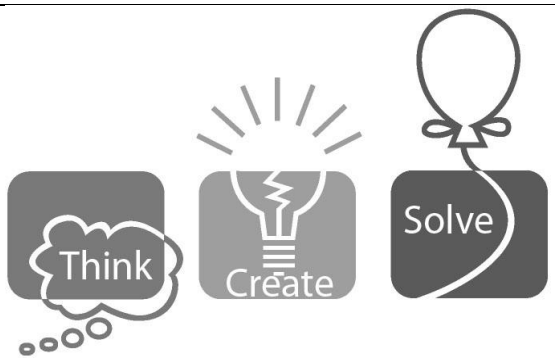
University of Central Florida



2020 Local Programming Contest (Round 1A)

Problems			
Problem#	Difficulty Level	Filename	Problem Name
1	Easy	bills	Briefcases Full of Money
2	Easy	mind	A Game Called Mind
3	Medium	unique	Unique Values
4	Medium	fish	Gone Fishing
5	Medium	sumfunc	Sum of a Function
6	Medium	hangglide	Hang Gliding
7	Medium	trading	Trading Cards
8	Medium-Hard	median	Median Inversion String
9	Medium-Hard	checklist	Check List

Call your program file: filename.c, filename.cpp, filename.java, or filename.py

For example, if you are solving Briefcases Full of Money:

Call your program file: bills.c, bills.cpp, bills.java, or bills.py

UCF Local Contest (Round 1A) — September 5, 2020

Briefcases Full of Money

filename: `bills`
Difficulty Level: `Easy`
Time Limit: `5 seconds`

It is your birthday party and the six UCF programming team coaches arrive, each holding a briefcase containing money (gift) for you. The coaches were planning to give you the six briefcases but Dr. "O" points out that the team needs money to travel to World Contest Finals! So, you get to choose one briefcase, i.e., you are not getting all the briefcases (sorry).

Each briefcase contains a stack of bills; each briefcase containing one of the 6 denominations $1, $5, $10, $20, $50, $100, i.e., first briefcase contains only $1 bills, second contains only $5 bills, third only $10 bills, fourth only $20 bills, fifth only $50 bills, and sixth only $100 bills. You, of course, want to pick the one with the highest total amount.

The Problem:

You can randomly pick one briefcase but, as a programmer, you trust your coding skills more than chance and decide to write a program to help you pick the briefcase with the highest amount.

The Input:

There is only one input line; it contains six integers (each integer between 1 and 1000, inclusive). These integers represent, respectively, the number of $1, $5, $10, $20, $50, $100 bills.

The Output:

Output which briefcase to choose by printing the denomination in that briefcase (1, 5, 10, 20, 50, 100). If two or more briefcases have the highest total, pick (print) the one with fewest number of bills among those briefcases since that one is lighter!

Sample Input	**Sample Output**
`84 111 2 3 2 3`	`5`
`200 3 20 5 4 1`	`50`
`1000 2 2 2 2 2`	`1`

Explanation of the second Sample Input/Output: Three briefcases ($1, $10, $50) have the highest total ($200) so the output is the briefcase with the fewest number of bills.

UCF Local Contest (Round 1A) — September 5, 2020

A Game Called Mind

filename: `mind`
Difficulty Level: `Easy`
Time Limit: `5 seconds`

This problem deals with a simplified version of a game called "Mind". In this game, a group tries to cooperatively solve a problem. Assume:

- There are 2-6 players (called A, B, C, D, E, F).
- Each player has 1-9 cards in sorted order.
- Each card value is between 10 and 99 (inclusive).
- There are no duplicate values, i.e., a card (number) is in at most one hand.

The Problem:

The objective of the game is for all the players to put all the cards down in sorted order (in the real game, players do not see their own cards, hence the name for the game – Mind). Given the cards in each hand, you are to provide the order for the players to place their cards down.

The Input:

The first input line contains an integer, p ($2 \le p \le 6$), indicating the number of players. Each of the following p input lines provides the cards for one player. Each line starts with an integer, c ($1 \le c \le 9$), indicating the number of cards for that player; the count is followed by the cards in sorted order (each card between 10 and 99, inclusive).

The Output:

Print the order for the players to place down the cards so that the cards are in order.

Sample Input **Sample Output**

2 3 10 40 50 2 20 30	ABBAA
3 4 40 51 60 70 3 12 32 42 5 20 53 80 90 95	BCBABACAACCC

UCF Local Contest (Round 1A) — September 5, 2020

Unique Values

filename: `unique`
Difficulty Level: `Medium`
Time Limit: `2 seconds`

Arup has to make many practice questions for his Computer Science 1 students. Since many of the questions deal with arrays, he has to generate arrays for his students. Since he doesn't want to give them difficult practice problems, he always guarantees that the arrays (given to the students) have unique values. Namely, no value will appear twice in any of his arrays.

Unfortunately, Arup is too lazy to generate arrays! About 20 years ago when he started teaching Computer Science 1, he made one really long array to reuse but this long array may have duplicate values. When he makes problems, he simply grabs a contiguous subsequence of this long array to be the array to be used for a problem but he needs to make sure the contiguous subsequence does not contain duplicates. If the long array has terms a[0], a[1], …, a[n-1], a contiguous subsequence of the long array is any sequence of $j - i + 1$ terms a[i], a[i+1], …, a[j] where $0 \le i \le j \le n - 1$.

The Problem:

Given an array of *n* integers, determine how many contiguous subsequences of the array do not contain any repeated values. Note that two subsequences with the same contents are considered different (i.e., both counted in the total) if they start at different locations of the original long array.

The Input:

The first input line contains a single positive integer, *n* ($1 \le n \le 10^5$), representing the size of the input array. The following line contains *n* space separated integers, representing the values of the input array, in the order they appear in the array. Each of the array values will be an integer between 1 and 10^9, inclusive.

The Output:

On a line by itself, output the total number of contiguous subsequences of the input array that do not contain any repeated values.

Sample Input	**Sample Output**
5 1 1 2 1 5	9
8 2 12 3 12 3 2 6 9	22

UCF Local Contest (Round 1A) — September 5, 2020

Gone Fishing

filename: `fish`
Difficulty Level: `Medium`
Time Limit: `1 second`

It is getting dark and the mosquitoes are attacking Fisherman Floyd. Floyd decides to throw his circular net one last time and wants to make the most out of the last throw.

The Problem:

Given the size (radius) of Floyd's net and positions (x,y) of a set of fish, what is the maximum fish Floyd can catch with one throw? That is, find the maximum points you can have in the net (circle). If a point (fish) is within 10^{-6} of the circle boundary, consider the point in the circle.

The Input:

The first input line provides the radius of the circle. The second input line contains an integer, n ($1 \le n \le 100$), indicating the number of fish. Each of the next n input lines provides the location (x,y) for one fish. Assume all these points are distinct. Also assume that all the x and y values in the input (as well as the circle radius) are integers between 1 and 1000, inclusive.

The Output:

Print the maximum fish Floyd can catch with one throw.

Sample Input	Sample Output
20 4 6 7 5 5 190 100 4 4	3
3 2 1 1 95 4	1

UCF Local Contest (Round 1A) — September 5, 2020

Sum of a Function

filename: `sumfunc`
Difficulty Level: `Medium`
Time Limit: `1 second`

Everyone knows that Arup loves prime numbers! This is why he teaches the cryptography course at UCF. Recently, Arup defined the following function on positive integers, n, greater than 1:

f(n) = the smallest prime factor of n

For example, f(14) = 2, f(15) = 3, f(16) = 2 and f(17) = 17.

Using this function, we can generate a sequence of terms f(s), f(s+*1*), f(s+2), …, f(e), where s designates the starting function input and e designates the ending function input.

Arup thinks these sequences are interesting, but what he's really curious about is finding the sum of the k minimum elements in one of these sequences. Can you write a program to help him?

The Problem:

Given s, e, and k, find the sum of the k minimum values in the sequence f(s), f(s+*1*), f(s+2), …, f(e).

The Input:

The first and only input line will contain three positive integers, s ($2 \le s \le 10^{18}$), e (s+100 $\le e \le$ s+10^6), and k ($1 \le k \le$ *0.9 * (e – s + 1)*), representing (respectively) the starting function input, the ending function input, and the number of minimum terms to sum.

The Output:

On a line by itself, print the sum of the k minimum terms of the designated sequence.

Sample Input	**Sample Output**
`100 200 70`	`165`
`213 419 169`	`546`

Note: Even though the input specification does not allow "`14 17 3`" as an input case (i.e., this case will not be in the judge data), it is a simple case that you may want to use for testing purposes – the output (`7`) can be verified easily on paper. (BTW, the intended solution should solve this case properly anyway.)

UCF Local Contest (Round 1A) — September 5, 2020

Hang Gliding

filename: `hangglide`
Difficulty Level: `Medium`
Time Limit: `3 seconds`

The 2020 hang gliding world championships are coming to Florida next spring! (You may think it is odd to hold in Florida a sport that requires mountains, but it turns out that hang gliders can be towed to altitude by other aircraft.) The competition is divided into a set of tasks; completing a task successfully gives a pilot a number of points. Either all points are awarded for a task, or none are. For each task, every pilot has a probability of success. A pilot's score is the total of all successfully completed tasks at the end of the competition.

This year, the organizing committee couldn't decide on a reasonable number of tasks, so the time slots for tasks overlap. At any given time, there can be multiple tasks going on, but a pilot may only choose one to be competing in at that time. Each pilot may compete in as many tasks as they want given this constraint. The pilots know their own strengths and weaknesses, and will choose tasks in order to maximize their expected score.

You have been offered free hang gliding lessons if you help with scoring software for the event. Among other things, the committee wants the software to be able to predict the winners ahead of time.

The Problem:

Given a set of tasks, each with a time slot and a point score to be awarded for success, and a list of pilots, each with success probabilities for each task, compute the expected score for each pilot, and report the top 3 expected scores.

The Input:

The first input line contains two integers: t ($1 \le t \le 10000$), indicating the number of tasks, and p ($3 \le p \le 100$), indicating the number of pilots.

The next t input lines contain the task descriptions. Each line contains three integers (s, e, and a) separated by a space: $0 \le s < e \le 10000$, s is the start time of the task, and e is the end time of the task, in minutes after the competition starts; a ($1 \le a \le 100$) is the number of points awarded for the task. Note that the task start times and end times are non-inclusive, i.e., if a task ends at time T and another task starts at time T, a pilot can compete in both tasks.

After the task descriptions, there are t lines for each pilot. The first t lines in this section are the probabilities of success for each task for pilot 1; the next t lines are the probabilities of success for pilot 2, and so on. The probabilities are floating point numbers in the range 0 to 1, inclusive.

The Output:

Print the top 3 pilots. Print, in order of descending expected score, the pilot's number, a space, and the pilot's expected score, rounded to 2 decimal places (i.e., a score of 42.494 would be printed as 42.49; a score of 42.495 would be printed as 42.50). Note that pilots are numbered starting at 1, not zero.

Sample Input	**Sample Output**
3 3 0 1 5 1 2 10 2 3 15 0.0 0.0 0.2 1.0 0.0 0.0 0.0 0.75 0.0	3 7.50 2 5.00 1 3.00
3 4 0 3 50 3 6 50 0 6 75 0.2 0.3 0.6 0.6 0.6 0.5 0.6 0.5 0.4 0.9 0.9 0.9	4 90.00 2 60.00 3 55.00

UCF Local Contest (Round 1A) — September 5, 2020

Trading Cards

filename: `trading`
Difficulty Level: `Medium`
Time Limit: `1 second`

You've decided to get rid of your Trading Card Game (TCG) collection. The possible financial activities are as follows:

- You can sell individual cards to Bearimy's Card Emporium.
- You can buy individual cards from Bearimy's Card Emporium.
- Card price is the same regardless of the transaction (buy or sell) with the Emporium.
- You can sell a set of cards (called a collection) to your friend Jeremy. Jeremy enjoys showing off different collection sets, so Jeremy will buy multiple sets but only one of each set. You have to, of course, have the cards in a set to be able to sell such set to Jeremy.
- If a card is needed in different collection sets, you only need one copy of that card (and not multiple copies of that card) to form all those sets to sell to Jeremy; you need the other cards in the sets as well. For example, if cards {1,2} is a set, cards {2,3} is a set, and cards {1,3,4} is a set, to sell these three sets to Jeremy, you only need one of each card 1, 2, 3, and 4.
- Jeremy only buys collection sets and not individual cards.
- A collection set is not necessarily more/less expensive than the sum of its component cards.

The Problem:

Given a list of cards, including their card shop cost and whether you own them, and a list of collection sets and the value Jeremy will pay for those sets, determine the maximum amount of money you can earn. This is done by buying from (and/or selling to) the card shop and selling the sets to Jeremy. Note that you choose what to buy from (and/or sell to) the card shop and what to sell to Jeremy. Your earning will be:

(cards sold to the shop) + (sets sold to Jeremy) – (cards bought from the shop)

The Input:

The first input line contains an integer, n ($1 \le n \le 50$), representing the number of cards. Each of the next n input lines contains two integers: v_i ($1 \le v_i \le 10000$), representing the Bearimy card value, and h_i ($0 \le h_i \le 1$), representing whether you currently own the card ($h_i = 1$) or not ($h_i = 0$). The cards are provided in the order of indices (1-indexed).

Following the individual card specification will be the description for the collection sets. The first input line in this section contains an integer, m ($1 \le m \le 50$), representing the number of sets. The set descriptions follow, each set consisting of two consecutive input lines. The first line of each set description contains two integers: c_j ($1 \le c_j \le n$), representing the number of cards in the set, and w_j ($1 \le w_j \le 10000$), representing the value of the set. The second input line of each set

description contains c_j distinct positive integers between 1 and n; these values represent the indices (1-indexed) of the cards that belong to the set.

The Output:

Print a single integer, P, representing the maximum profit that can be earned by buying and selling cards with Bearimy and selling collection sets to Jeremy.

Sample Input	**Sample Output**
3 2 0 13 1 15 1 1 3 6 1 2 3	28
3 2 0 13 1 15 1 1 3 600 1 2 3	598
4 7 1 3 0 2 1 1 0 3 4 4 1 2 3 4 2 4 2 3 2 4 3 4	11

UCF Local Contest (Round 1A) — September 5, 2020

Median Inversion String

filename: `median`
Difficulty Level: `Medium-Hard`
Time Limit: `1 second`

This problem uses two concepts: *median* and *inversion*.

Median: Assume we have a sorted list of elements e_1, e_2, e_3, … We define median as the element in the middle of the list (i.e., halfway through the list). If there are an odd number of elements in the list, there is only one median element, e.g., if the list has 13 elements, then e_7 is the median element (there are as many elements in the list before e_7 as there are after e_7). If there are an even number of elements in the list, there are two median elements, e.g., if the list has 14 elements, then e_7 and e_8 are the median elements (there are as many elements in the list before e_7 as there are after e_8).

Inversion: Assume we have a string containing only the letters '*A*' and '*B*', e.g., *AAABABAA*. We define an inversion to be a pair of letters in the string where one letter is *B*, the other letter is *A*, and the position of *B* in the string is earlier than the position of *A* in the string. For our sample string, the *B* at position 4 and the *A* at position 5 form an inversion. Similarly, the *B* at position 4 and the *A* at position 7 form an inversion. There is a total of five inversions in the sample string.

The Problem:

Your friend, Michio, has a list of strings, each string containing only the letters '*A*' and '*B*'. Each string has *n* letters. Each string Michio has contains exactly *k* inversions. Michio claims that he has all strings of exactly *k* inversions. You want to test his theory.

Michio has sorted the strings in lexicographical order, and you will give the *median* string(s) to check if he has the correct strings.

Given the length (number of letters) in each string and the desired number of inversions, output the median string(s) in the list of strings with the given length.

The Input:

There is only one input line; it consists of two integers: n ($1 \le n \le 60$), representing the length of each string in the list Michio has, and k, representing the number of inversions per string. It is guaranteed that k will be selected such that at least one string containing *A*'s and *B*'s of length n will have exactly k inversions. (Note that the bounds for k are implied by the other constraints in the problem, i.e., the bounds for k need not be provided.)

The Output:

If the number of unique strings of *A*'s and *B*'s of length *n* with *k* inversions is odd, print a single line with the median string. If the number of unique strings of *A*'s and *B*'s of length *n* with *k* inversions is even, then print two lines each containing one of the median strings (these two strings should be printed in lexicographical order).

Sample Input	**Sample Output**
10 3	AAABABABBB
3 0	AAB ABB

UCF Local Contest (Round 1A) — September 5, 2020

Check List

filename: `checklist`
Difficulty Level: `Medium-Hard`
Time Limit: `5 seconds`

When test driving a vehicle, you are asked to make a checkmark on a tablet. Upon inspection, you have noticed many fingerprints on the screen. Aside from the rush of disgust, you notice that the fingerprints on the screen can be represented by a series of 2D points on a standard Cartesian grid. You know that checkmarks can be uniquely defined by three points; these three points have distinct x coordinates and the three points also have distinct y coordinates. The leftmost point must be lower than the rightmost point and higher than the middle point. Now you want to know how many unique checkmarks you can make using the 2D points.

Let's consider some examples. The three points (1,2), (2,1), and (3,2) do not form a valid checkmark because the leftmost and rightmost points are at the same height. The three points (1,2), (2,1), and (2,3) do not form a valid checkmark because two points have the same x coordinates. The three points (1,2), (3,1), and (4,9) do form a valid checkmark.

The Problem:

Given a list of 2D points, determine the number of unique checkmarks that can be formed from them.

The Input:

The first input line contains a single integer, n ($1 \leq n \leq 100{,}000$), representing the number of points. Each of the following n input lines contains two integers, x_i and y_i ($1 \leq x_i, y_i \leq 1{,}000{,}000{,}000$), representing the location of a point on the 2D grid. No two points will be identical.

The Output:

Print the number of distinct checkmarks that can be formed by the given 2D points.

Sample Input **Sample Output**

6 6 6 5 3 1 5 3 2 4 4 2 1	5
10 4 2 9 4 1 5 2 4 10 5 6 1 3 3 8 3 5 1 7 2	20