# 2019 NORTHERN INDIANA CODING COMPETITION

April 13, 2019


COMPUTER SCIENCE AND INFORMATICS
COLLEGE OF LIBERAL ARTS AND SCIENCES
INDIANA UNIVERSITY SOUTH BEND

# Round Two

**Problem 1.** This is the formula to determine the day of a week given the date information

W = (k + floor(2.6*m - 0.2) - 2*C + Y + floor(Y/4.0) + floor(C/4.0)) % 7

- floor(x) function returns the smallest integer that is greater or equal to x; for example, floor(4.3) returns 4, floor(2) returns 2.
- % is the modular operator; x%y returns the remainder of the division of x by y; for example, 12%5 returns 2, 12%4 returns 0.
- k is day (1 to 31).
- m is month (1 = March, ..., 10 = December, 11 = January, 12 = February), treat January & February as months of the preceding year.
- C is century (1987 has C = 19, 2000 has C=20 except C=19 for January & February, because January & February of 2000 should be treated as months of the preceding year, 1999)
- Y is year (1987 has Y = 87 except Y = 86 for January & February, because January & February of 1987 should be treated as months of the preceding year, 1986)
- W is weekday (0 = Sunday, 1=Monday, 2=Tuesday, 3=Wednesday, 4=Thursday, 5=Friday, 6 = Saturday), if W is less than 0, add 7.

Write a program that will accept **a valid date** in format **mm dd yyyy** (yyyy is in range 1000 to 9999) as input and display the **weekday** of this date. You can only use necessary built-in math functions, **no built-in time or calendar functions are allowed**.

Sample input (red color in **mm dd yyyy** format, numbers are separated by a space) and output (blue color):

Input: 04 13 2019
Output: Saturday

Input: 01 01 2000
Output: Saturday

Input: 02 28 2019
Output: Thursday

Input: 12 27 1980
Output: Saturday

**Problem 2.** Tic-Tac-Toe is a two-player game played on a 3x3 board. The players take turns placing tokens on the table. The goal is to form a line of 3 token, horizontally, vertically, or diagonally.

Given a configuration of a Tic-Tac-Toe game, and a player whose turn it is to play, determine if this player can make a move that wins the game. If several such moves are possible, return the position that has smaller row number, and if row numbers are same, return the position that has smaller column number.

The board, containing integers, will be provided as a 3x3 matrix with 0 representing empty spaces, 1 a token of the first player, and 2 a token of the second player. The answer should be provided as row and column numbers, each taking values going from 0 to 2. If no such move is possible, -1 -1 should be returned.

Sample input (red color, numbers are separated by a space on the same line) and output (blue color, numbers are separated by a space):

**Note**: (1) The number after the 3x3 board indicates the turn of the player; (2) Assume the row and column ranges are [**0, 1, 2**].

Input: 0 2 1
0 1 2
0 0 0
1
Output: 2 0

Input: 1 0 0
0 1 0
2 0 2
2
Output: 2 1

Input: 0 1 0
2 0 0
0 0 0
1
Output: -1 -1

Input: 2 1 2
0 1 1
0 0 2
1
Output: 1 0

Input: 1 0 2
0 0 2
1 2 1
1
Output: 1 0

**Problem 3.** Given an array of **maximum size 15** containing positive integer numbers (1 to 20), a block is defined as an interval in the array where the value is contiguous. For example, in

1 2 2 2 2 3 3 3 1

there are 4 blocks: 1, then 2 2 2 2, followed by 3 3 3, and then another 1 at the end. Removing a block would give you a score equal to the **square** of its **length**, and the remaining values are collapsed together. In the array above, removing the block 3 3 3 would give a score of 9 and reduce the array to

1 2 2 2 2 1

Subsequent moves can be made until the array becomes empty. In this example, removing the block of 2 2 2 2 would add a score of 16, and then leave the array 1 1. Removing the last block would give a score of 4, for a total from the three moves of 29.

Write a program that accepts such **an integer array** of maximum size 15 and displays **the maximum score** that can be achieved by removing blocks in any order from the array.

Sample input (red color, **first line is array size and the second line is the array**, numbers are separated by a space) and output (blue color):

Input: 9
1 2 2 2 2 3 3 3 1
Output: 29

Input: 6
1 3 3 3 5 5
Output: 14

Input: 9
2 2 3 3 3 1 2 2 1
Output: 27