PROBLEMS

# 1.

Write a program that computes the sum of all even two-digit numbers. Print the result on standard output.

# 2.

Write a program that computes the sum of all odd two-digit numbers. Print the result on standard output.

# 3.

Write a program that computes y = x^n for given natural number n, n >= 1 and real number x.

# 4.

Write a program that for n numbers read from SI will count the numbers divisible by 3, have residue 1, and have residue 2.

# 5.

Write a program that will print all 4-digit numbers in which the sum of the three least significant digits is equal to the most significant digit.

4031 (4=0+3+1), 5131 (5=1+3+1)

# 6.

Write a program that will print all numbers in given range which are read the same from left to right and opposite.

# 7.

Write a program that for unknown count of integers read from SI will find the number with maximum value. The program stops when the reading of integer fails.

# 8.

Write a program that for unknown count of integers read from SI will find the number with maximum value. Numbers larger than 100 should be ignored. The program stops when the reading of integer fails.

# 9.

Write a program that for unknown count of integers read from SI will find the two with maximum values.

# 10.

Write a program that for N integers read from SI will find the difference of the sums of numbers on odd and even positions (by the order of reading). If this difference is less than 10 print the message "The two sums are close" else print "The two sums are far".

# 11.

Write a program that for unknown count of integers read from SI will find the positions of the successive numbers with maximum sum. The program stops when two successive read numbers are negative.

# 12.

Write a program that will compute the average of a student, and the number of failed exams. The number of taken exams by the student is not known.

# 13.

Given x = sin y with initial value 0.02 up to 0.80 with step 0.01. Write a program that will print table of y in degrees, minutes and seconds. y=arctan(\frac{x}{\sqrt{1 - x^2}})

# 14.

Write a program for computing approximation of \pi using the series of Gregory–Leibniz 1 − \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} − \ldots= \frac{\pi}{4} \sum\_{n=0}^{\infty}\frac{(-1)^n}{2n + 1} = \frac{\pi}{4} • using the first 100 members. • until the absolute value of the member that is adding is less then 10e-6

# 15.

Write a program for approximate computation of \pi using the expression 4\*\pi=arctan(1) and continuation of arctan into infinite series.

# 16.

Write a program in C that will find the number and sum of all integers between 100 and 200 that are divisible by 9.

# 17.

Write a program that will print the sum of the numbers divisible by 3 in the range [A,B) where A and B are numbers read from SI.

Explanation.

А=10, B = 20. Numbers that belong in the interval are 10,11,12,13,..,19. Divisible by 3 are the numbers 12,15,18, and their sum is 45.

# 18.

A natural number 'N' is read from the standard input. Then, N pairs of integers (air measurements of PM10 and PM2.5 particles from the corresponding measuring station for a given municipality) are read. Your task is to calculate the mean value of PM10 and PM2.5 particles only from those measuring stations that are in use. One measuring station is considered to be out of use if it sends a pair of measurements -1 -1.

Note: The result should be printed with 2 decimal places.

**Example 1:**

Input:

5 *(number of pairs of integers)*

35 56 *(the first pair)*

-1 -1 *(the second pair ...)*

0 0

102 189

200 225

Output:

PM10: 84.25 *(the mean value of PM10 particles)*

PM2.5: 117.50 *(the mean value of PM2.5 particles)*

**Example 2:**

Input:

1

-1 -1

Output:

PM10: Can not be calculated

PM2.5: Can not be calculated

# 19.

From standard input, a number x and n are entered, then n integers. You need to check if the number x is contained in the entered n numbers from the keyboard. x will always be less than the n numbers entered after it.

Numbers that meet the requirement need to be printed on the screen. If there are none, print None.

**Example**: First x and n are entered (let x = 23, n = 4). Then n (in this case 4) numbers are entered from the keyboard (let them be 1234, 2333, 1122, 1114455). The numbers to be printed on the screen are:

1**23**4, **23**33, because they contain the number x (in this case 23).

# 20.

From SI are read N pairs of integers, where the first number is the students' points, and the second is the max. points. Write a program where for each pair (points, max points), will print the students success in percentage and his grade following these rules:

if the student has 90% or more - "10".

if the student has 80% or more - "9".

if the student has 70% or more - "8".

if the student has 60% or more - "7".

if the student has 50% or more - "6".

If the student has less than 50% print "FAIL".

# 21.

A number X is read from SI. After that an undefined number of numbers are read(until something that i not a number is read).  
For each of those numbers, the program should check whether the count of digits of the numbers is the same as the count of digits of the number X. All numbers that satisfy this condition should be printed on SO.

# 22.

A sorted number is a number in which each consecutive digit in a number is smaller than the previous one (going left to right). For example, 7421 is such a number. One-digit numbers should be ignored.

First read one positive whole number N and than additional N numbers from SI. On SO, print all the numbers that are sorted numbers and at the end print the smallest sorted number that was entered.

If the are no such numbers, print only -1 on SO.

|  |  |
| --- | --- |
| Input | Output |
| 5  435  643  12234  721  7720 | 643  721  643 |

# 23.

Characters are read from the keyboard (from a text that consists of different characters and represents a sentence, i.e. it consists of several words made up of lowercase letters and always ends with a period) until a period is read. To make a program that will print out the same words with the difference that for each one appearing on two consonants next to each other, it will exchange their places. Note: assume that there are no words that have more than two consonants in a row.

# 24.

A rosary is a number obtained by gluing together the first N numbers, in order. For example, for N=4 it is 1234, for N=11 it is 1234567891011. Reverse rosary is similar to above but reversed. So for N=5 it is 54321. For total N, write them on the screen in order: Rosary for 1, reverse rosary for 1, rosary for 2, reverse rosary for 2..., so on until reverse rosary for N.

# 25.

A natural number N is read from SI, followed by N triples for integers (a,b,c). Calculate for each triplet the value

res = 2\*min - max/2 + |min-a|,

where min is the minimum number from the triplet and max is the maximum number of the triplet.

On SO print the value res for each of the triplets.

Input: 5  
5 1 6  
2 -10 6  
5 5 1  
1 5 5  
5 1 5\

Output: 3.00  
-11.00  
3.50  
-0.50  
3.50\

# 26.

Two integers X and N are read from SI, followed by N natural numbers (all of them larger than 9). For each read number, check whether the number X is a divisor of the number which is constructed by relocating the first digit of the number to the end of the number.

For each read number print YES on the screen if the condition is satisfied, and print NO otherwise.

Example: The number 1234 after the relocation of the first digit to the end of the number will be 2341. The number 10 will be 1, the number 200 will be 2.

Example. Input 2 2 43 100

Output YES NO

An explanation for the example: X=2, N=2 (the program should read two more natural numbers) 43 -> 34, 2 is divisor of 34 (output: YES) 100 -> 1, 2 is not a divisor of 1 (output: NO)

# 27.

Three natural numbers are read from the standard input: N, P (P <= 5) and C (C <= 9). Write a program that will print the first number greater than N in which the digit C appears exactly P times.

# 28.

Read from SI two integers a and b. Write a program that will print all numbers from a to b (a < b), such as the number created as a product of the digits from the next to the first to the next to last digit (middle digits = the digits that will remain if we remove the first and the last digit from the original number) is divisible with the number created from the last and the first digit. Do not print the numbers where the product of the middle digits is 0. At the end print the count of such numbers.

When printing out the numbers, next to each number that satisfies the condition, print the equation of that condition, for example: 55650 -> (150 == 50 \* 3), where 150 = 5 \* 6 \* 5 and 50 is constructed from 5 (the first digit) and the 0 (the last digit of the number)

# 29.

Using the characters + and - form a square image with "width" n (n > 2) that represent square of the chars -, enclosed by a square of chars + (view examples). For given n, print the appropriate image.

Attention: do not print additional characters as empty spaces, or unnecessary new line... The image ends with new line char. Do not use any matrix or arrays because n is not limited.

# 30.

From SI read one number that represents the today's date in format DDMMYYYY (DD-day, MM-month, YYYY-year). Then read one integer N, after you read N dates of birth in the given format. For each of the N read dates your program should print "YES" if on today's date (read in the beginning) the person with that date have more or equal to 18 years, otherwise should print "NO".

# 31.

One number is **interesting** if his **flipped** number is divisible by the number of digits it contains. The flipped number is composed from the same digits, but in opposite order (ex. 653 is flipped of the number 356). Read from SI one integer **n** ( **n > 9**). Find and print the largest integer small than **n** that is "interesting". If the read number is not valid print a message "The number is invalid".

# 32.

Read from SI one integer n. From the integers smaller than n, find the one with maximum sum of divisors. Do not count the number itself in the sum of divisors.

# 33.

Read from SI two integers a and b. Write a program that will print all numbers from a to b (a < b), such as the number created from all the digits from next to the first to the next to last digit in the original number (the digits that will remain if we remove the first and the last digit from the original number), reversed (read from right to left) is divisible with the number created as the sum of the first and the last digit. For single and two digit numbers, the number created from the middle digits is 0, and these numbers are not printed. At the end print out the count of such numbers.

Example. For the number 82675 the number created from the digits from the next to the first up to the next to last digit, reversed is 762, and the sum of the first and the last digit is (8 + 5) = 13. When printing out the numbers, next to each number that is satisfying the condition, print the equation of the condition in the following form:

original\_number -> (reversed\_middle == (last\_digit + first\_digit) \* quotient)

for example: 291 -> (9 == (1 + 2) \* 3) 84575 -> (754 == (5 + 8) \* 58)

# 34.

Read from SI unknown number of triplets of integers (three integers). The reading ends when the program can not read successfully read three integers. For each triplet of integers print out the two of them that have the minimum difference between them. First print the smaller one, and then the bigger integer. If two pairs of integers in the triplet have equal difference, than print out all the integers, ordered from the smallest to the largest.

Example: if 5 1 7 is entered -> 5 - 1 = 4, 7 - 5 = 2, 7 - 1 = 6 -> 2 is the smallest difference -> 5 7 should be printed

Input:

5 1 7

10 2 18

Output:

5 7

2 10 18

# 35.

Write a program in which from the standard input you read one positive integer z, and afterwards you read one after another pairs of integers (a, b). Thereby, the loop should end if you read the pair (0, 0). The program should calculate how many times the integer z is equal to the sum of the pairs a and b, as well as the percentage of the pairs (a,b) that give sum equal to z (NOTE: the pair (0,0) is not taken into account when doing calculations!).

# 36.

A whole number N greater then 0 is entered from the SI followed by N pairs of natural numbers greater than 10. Write a program that for every pair of entered numbers will check if the number obtained by moving the last digit of the first number on its first place is greater that the second number in the pair or not. Print YES if the condition is satisfied or NO if the condition is not satisfied on the standard output in a single line for each pair of numbers.

(Example for moving the last digit of a number on its first place: 1234 -> 4123, 98700 -> 9870). If the numbers 3456 and 6210 are entered as a pair YES should be printed since 6345 > 6210.

# 37.

Sweet number is number that contains only even digits (0, 2, 4, 6, 8). In given range (m and n read from SI) find the smallest "sweet number". If there is no such number, print NSN.

# 38.

Write a program for ‘’right folding a number’’. Right folding is defined as follows:

Each digit in the left half of the number is added to the corresponding opposite digit in the right half of the number (and removed from the number) – the leftmost digit is added to the rightmost digit, the second from the left with the second from the right etc. The addition of the digits is performed to module 10 so that if the addition of two digits yields a number greater than 9, only the least significant digit is taken. If the number has an odd number of digits, the middle digit si added to itself and its not removed(becoming the first digit of the new number). Write a program that for a given integer n>=10 will print number folded to the right. The task should be solved without the use of arrays.

# 39.

From SI read characters until you read the char !. In these characters are hidden integers (smaller than 100). Write a program that will read all characters and print the sum of all integers hidden in the text.

# 40.

Write a program that will determine out of n numbers (entered from the keyboard). The number of numbers that are divisible by 3, when dividing by 3 have a remainder of 1, i.e. 2.

# 41.

Write a program that calculates the sum of the digits of n prime numbers that are entered from the keyboard.

# 42.

write a program that will check if given a positive integer whose value read from the keyboard is a perfect number. We say that a given number is perfect if it is divisible by the sum of its divisors (including 1, but not itself number).

Example: The number 6 is perfect: 6 = 1 + 2 + 3

# 43.

write a program in which the number of rows n is entered from the keyboard.

The program should print the following image:

1

12

123

1234

12345

123456

1234567

if n = 7

# 44.

SOLUTIONS

# 1.

*/\*Write a program that computes the sum of all even two-digit numbers. Print the result on standard output.\*/*#include **<stdio.h>  
  
int** main() {  
 **int** sum=0;  
 **for**(**int** i=10 ; i<100 ; i+=2){  
 sum+=i;  
 **if**(i<98){  
 printf(**"%d + "**,i);  
 }**else**{  
 printf(**"%d = %d"**,i,sum);  
 }  
 }  
 **return** 0;  
}

# 2.

*/\*Write a program that computes the sum of all odd two-digit numbers. Print the result on standard output.\*/*#include **<stdio.h>  
  
int** main() {  
 **int** sum=0;  
 **for**(**int** i=11 ; i<100 ; i+=2){  
 sum+=i;  
 **if**(i<99){  
 printf(**"%d + "**,i);  
 }**else**{  
 printf(**"%d = %d"**,i,sum);  
 }  
 }  
 **return** 0;  
}

# 3.

*/\*Write a program that computes на y = x^n for given natural number n, n >= 1 and real number x.\*/*#include **<stdio.h>  
  
int** main() {  
 **int** x,n;  
 scanf(**"%d%d"**,&x,&n);  
 **int** y=1;  
 **for**(**int** i=0 ; i<n ; i++){  
 y\*=x;  
 }  
 printf(**"%d^%d = %d"**,x,n,y);  
 **return** 0;  
}

# 4.

*/\*Write a program that for n numbers read from SI will count the numbers divisible by 3, have residue 1, and have residue 2.\*/*#include **<stdio.h>  
  
int** main() {  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** d3=0, r1=0, r2=0;  
 **for**(**int** i=0 ; i<n ; i++){  
 **int** number;  
 scanf(**"%d"**,&number);  
 **if**(number%3==0){  
 d3++;  
 } **else**{  
 **if**(number%3==1){  
 r1++;  
 }**else**{  
 r2++;  
 }  
 }  
 }  
 printf(**"Divisible by 3 -> %d\nRemainder of 1 -> %d\nRemainder of 2 -> %d"**,d3,r1,r2);  
 **return** 0;  
}

# 5.

*/\*Write a program that will print all 4-digit numbers in which the sum of the three least significant digits is equal to the most significant digit.  
4031 (4=0+3+1), 5131 (5=1+3+1)  
\*/*#include **<stdio.h>  
  
int** main() {  
 **for**(**int** i=1000 ; i<10000 ; i++){  
 **int** ld=i%10;  
 **int** td=i/10%10;  
 **int** sd=i/100%10;  
 **int** fd=i/1000;  
 **if**(fd==sd+td+ld){  
 printf(**"%d (%d = %d + %d + %d)\n"**,i,fd,sd,td,ld);  
 }  
 }  
 **return** 0;  
}

# 6.

*/\*Write a program that will print all numbers in given range which are read the same from left to right and opposite.\*/*#include **<stdio.h>  
  
int** main() {  
 **int** a,b;  
 scanf(**"%d%d"**,&a,&b);  
  
 **for**(**int** i=a ; i<=b ; i++){  
 **int** tmp=i;  
 **int** reverseNumber=0;  
 **while**(tmp){  
 **int** lastDigit=tmp%10;  
 reverseNumber=reverseNumber\*10+lastDigit;  
 tmp/=10;  
 }  
 **if**(i==reverseNumber){  
 printf(**"%d\t"**,i);  
 }  
 }  
 **return** 0;  
}

# 7.

*/\*Write a program that for unknown count of integers read from SI will find the number with maximum value.  
 The program stops when the reading of integer fails.\*/*#include **<stdio.h>  
  
int** main() {  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** max=n;  
 **int** number;  
 **while** (scanf(**"%d"**,&number)){  
 **if**(number>max){  
 max=number;  
 }  
 }  
 printf(**"%d"**,max);  
 **return** 0;  
}

# 8.

*/\*Write a program that for unknown count of integers read from SI will find the number with maximum value.  
 Numbers larger than 100 should be ignored. The program stops when the reading of integer fails.\*/*#include **<stdio.h>  
  
int** main() {  
 **int** n,max;  
 **int** flag=1;  
  
 **while**(scanf(**"%d"**,&n)){  
 **if**(n>100){  
 **continue**;  
 }  
 **if**(flag){  
 max=n;  
 flag=0;  
 } **else if** (n>max){  
 max=n;  
 }  
 }  
 printf(**"%d"**,max);  
 **return** 0;  
}

# 9.

*/\*Write a program that for unknown count of integers read from SI will find the two with maximum values.\*/*#include **<stdio.h>  
  
int** main() {  
 **int** n1,n2;  
 **if**(scanf(**"%d%d"**,&n1,&n2)!=2){  
 **return** 0;  
 }  
 **int** max1,max2;  
 **if**(n1>n2){  
 max1=n1;  
 max2=n2;  
 }**else**{  
 max1=n2;  
 max2=n1;  
 }  
 **int** number;  
 **while**(scanf(**"%d"**,&number)){  
 **if**(number>max1){  
 max2=max1;  
 max1=number;  
 }**else if**(number>max2){  
 max2=number;  
 }  
 }  
 printf(**"%d, %d"**,max1,max2);  
 **return** 0;  
}

# 10.

*/\*Write a program that for N integers read from SI will find the difference of the sums of numbers on odd and even positions (by the order of reading).  
 If this difference is less than 10 print the message "The two sums are close" else print "The two sums are far".\*/*#include **<stdio.h>**#include **<math.h>  
int** main() {  
 **int** N;  
 scanf(**"%d"**,&N);  
 **int** diff, sumEvenPos=0, sumOddPos=0;  
 **for**(**int** i=1 ; i<=N ; i++){  
 **int** number;  
 scanf(**"%d"**,&number);  
 **if**(i%2==0){  
 sumEvenPos+=number;  
 } **else if**(i%2!=0){  
 sumOddPos+=number;  
 }  
 }  
 printf(**"Sum of numbers on odd positions = %d\nSum of numbers on even positions = %d\n"**,sumOddPos,sumEvenPos);  
 diff= abs(sumOddPos-sumEvenPos);  
 printf(**"Difference = %d\n"**,diff);  
 **if**(diff>10){  
 printf(**"The two sums are far"**);  
 } **else**{  
 printf(**"The two sums are close"**);  
 }  
 **return** 0;  
}

# 11.

*\*Write a program that for unknown count of integers read from SI will find the positions of the successive numbers with maximum sum.  
 The program stops when two successive read numbers are negative.\*/*#include **<stdio.h>  
  
int** main() {  
 **int** previous,current;  
 **int** sum,maxSum,maxPosition;  
  
 scanf(**"%d%d"**,&previous,&current);  
 **int** currentPosition=2;  
 sum=previous+current;  
 maxPosition=1;  
 maxSum=sum;  
 previous=current;  
 **while**(scanf(**"%d"**,&current)){  
 **if**(previous<0 && current<0){  
 **break**;  
 }  
 sum=previous+current;  
 **if**(sum>maxSum){  
 maxSum=sum;  
 maxPosition=currentPosition;  
 }  
 currentPosition++;  
 previous=current;  
 }  
 printf(**"Max sum: %d\nPositions: %d,%d"**, maxSum, maxPosition-1, maxPosition);  
 **return** 0;  
}

# 12.

*/\*Write a program that will compute the average of a student, and the number of failed exams. The number of taken exams by the student is not known.\*/*#include **<stdio.h>  
  
int** main() {  
 **int** grade, sum = 0, num\_grades = 0, failed = 0;  
 **while** (scanf(**"%d"**, &grade) == 1) {  
 **while** (grade < 5 || grade > 10) {  
 printf(**"Enter grade from 5 to 10: "**);  
 **if** (scanf(**"%d"**, &grade) != 1) **break**;  
 }  
 */\* counting \*/* **if** (grade >= 6 && grade <= 10) {  
 sum += grade;  
 num\_grades++;  
 }  
 **else** */\*if (grade == 5) \*/* failed++;  
 }  
 **if** (num\_grades == 0)  
 printf(**"No passed exams"**);  
 **else** printf(**"Average is %4.2f\n"**, (**float**)sum / num\_grades);  
 **if** (failed != 0)  
 printf(**"Failed on %d exams\n"**, failed);  
 **else** printf(**"No failed exams\n"**);  
 **return** 0;  
}

# 13.

*/\*Given x = sin y with initial value 0.02 up to 0.80 with step 0.01.  
Write a program that will print table of y in degrees, minutes and seconds. y=arctan(\frac{x}{\sqrt{1 - x^2}})\*/*#include **<stdio.h>***/\* #define \_USE\_MATH\_DEFINES \*/*#include **<math.h>  
int** main() {  
 **float** x, y;  
 **int** step, min, sec;  
 printf(**" x\tdegrees\tminutes\tseconds\n"**);  
 **for** (x = 0.02; x <= 0.8; x += 0.01) {  
 y = 180/**M\_PI** \* atan(x / sqrt(1 - x\*x));  
 step = y;  
 min = (y - step) \* 60;  
 sec = ((y - step) \* 60 - min) \* 60 + 0.5;  
 printf(**"%5.2f\t%3d\t %2d\t %2d\n"**, x, step, min, sec);  
 }  
 **return** 0;  
}

# 14.

*/\*Write a program for computing approximation of \pi using the series of Gregory–Leibniz   
 \* 1 − \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} − \ldots= \frac{\pi}{4} \sum\_{n=0}^{\infty}\frac{(-1)^n}{2n + 1} = \frac{\pi}{4}   
 \* • using the first 100 members.   
 \* • until the absolute value of the member that is adding is less then 10e-6\*/*#include **<stdio.h>**#define **\_USE\_MATH\_DEFINES**#include **<math.h>  
int** main() {  
 **double** member = 1, pi = 0;  
 **int** i, sign = 1, denominator = 1;  
 **for**(i=0; i<100; i++) {  
 pi += member;  
 denominator += 2;  
 sign = -sign;  
 member = (**double**)sign / denominator;  
 }  
 pi\*=4;  
 printf(**"pi (approximate) = %lf\n"**, pi);  
 **return** 0;  
}

# 15.

*/\*Write a program for approximate computation of \pi using the expression  
4\*\pi=arctan(1) and continuation of arctan into infinite series.\*/*#include **<stdio.h>**#include **<math.h>  
int** main()  
{  
 **int** faktori = 1, faktorb = 1;  
 **double** pi = 1, clen = 1;  
 */\* vo ciklusot se presmetuva pi/2 \*/* **do** {  
 faktori += 2;  
 clen \*= (**double**)faktorb / faktori;  
 faktorb++;  
 pi += clen;  
 } **while** (clen > 1e-6);  
*/\* do  
 pi += clen\*= (double)faktorb++ / (faktori+=2);  
 while (clen > 1e-6); \*/* pi \*= 2;  
 printf(**"pi (priblizno) = %10.8lf \t pi = %10.8lf\n"**, pi, **M\_PI**);  
 **return** 0;  
}

# 16.

*/\*Write a program in C that will find the number and sum of all integers between 100 and 200 that are divisible by 9.\*/*#include **<stdio.h>  
  
int** main()  
{  
 **for**(**int** i=100 ; i<=200 ; i++){  
 **if**(i%9==0){  
 printf(**"%d\t"**,i);  
 }  
 }  
 **return** 0;  
}

# 17.

*/\*Write a program that will print the sum of the numbers divisible by 3 in the range [A,B) where A and B are numbers read from SI.  
Explanation.  
А=10, B = 20. Numbers that belong in the interval are 10,11,12,13,..,19. Divisible by 3 are the numbers 12,15,18, and their sum is 45.  
\*/*#include **<stdio.h>  
  
int** main()  
{  
 **int** A,B;  
 scanf(**"%d%d"**,&A,&B);  
 **int** sum=0;  
 **for**(**int** i=A ; i<B ; i++){  
 **if**(i%3==0){  
 sum+=i;  
 }  
 }  
 printf(**"%d"**,sum);  
 **return** 0;  
}

# 18.

*/\*A natural number 'N' is read from the standard input. Then, N pairs of integers (air measurements of PM10 and PM2.5 particles from the  
corresponding measuring station for a given municipality) are read. Your task is to calculate the mean value of PM10 and PM2.5 particles  
only from those measuring stations that are in use. One measuring station is considered to be out of use if it sends a pair of measurements -1 -1.  
Note: The result should be printed with 2 decimal places.  
Example 1:  
Input:  
5 (number of pairs of integers)  
35 56 (the first pair)  
-1 -1 (the second pair ...)  
0 0  
102 189  
200 225  
Output:  
PM10: 84.25 (the mean value of PM10 particles)  
PM2.5: 117.50 (the mean value of PM2.5 particles)  
Example 2:  
Input:  
1  
-1 -1  
Output:  
PM10: Can not be calculated  
PM2.5: Can not be calculated  
  
\*/*#include **<stdio.h>  
  
int** main()  
{  
 **int** N;  
 scanf(**"%d"**,&N);  
  
 **int** sumPM10=0, sumPM2\_5=0, counter=0;  
 **for**(**int** i=0 ; i<N ; i++){  
 **int** PM10,PM2\_5;  
 scanf(**"%d%d"**,&PM10,&PM2\_5);  
 **if**(PM10==-1 && PM2\_5==-1){  
 **continue**;  
 } **else**{  
 sumPM10+=PM10;  
 sumPM2\_5+=PM2\_5;  
 counter++;  
 }  
 }  
 **if**(counter==0){  
 printf(**"PM10: Can not be calculated\nPM2.5: Can not be calculated"**);  
 } **else**{  
 printf(**"PM10: %.2f\nPM2.5: %.2f"**,(**float**)sumPM10/counter,(**float** )sumPM2\_5/counter);  
 }  
 **return** 0;  
}

# 19.

*/\*From standard input, a number x and n are entered, then n integers. You need to check if the number x is contained  
 in the entered n numbers from the keyboard. x will always be less than the n numbers entered after it.  
Numbers that meet the requirement need to be printed on the screen. If there are none, print None.  
Example: First x and n are entered (let x = 23, n = 4). Then n (in this case 4) numbers are entered from the keyboard  
(let them be 1234, 2333, 1122, 1114455). The numbers to be printed on the screen are:  
1234, 2333, because they contain the number x (in this case 23).\*/*#include **<stdio.h>**#include **<math.h>  
int** main(){  
 **int** x,n;  
 scanf(**"%d%d"**,&x,&n);  
 **int** tmp=x;  
 **int** counter=0;  
 **int** flag=0;  
 **while**(tmp){  
 tmp/=10;  
 counter++;  
 }  
 **for**(**int** i=0 ; i<n ; i++){  
 **int** number;  
 scanf(**"%d"**,&number);  
 **int** tmp=number;  
 **while** (tmp){  
 **if**(tmp%(**int**) pow(10,counter)==x){  
 printf(**"%d\n"**,number);  
 flag=1;  
 **break**;  
 }  
 tmp/=10;  
 }  
 }  
 **if**(!flag){  
 printf(**"None"**);  
 }  
 **return** 0;  
}

# 20.

*/\*From SI are read N pairs of integers, where the first number is the students' points, and the second is the max points.  
Write a program where for each pair (points, max points), will print the students success in percentage and his grade following these rules:  
if the student has 90% or more - "10".  
if the student has 80% or more - "9".  
if the student has 70% or more - "8".  
if the student has 60% or more - "7".  
if the student has 50% or more - "6".  
If the student has less than 50% print "FAIL".  
\*/*#include **<stdio.h>  
  
int** main(){  
 **int** N;  
 scanf(**"%d"**,&N);  
  
 **for**(**int** i=0 ; i<N ; i++){  
 **int** studetsPoints,maxPoints;  
 scanf(**"%d%d"**,&studetsPoints,&maxPoints);  
 **float** percentage=studetsPoints\*100.0/maxPoints;  
 **int** grade;  
 **if**(percentage<50){  
 printf(**"%.2f FAIL\n"**,percentage);  
 } **else**{  
 **if**(percentage>=50 && percentage<60){  
 grade=6;  
 printf(**"%.2f %d\n"**,percentage,grade);  
 } **else if**(percentage>=60 && percentage<70){  
 grade=7;  
 printf(**"%.2f %d\n"**,percentage,grade);  
 } **else if**(percentage>=70 && percentage<80){  
 grade=8;  
 printf(**"%.2f %d\n"**,percentage,grade);  
 } **else if**(percentage>=80 && percentage<90){  
 grade=9;  
 printf(**"%.2f %d\n"**,percentage,grade);  
 } **else if**(percentage>=90){  
 grade=10;  
 printf(**"%.2f %d\n"**,percentage,grade);  
 }  
 }  
 }  
 **return** 0;  
}

# 21.

*/\*A number X is read from SI. After that an undefined number of numbers are read(until something that i not a number is read).  
For each of those numbers, the program should check whether the count of digits of the numbers is  
the same as the count of digits of the number X. All numbers that satisfy this condition should be printed on SO.\*/*#include **<stdio.h>  
  
int** main(){  
 **int** X;  
 scanf(**"%d"**,&X);  
 **int** tmp=X;  
 **int** counterX=0;  
 **while**(tmp){  
 tmp/=10;  
 counterX++;  
 }  
  
 **int** number;  
 **while**(scanf(**"%d"**,&number)){  
 **int** tmp=number;  
 **int** counterN=0;  
 **while**(tmp){  
 tmp/=10;  
 counterN++;  
 }  
 **if**(counterX==counterN){  
 printf(**"%d\n"**,number);  
 }  
 }  
 **return** 0;  
}

# 22.

*/\*A sorted number is a number in which each consecutive digit in a number is smaller than the previous one (going left to right).  
 For example, 7421 is such a number. One-digit numbers should be ignored.  
First read one positive whole number N and than additional N numbers from SI. On SO, print all the numbers that are sorted numbers  
and at the end print the smallest sorted number that was entered.  
If the are no such numbers, print only -1 on SO.  
  
Input Output  
5  
435  
643  
12234  
721  
7720 643  
 721  
 643  
\*/*#include **<stdio.h>  
  
  
int** main(){  
 **int** N;  
 scanf(**"%d"**,&N);  
  
 **int** firstFlag=1;  
 **int** min;  
 **for**(**int** i=0 ; i<N ; i++){  
 **int** number;  
 scanf(**"%d"**,&number);  
 **int** temp=number;  
 **int** countD=0;  
 **while**(temp){  
 temp/=10;  
 countD++;  
 }  
  
 **int** tmp=number;  
 **int** countS=1;  
 **while**(tmp){  
 **int** ld=tmp%10;  
 **int** sld=tmp/10%10;  
 **if**(sld>ld){  
 countS++;  
 }**else**{  
 **break**;  
 }  
 tmp/=10;  
 }  
 **if**(countD==countS){  
 **if**(countD==1){  
 **continue**;  
 }  
 printf(**"%d\n"**,number);  
 **if**(firstFlag){  
 min=number;  
 firstFlag=0;  
 } **else if**(min>number){  
 min=number;  
 }  
 }  
 }  
 **if**(min==0){  
 printf(**"-1"**);  
 } **else**{  
 printf(**"%d"**,min);  
 }  
 **return** 0;  
}

# 23.

*/\*Characters are read from the keyboard (from a text that consists of different characters and represents a sentence,  
i.e. it consists of several words made up of lowercase letters and always ends with a period) until a period is read.  
To make a program that will print out the same words with the difference that for each one appearing on two consonants  
next to each other, it will exchange their places. Note: assume that there are no words that have more than two consonants in a row.\*/*#include **<stdio.h>  
  
int** main(){  
 **int** f = 0;  
 **char** prev=**'-'**, next, c;  
 **while**(scanf(**"%c"**, &next)){  
 **if**(prev==**'-'**){  
 prev = next;  
 }  
 **else if**(next == **'.'**){  
 **break**;  
 }  
 **else**{  
 **if**(prev >= **'a'** && prev <= **'z'** &&  
 prev != **'a'** &&  
 prev != **'e'** &&  
 prev != **'i'** &&  
 prev != **'o'** &&  
 prev != **'u'**&&  
 next >= **'a'** && next <= **'z'** &&  
 next != **'a'**&&  
 next != **'e'** &&  
 next != **'i'** &&  
 next != **'o'** &&  
 next != **'u'**) {  
 f = 1;  
 c = prev;  
 printf(**"%c"**, next);  
 }  
 **else if**(f) {  
 printf(**"%c"**, c);  
 f = 0;  
 }  
 **else** {  
 printf(**"%c"**, prev);  
 }  
 prev = next;  
 }  
 }  
 printf(**"%c%c"**,prev,next);  
 **return** 0;  
}

# 24.

*/\*A rosary is a number obtained by gluing together the first N numbers, in order.  
For example, for N=4 it is 1234, for N=11 it is 1234567891011. Reverse rosary is similar to above but reversed.  
So for N=5 it is 54321. For total N, write them on the screen in order: Rosary for 1, reverse rosary for 1,  
 rosary for 2, reverse rosary for 2..., so on until reverse rosary for N.\*/*#include **<stdio.h>  
  
int** main(){  
 **int** N;  
 scanf(**"%d"**,&N);  
  
 **int** number=0;  
 **for**(**int** i=1 ; i<=N ; i++){  
 number=number\*10+i;  
 printf(**"%d\n"**,number);  
  
 **int** tmp=number, reverse=0;  
 **while**(tmp){  
 **int** lastDigit=tmp%10;  
 reverse=reverse\*10+lastDigit;  
 tmp/=10;  
 }  
 printf(**"%d\n"**,reverse);  
 }  
 **return** 0;  
}

# 25.

*/\*A natural number N is read from SI, followed by N triples for integers (a,b,c). Calculate for each triplet the value  
res = 2\*min - max/2 + |min-a|,  
where min is the minimum number from the triplet and max is the maximum number of the triplet.  
On SO print the value res for each of the triplets.  
Input: 5  
5 1 6  
2 -10 6  
5 5 1  
1 5 5  
5 1 5\  
Output: 3.00  
-11.00  
3.50  
-0.50  
3.50\  
\*/*#include **<stdio.h>**#include **<math.h>  
  
int** main(){  
 **int** N;  
 scanf(**"%d"**,&N);  
 **int** min,max;  
 **for**(**int** i=0 ; i<N ; i++){  
 **int** a,b,c;  
 scanf(**"%d%d%d"**,&a,&b,&c);  
 min=max=a;  
 **if**(min>b){  
 min=b;  
 }  
 **if**(min>c){  
 c=min;  
 }  
 **if**(b>max){  
 max=b;  
 }  
 **if**(c>max){  
 max=c;  
 }  
  
 **float** res=2.0\*min-max/2.0 + abs(min-a);  
 printf(**"%.2f\n"**,res);  
 }  
 **return** 0;  
}

# 26.

*/\*Two integers X and N are read from SI, followed by N natural numbers (all of them larger than 9).  
For each read number, check whether the number X is a divisor of the number which is constructed by  
relocating the first digit of the number to the end of the number.  
For each read number print YES on the screen if the condition is satisfied, and print NO otherwise.  
Example: The number 1234 after the relocation of the first digit to the end of the number will be 2341.  
 The number 10 will be 1, the number 200 will be 2.  
Example. Input 2 2 43 100  
Output YES NO  
An explanation for the example: X=2, N=2 (the program should read two more natural numbers) 43 -> 34, 2 is divisor of 34 (output: YES)  
 100 -> 1, 2 is not a divisor of 1 (output: NO)\*/*#include **<stdio.h>**#include **<math.h>  
  
int** main(){  
 **int** X,N;  
 scanf(**"%d%d"**,&X,&N);  
  
 **for**(**int** i=0 ; i<N ; i++){  
 **int** number;  
 scanf(**"%d"**,&number);  
 **int** tmp=number;  
 **int** digits=0;  
 **while**(tmp){  
 tmp/=10;  
 digits++;  
 }  
 tmp=number/(**int**) pow(10,digits-1)+number%(**int**)pow(10,digits-1)\*10;  
 **if**(tmp%X==0){  
 printf(**"YES"**);  
 } **else**{  
 printf(**"NO"**);  
 }  
 }  
 **return** 0;  
}

# 27.

*/\*Three natural numbers are read from the standard input: N, P (P <= 5) and C (C <= 9).  
Write a program that will print the first number greater than N in which the digit C appears exactly P times.\*/*#include **<stdio.h>  
  
int** main(){  
 **int** N,P,C;  
 scanf(**"%d%d%d"**,&N,&P,&C);  
  
 **for**(**int** i=N+1 ;; i++){  
 **int** tmp=i;  
 **int** countOccurrencesOfC=0;  
 **while**(tmp){  
 **if**(tmp%10==C){  
 countOccurrencesOfC++;  
 }  
 tmp/=10;  
 }  
 **if**(countOccurrencesOfC==P){  
 printf(**"%d"**,i);  
 **break**;  
 }  
  
 }  
 **return** 0;  
}

# 28.

*/\*Read from SI two integers a and b. Write a program that will print all numbers from a to b (a < b),  
such as the number created as a product of the digits from the next to the first to the next  
to last digit (middle digits = the digits that will remain if we remove the first and the last digit from the original number)  
is divisible with the number created from the last and the first digit. Do not print the numbers where the product of  
the middle digits is 0. At the end print the count of such numbers.  
When printing out the numbers, next to each number that satisfies the condition, print the equation of that condition,  
 for example: 55650 -> (150 == 50 \* 3), where 150 = 5 \* 6 \* 5 and 50 is constructed from 5 (the first digit) and the 0 (the last digit of the number)\*/*#include **<stdio.h>**#include **<math.h>  
  
int** main(){  
 **int** a,b;  
 scanf(**"%d%d"**,&a,&b);  
 **int** counter=0;  
 **for**(**int** i=a+1 ; i<b ; i++){  
 **int** tmp=i;  
 **int** countOfDigits=0;  
 **while**(tmp){  
 tmp/=10;  
 countOfDigits++;  
 }  
 **int** fd=i/(**int**) pow(10,countOfDigits-1);  
 **int** ld=i%10;  
 **int** newNumber=fd\*10+ld;  
 **int** MiddleNumbers=i%(**int**) pow(10,countOfDigits-1)/10;  
 **int** product=1;  
 tmp=MiddleNumbers;  
 **while**(tmp){  
 ld=tmp%10;  
 product\*=ld;  
 tmp/=10;  
 }  
  
 **if**(product%newNumber==0){  
 **if**(product==0 || i/(**int**)pow(10,countOfDigits-2)%10==0){  
 **continue**;  
 } **else**{  
 printf(**"%d -> (%d == %d \* %d)\n"**,i,product,newNumber,product/newNumber);  
 counter++;  
 }  
 }  
 }  
 printf(**"%d"**,counter);  
 **return** 0;  
}

# 29.

# 30.

*/\*From SI read one number that represents the today's date in format DDMMYYYY (DD-day, MM-month, YYYY-year).  
Then read one integer N, after you read N dates of birth in the given format.  
For each of the N read dates your program should print "YES" if on today's date (read in the beginning)  
the person with that date have more or equal to 18 years, otherwise should print "NO".\*/*#include **<stdio.h>  
  
int** main(){  
 **long long int** date;  
 scanf(**"%lld"**,&date);  
 **long long int** day=date/1000000;  
 **long long int** month=date/10000%100;  
 **long long int** year=date%10000;  
 **int** N;  
 scanf(**"%d"**,&N);  
 **for**(**int** i=0 ; i<N ; i++){  
 **long long int** nDates;  
 scanf(**"%lld"**,&nDates);  
 **long long int** nDay=nDates/1000000;  
 **long long int** nMonth=nDates/10000%100;  
 **long long int** nYear=nDates%10000;  
 **if**(nYear+18==year){  
 **if**(nMonth==month){  
 **if**(nDay<=day){  
 printf(**"YES\n"**);  
 } **else if**(nDay>day){  
 printf(**"NO\n"**);  
 }  
 }  
 **if**(nMonth>month){  
 printf(**"NO\n"**);  
 }  
 **if**(nMonth<month){  
 printf(**"YES\n"**);  
 }  
 }  
 **else if**(nYear+18>year){  
 printf(**"NO\n"**);  
 }  
 **else if**(nYear+18<year){  
 printf(**"YES\n"**);  
 }  
 }  
 **return** 0;  
}

# 31.

*/\*One number is interesting if his flipped number is divisible by the number of digits it contains.  
The flipped number is composed from the same digits, but in opposite order (ex. 653 is flipped of the number 356).  
Read from SI one integer n ( n > 9). Find and print the largest integer small than n that is "interesting".  
If the read number is not valid print a message "The number is invalid".\*/*#include **<stdio.h>  
  
int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **if**(n<=9){  
 printf(**"The number is invalid"**);  
 **return** 0;  
 }  
 **for**(**int** i=n-1 ;; i--){  
 **int** tmp=i;  
 **int** reverse=0;  
 **while** (tmp){  
 **int** lastDigit=tmp%10;  
 reverse=reverse\*10+lastDigit;  
 tmp/=10;  
 }  
 tmp=i;  
 **int** countOfDigits=0;  
 **while** (tmp){  
 tmp/=10;  
 countOfDigits++;  
 }  
 **if**(reverse%countOfDigits==0){  
 printf(**"%d"**,i);  
 **break**;  
 }  
 }  
 **return** 0;  
}

# 32.

*/\*Read from SI one integer n. From the integers smaller than n,  
find the one with maximum sum of divisors. Do not count the number itself in the sum of divisors.\*/*#include **<stdio.h>  
  
int** main(){  
 **int** n, maxSumOfDivisors=0;  
 scanf(**"%d"**,&n);  
  
 **for**(**int** i=2 ; i<=n/2 ; i++){  
 **if**((n-1)%i==0){  
 maxSumOfDivisors++;  
 }  
 }  
 **int** maxNumber=n-1;  
  
 **for**(**int** i=n-2 ; i>n/2 ; i--){  
 **int** maxSumOfDivisors\_1=0;  
 **for**(**int** j=2 ; j<=i/2 ; j++){  
 **if**(i%j==0){  
 maxSumOfDivisors\_1++;  
 }  
 }  
 **if**(maxSumOfDivisors\_1>maxSumOfDivisors){  
 maxSumOfDivisors=maxSumOfDivisors\_1;  
 maxNumber=i;  
 }  
 }  
 printf(**"%d"**,maxNumber);  
 **return** 0;  
}

# 33.

*/\*Read from SI two integers a and b. Write a program that will print all numbers from a to b (a < b),  
such as the number created from all the digits from next to the first to the next to last digit in the original number  
(the digits that will remain if we remove the first and the last digit from the original number), reversed (read from right to left)  
is divisible with the number created as the sum of the first and the last digit. For single and two digit numbers,  
the number created from the middle digits is 0, and these numbers are not printed. At the end print out the count of such numbers.  
Example. For the number 82675 the number created from the digits from the next to the first up to the next to last digit,  
reversed is 762, and the sum of the first and the last digit is (8 + 5) = 13. When printing out the numbers,  
next to each number that is satisfying the condition, print the equation of the condition in the following form:  
original\_number -> (reversed\_middle == (last\_digit + first\_digit) \* quotient)  
for example: 291 -> (9 == (1 + 2) \* 3) 84575 -> (754 == (5 + 8) \* 58)  
\*/*#include **<stdio.h>**#include **<math.h>  
  
int** main(){  
 **int** a,b;  
 scanf(**"%d%d"**,&a,&b);  
 **int** count=0;  
 **for**(**int** i=a ; i<b ; i++){  
 **int** countOfDigits=1;  
 **int** tmp=i;  
 **while** (tmp){  
 tmp/=10;  
 countOfDigits\*=10;  
 }  
 **int** d=countOfDigits/10;  
 **int** firstDigit=i/d;  
 **int** lastDigit=i%10;  
 **int** middle=i%d/10;  
 **int** SumNewNumber=firstDigit+lastDigit;  
 tmp=middle;  
 **int** reverseMiddle=0;  
 **while**(tmp){  
 **int** ld=tmp%10;  
 reverseMiddle=reverseMiddle\*10+ld;  
 tmp/=10;  
 }  
 **int** quotient=reverseMiddle/SumNewNumber;  
  
 **if**(reverseMiddle%SumNewNumber==0){  
 **if**(i<=100 || middle==0){  
 **continue**;  
 } **else**{*//291 -> (9 == (1 + 2) \* 3) 84575 -> (754 == (5 + 8) \* 58)* printf(**"%d -> (%d == (%d + %d) \* %d)\n"**,i,reverseMiddle,lastDigit,firstDigit,quotient);  
 count++;  
 }  
 }  
  
 }  
 printf(**"%d"**,count);  
 **return** 0;  
}

# 34.

*/\*Read from SI unknown number of triplets of integers (three integers).   
The reading ends when the program can not read successfully read three integers.  
For each triplet of integers print out the two of them that have the minimum difference between them.  
First print the smaller one, and then the bigger integer. If two pairs of integers in the triplet have equal difference,  
than print out all the integers, ordered from the smallest to the largest.  
Example: if 5 1 7 is entered -> 5 - 1 = 4, 7 - 5 = 2, 7 - 1 = 6 -> 2 is the smallest difference -> 5 7 should be printed  
Input:  
5 1 7  
10 2 18  
Output:  
5 7  
2 10 18  
\*/*#include **<stdio.h>  
  
int** main(){  
 **int** a,b,c;  
 **int** tmp;  
  
 **while**(scanf(**"%d%d%d"**,&a,&b,&c)){  
 **if**(a>b && a>c && c>b){  
 tmp=b;  
 b=c;  
 c=tmp;  
 }  
 **if**(b>=a && b>=c){  
 **if**(a>=c){  
 tmp=a;  
 a=b;  
 b=tmp;  
 }**else if**(c>=a){  
 tmp=a;  
 a=b;  
 b=c;  
 c=tmp;  
 }  
 }**else if**(c>=a && c>=b){  
 **if**(a>=b){  
 tmp=b;  
 b=a;  
 a=c;  
 c=tmp;  
 }**else if**(b>=a){  
 tmp=a;  
 a=c;  
 c=tmp;  
 }  
 }  
 **if**(a==b || b==c){  
 printf(**"%d %d\n"**,b,b);  
 }**else**{  
 **if**(a-b==a-c || a-b==b-c || b-c==a-c){  
 printf(**"%d %d %d\n"**,c,b,a);  
 }**else**{  
 **if**(a-b<b-c && a-b<a-c){  
 printf(**"%d %d\n"**,b,a);  
 }**else if**(b-c<a-b && b-c<a-c){  
 printf(**"%d %d\n"**,c,b);  
 }**else**{  
 printf(**"%d %d\n"**,c,a);  
 }  
 }  
 }  
 }  
  
 **return** 0;  
}

# 35.

*/\*Write a program in which from the standard input you read one positive integer z, and afterwards you read one after  
another pairs of integers (a, b). Thereby, the loop should end if you read the pair (0, 0). The program should calculate  
how many times the integer z is equal to the sum of the pairs a and b, as well as the percentage of the pairs (a,b) that give sum equal to z  
(NOTE: the pair (0,0) is not taken into account when doing calculations!).\*/*#include **<stdio.h>  
  
int** main(){  
 **int** z;  
 scanf(**"%d"**,&z);  
 **int** a,b,countOfPairsWithSumZ=0,countOfPairs=0;  
 **while**(scanf(**"%d%d"**,&a,&b)){  
 **if**(a==0 && b==0){  
 **break**;  
 }  
 **if**(z==a+b){  
 countOfPairsWithSumZ++;  
 }  
 countOfPairs++;  
 }  
 **float** percentage=((**float**)countOfPairsWithSumZ\*100)/(**float**)countOfPairs;  
 printf(**"You entered %d pair of numbers whose sum is %d\nThe percentage of pairs with sum %d is %.2f%%"**,countOfPairsWithSumZ,z,z,percentage);  
 **return** 0;  
}

# 36.

*/\*A whole number N greater then 0 is entered from the SI followed by N pairs of natural numbers greater than 10.  
Write a program that for every pair of entered numbers will check if the number obtained by moving the last digit of the first number  
on its first place is greater that the second number in the pair or not. Print YES if the condition is satisfied  
or NO if the condition is not satisfied on the standard output in a single line for each pair of numbers.  
(Example for moving the last digit of a number on its first place: 1234 -> 4123, 98700 -> 9870).  
If the numbers 3456 and 6210 are entered as a pair YES should be printed since 6345 > 6210.\*/*#include **<stdio.h>  
  
int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **if**(n<=0){  
 **return** 0;  
 }  
 **for**(**int** i=0 ; i<n ; i++){  
 **int** a,b;  
 scanf(**"%d%d"**,&a,&b);  
 **if**(a <= 10){  
 **break**;  
 }  
 **int** countOfDigits=1;  
 **int** tmp=a;  
 **while** (tmp){  
 tmp/=10;  
 countOfDigits\*=10;  
 }  
 **int** d=countOfDigits/10;  
 **int** lastDigit= a % 10;  
 **int** restOfTheNumber= a / 10;  
 **int** newNumber=lastDigit\*d+restOfTheNumber;  
 **if**(newNumber>b){  
 printf(**"YES\n"**);  
 } **else**{  
 printf(**"NO\n"**);  
 }  
 }  
 **return** 0;  
}

# 37.

*/\*Sweet number is number that contains only even digits (0, 2, 4, 6, 8).  
 In given range (m and n read from SI) find the smallest "sweet number". If there is no such number, print NSN.\*/*#include **<stdio.h>  
  
int** main(){  
 **int** m,n;  
 scanf(**"%d%d"**,&m,&n);  
 **int** countEvenDigits;  
 **for**(**int** i=m ; i<=n ; i++){  
 **int** tmp=i;  
 **int** countOfDigits=0;  
 **while**(tmp){  
 tmp/=10;  
 countOfDigits++;  
 }  
 tmp=i;  
 countEvenDigits=0;  
 **while** (tmp){  
 **int** ld=tmp%10;  
 **if**(ld%2==0){  
 countEvenDigits++;  
 }  
 tmp/=10;  
 }  
 **if**(countEvenDigits==countOfDigits){  
 printf(**"%d"**,i);  
 **break**;  
 }  
 }  
 **if**(countEvenDigits==0){  
 printf(**"NSN"**);  
 }  
 **return** 0;  
}

# 38.

*/\*Write a program for ‘’right folding a number’’. Right folding is defined as follows:  
Each digit in the left half of the number is added to the corresponding opposite digit in the right half of the number  
(and removed from the number) – the leftmost digit is added to the rightmost digit, the second from the left with the second from the right etc.  
The addition of the digits is performed to module 10 so that if the addition of two digits yields a number greater than 9,  
only the least significant digit is taken. If the number has an odd number of digits, the middle digit si added to itself and  
its not removed(becoming the first digit of the new number). Write a program that for a given integer n>=10  
will print number folded to the right. The task should be solved without the use of arrays.\*/*#include **<stdio.h>**#include **<math.h>  
  
int** main(){  
 **int** number;  
 scanf(**"%d"**,&number);  
 **if**(number<10){  
 **return** 0;  
 }  
 **int** tmp=number;  
 **int** FoldedNumber=0;  
 **while** (tmp){  
 **int** temporary=tmp;  
 **int** countOfTmp=0;  
 **while**(temporary){  
 temporary/=10;  
 countOfTmp++;  
 }  
 **int** firstDigt=tmp/(**int**)pow(10,countOfTmp-1);  
 **int** lastDigit=tmp%10;  
 FoldedNumber=FoldedNumber\*10+(firstDigt+lastDigit)%10;  
 tmp=tmp%(**int**)pow(10,countOfTmp-1)/10;  
 }  
 tmp=FoldedNumber;  
 **int** reverseFoldedNumber=0;  
 **while** (tmp){  
 **int** ld=tmp%10;  
 reverseFoldedNumber=reverseFoldedNumber\*10+ld;  
 tmp/=10;  
 }  
 printf(**"%d"**,reverseFoldedNumber);  
 **return** 0;  
}

# 39.

2nd midterm & exams file

# 40.

*/\*Write a program that will determine out of n numbers (entered from the keyboard).  
The number of numbers that are divisible by 3, when dividing by 3 have a remainder of 1, i.e. 2.\*/*#include **<stdio.h>  
  
int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** div3=0,r1=0,r2=0;  
 **for**(**int** i=0 ; i<n ; i++){  
 **int** number;  
 scanf(**"%d"**,&number);  
 **if**(number%3==0){  
 div3++;  
 } **else**{  
 **if**(number%3==2){  
 r2++;  
 } **else**{  
 r1++;  
 }  
 }  
 }  
 printf(**"The number of numbers that are divisible by 3 is %d\nThe number of numbers when divided by 3 that have residue 1 is %d\nThe number of numbers when divided by 3 that have residue 2 is %d"**,div3,r1,r2);  
 **return** 0;  
}

# 41.

Find this and others below at Functions file better solutions