }

    return 0;
}
```

Filename: LQ01_D_Q4.cpp **(14 marks)**

Write a C++ program to take two inputs from the user, the year N, and d, the day of the week that January 1st falls on. Assume that the days are encoded as follows: (Mon: 1, Tues: 2, Wed: 3, … Sun: 7). Based on this, print all the dates on which Sunday falls on in that year.

You must also take into account if the year is a **leap year or not**. The year is said to be a leap year if the year is divisible by four, except for end-of-century years where the year should be divisible by 400. For example, 2000 was a leap year as its divisible by 400, but 1900 was not a leap year (i.e. 1900 is divisible by 4 but not by 400).

Example:
Let's say you were interested in finding out all the dates when Sunday is going to be this year i.e. 2024, then the input to your code should be:
2024 1

Explanation for the above input: 2024 is the year and 1st Jan happened to be on Monday this year, hence was number-encoded as 1

The output should list the dates of all Sundays in the year N, month-wise on a new line. Each month's Sundays should be separated by a space on the same line.

First find, if it is a leap year

- It is not an end-of-century year
- So, check if 2024 is divisible by 4. Since it is, 2024 is a leap year

For the example for the input: 2024 1, the following should be the output:

```
Month 1: 7 14 21 28
Month 2: 4 11 18 25
Month 3: 3 10 17 24 31
Month 4: 7 14 21 28
Month 5: 5 12 19 26
Month 6: 2 9 16 23 30
Month 7: 7 14 21 28
Month 8: 4 11 18 25
Month 9: 1 8 15 22 29
Month 10: 6 13 20 27
Month 11: 3 10 17 24
Month 12: 1 8 15 22 29
```

Explanation of the output above: Above listed are the exact calendar dates of all the Sundays appearing this year i.e. 2024, written in a space-separated fashion for a month and on a new line for every month. Note also that you should print "Month x:" where x∈{1,2,..12} as a prefix to each line representing a month in a year.

Input Format:
The input consists of a year and the day of the week that January 1st falls on (use the number encoding for the day)

Output Format:
Print the dates of all Sundays in the specified year as mentioned above

Constraints (Assume)
1000 <= N (Year) <= 3500
1 <= d <= 12

Note:

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

  In addition, **do not** print unnecessary spaces unless specified in the program.
- You just have to print the Month-wise dates of all Sundays in the specified format.

Practice Test Cases:

Input	Output	Explanation
2029 1	Month 1: 7 14 21 28 Month 2: 4 11 18 25 Month 3: 4 11 18 25 Month 4: 1 8 15 22 29 Month 5: 6 13 20 27 Month 6: 3 10 17 24 Month 7: 1 8 15 22 29 Month 8: 5 12 19 26 Month 9: 2 9 16 23 30 Month 10: 7 14 21 28 Month 11: 4 11 18 25 Month 12: 2 9 16 23 30	Not a leap year
2026 4	Month 1: 4 11 18 25 Month 2: 1 8 15 22 Month 3: 1 8 15 22 29 Month 4: 5 12 19 26 Month 5: 3 10 17 24 31 Month 6: 7 14 21 28 Month 7: 5 12 19 26 Month 8: 2 9 16 23 30 Month 9: 6 13 20 27	Not a leap year

	Month 10: 4 11 18 25 Month 11: 1 8 15 22 29 Month 12: 6 13 20 27	
1948 4	Month 1: 4 11 18 25 Month 2: 1 8 15 22 29 Month 3: 7 14 21 28 Month 4: 4 11 18 25 Month 5: 2 9 16 23 30 Month 6: 6 13 20 27 Month 7: 4 11 18 25 Month 8: 1 8 15 22 29 Month 9: 5 12 19 26 Month 10: 3 10 17 24 31 Month 11: 7 14 21 28 Month 12: 5 12 19 26	Leap year
2000 6	Month 1: 2 9 16 23 30 Month 2: 6 13 20 27 Month 3: 5 12 19 26 Month 4: 2 9 16 23 30 Month 5: 7 14 21 28 Month 6: 4 11 18 25 Month 7: 2 9 16 23 30 Month 8: 6 13 20 27 Month 9: 3 10 17 24 Month 10: 1 8 15 22 29 Month 11: 5 12 19 26 Month 12: 3 10 17 24 31	End of Century year and Leap year
1900 1	Month 1: 7 14 21 28 Month 2: 4 11 18 25 Month 3: 4 11 18 25 Month 4: 1 8 15 22 29 Month 5: 6 13 20 27 Month 6: 3 10 17 24 Month 7: 1 8 15 22 29 Month 8: 5 12 19 26 Month 9: 2 9 16 23 30 Month 10: 7 14 21 28 Month 11: 4 11 18 25 Month 12: 2 9 16 23 30	End of Century year and But NOT a leap year

Evaluate Test case

- **2 marks per test case**
- **Do not cut marks if there is a spelling mistake**
- **Do not cut marks if there is slight change in the format as long as month number, and the sundays are printed (one month on one line)**

Input	**Output**	**Marks/Explanation**
2030 2	Month 1: 6 13 20 27 Month 2: 3 10 17 24 Month 3: 3 10 17 24 31 Month 4: 7 14 21 28 Month 5: 5 12 19 26 Month 6: 2 9 16 23 30 Month 7: 7 14 21 28 Month 8: 4 11 18 25 Month 9: 1 8 15 22 29 Month 10: 6 13 20 27 Month 11: 3 10 17 24 Month 12: 1 8 15 22 29	2 Not leap year
2032 4	Month 1: 4 11 18 25 Month 2: 1 8 15 22 29 Month 3: 7 14 21 28 Month 4: 4 11 18 25 Month 5: 2 9 16 23 30 Month 6: 6 13 20 27 Month 7: 4 11 18 25 Month 8: 1 8 15 22 29 Month 9: 5 12 19 26 Month 10: 3 10 17 24 31 Month 11: 7 14 21 28 Month 12: 5 12 19 26	2 Leap year
2040 7	Month 1: 1 8 15 22 29 Month 2: 5 12 19 26 Month 3: 4 11 18 25 Month 4: 1 8 15 22 29 Month 5: 6 13 20 27 Month 6: 3 10 17 24 Month 7: 1 8 15 22 29 Month 8: 5 12 19 26 Month 9: 2 9 16 23 30 Month 10: 7 14 21 28 Month 11: 4 11 18 25	2 Leap year

	Month 12: 2 9 16 23 30	
2041 2	Month 1: 6 13 20 27 Month 2: 3 10 17 24 Month 3: 3 10 17 24 31 Month 4: 7 14 21 28 Month 5: 5 12 19 26 Month 6: 2 9 16 23 30 Month 7: 7 14 21 28 Month 8: 4 11 18 25 Month 9: 1 8 15 22 29 Month 10: 6 13 20 27 Month 11: 3 10 17 24 Month 12: 1 8 15 22 29	2 Not leap year
3200 6	Month 1: 2 9 16 23 30 Month 2: 6 13 20 27 Month 3: 5 12 19 26 Month 4: 2 9 16 23 30 Month 5: 7 14 21 28 Month 6: 4 11 18 25 Month 7: 2 9 16 23 30 Month 8: 6 13 20 27 Month 9: 3 10 17 24 Month 10: 1 8 15 22 29 Month 11: 5 12 19 26 Month 12: 3 10 17 24 31	End of century leap year
1800 3	Month 1: 5 12 19 26 Month 2: 2 9 16 23 Month 3: 2 9 16 23 30 Month 4: 6 13 20 27 Month 5: 4 11 18 25 Month 6: 1 8 15 22 29 Month 7: 6 13 20 27 Month 8: 3 10 17 24 31 Month 9: 7 14 21 28 Month 10: 5 12 19 26 Month 11: 2 9 16 23 30 Month 12: 7 14 21 28	End of century not leap year
3300 5	Month 1: 3 10 17 24 31 Month 2: 7 14 21 28 Month 3: 7 14 21 28	End of century not leap year

	Month 4: 4 11 18 25 Month 5: 2 9 16 23 30 Month 6: 6 13 20 27 Month 7: 4 11 18 25 Month 8: 1 8 15 22 29 Month 9: 5 12 19 26 Month 10: 3 10 17 24 31 Month 11: 7 14 21 28 Month 12: 5 12 19 26	

Solution Program :

```
#include <simplecpp>
main_program {
    int year, startDay;
    cin >> year >> startDay;
    int daysInWeek = 7;
    int daysSoFar = 0;
    bool isLeapYear;
    if ((year % 4 == 0 && year % 100 != 0) || (year % 400 == 0))
        isLeapYear = true;
    else
        isLeapYear = false;
    for (int month = 1; month <= 12; month++) {
        int daysInCurrMonth;
        if (month == 2) {
            daysInCurrMonth = isLeapYear? 29 : 28;
        } else if (month == 4 || month == 6 || month == 9 || month ==
11) {
            daysInCurrMonth = 30;
        } else {
            daysInCurrMonth = 31;
        }
        cout << "Month " << month << ":";
        for (int day = 1; day <= daysInCurrMonth; day++) {
            daysSoFar++;
            if ((daysSoFar + startDay - 1) % daysInWeek == 0) {
                cout << " " << day;
            }
        }
        if(month!=12)
            cout << endl;
    }
    return 0;
}
```