LAB 1

# 1.

Write a program that will read two integers from SI and on SO will print their sum and product.

Solution

#include **<stdio.h>  
  
int** main(){  
 **int** a,b;  
 scanf(**"%d%d"**,&a,&b);  
 printf(**"%d\n%d"**,a+b,a\*b);  
 **return** 0;  
}

# 2.

Write a program that computes average grade of semester. The program reads 5 integers and should print out the average as floating point number with two decimal places.

Solution

#include **<stdio.h>  
  
int** main(){  
 **int** g1,g2,g3,g4,g5;  
 scanf(**"%d%d%d%d%d"**,&g1,&g2,&g3,&g4,&g5);  
 printf(**"%.2f"**,(**float** )(g1+g2+g3+g4+g5)/5);  
 **return** 0;  
}

# 3.

Write a program that for a given amount of money, will print the minimum bills and coins needed to make the payment. The amount is integer read from standard input. The result should be printed in 9 lines, the number of bills or coins for each of them.

Example: 1583 denars, will be best payed out as:

0 x 5000

1 x 1000

1 x 500

0 x 100

0 x 50

3 x 10

0 x 5

1 x 2

1 x 1

Solution

#include **<stdio.h>  
  
int** main(){  
 **int** money;  
 scanf(**"%d"**,&money);  
   
 printf(**"%d x 5000\n"**, money / 5000);  
 money%=5000;  
 printf(**"%d x 1000\n"**, money / 1000);  
 money%=1000;  
 printf(**"%d x 500\n"**, money / 500);  
 money%=500;  
 printf(**"%d x 100\n"**, money / 100);  
 money%=100;  
 printf(**"%d x 50\n"**, money / 50);  
 money%=50;  
 printf(**"%d x 10\n"**, money / 10);  
 money%=10;  
 printf(**"%d x 5\n"**, money / 5);  
 money%=5;  
 printf(**"%d x 2\n"**, money / 2);  
 money%=2;  
 printf(**"%d x 1\n"**, money);  
 **return** 0;  
}

LAB 2

# 1.

Read a number from SI. Then print 1 if the number , if not pritnt 0. The solution should be done by using logical operators(without if else).

Solution

#include **<stdio.h>  
  
int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 printf(**"%d"**,(n>-100 && n<100) || (n>=200 && n<300));  
 **return** 0;  
}

# 2.

Read a six digit number from the SI. Then print 1 if the number is a symmetric number, or 0 if it isn’t. A symmetric number is a number where the first digit is equal with the last digit, the second digit is equal with the fifth digit and dhe third digit is equal with the fourth digit.

Solution

#include **<stdio.h>  
  
int** main(){  
 **int** x;  
 scanf(**"%d"**,&x);  
 printf(**"%d"**,(x/100000==x%10) && (x/10000%10==x/10%10) && (x/1000%10==x/100%10));  
  
 **return** 0;  
}

# 3.

Three numbers are read from SI code, the price and the users balance. Calculate the total price including th VAT which is equal to the last two digits of the code and print 1 if the user has enough money to pay or 0 if he has not.

Solution

#include **<stdio.h>  
  
int** main(){  
 **float** price, usersBalance, totalPrice;  
 **int** code, VAT;  
 scanf(**"%d%f%f"**,&code,&price,&usersBalance);  
 VAT=code%100;  
 totalPrice=price\*(1+VAT/100.0);  
 printf(**"The Total price is: %f\n"**,totalPrice);  
 printf(**"%d"**,totalPrice<=usersBalance);  
  
 **return** 0;  
}

LAB 3

# 1.

A number N is read from SI. The program should print on SO "Tik" if the number is divisible by 3, "Tak" if the number is divisible by 5 or "Tik-Tak" if the number is divisible by both 3 and 5. If the number is not divisible by 3 nor 5, than a message "Bad number" should be printed.

Solution

#include **<stdio.h>  
  
int** main(){  
 **int** N;  
 scanf(**"%d"**,&N);  
 **if**(N%3==0 && N%5==0){  
 printf(**"Tik-Tak"**);  
 } **else**{  
 **if**(N%3==0){  
 printf(**"Tik"**);  
 } **else**{  
 **if**(N%5==0){  
 printf(**"Tak"**);  
 } **else**{  
 printf(**"Bad number"**);  
 }  
 }  
 }  
 **return** 0;  
}

# 2.

Write a program that for a date read from SI (in the format DD MM YYYY) will print on standard output a message YES if the date is correct and possible, or NO if the date is not correct.

When deciding whether the date is correct or not correct, you have to consider the following factors:

* is the month between January and December (1-12)
* does the number for days correspond with the number of days in the specified month
* if the month is February, is the year leap?
* Leap years are those years who are divisible with 400, or they are divisible with 4, but not with 100

Solution

#include **<stdio.h>  
  
int** main(){  
 **int** day,month,year;  
 scanf(**"%d %d %d"**,&day, &month, &year);  
 **switch** (month){  
 **case** 1: {*//jan* **if**(day>=1 && day<=31){  
 printf(**"YES"**);  
 } **else**{  
 printf(**"NO"**);  
 }  
 **break**;  
 }  
 **case** 2: {*//feb* **if**(day==29){  
 **if**(year%4==0 && year%100!=0 || year%4==0){  
 printf(**"YES"**);  
 }**else**{  
 printf(**"NO"**);  
 }  
 } **else if**(day>=1 && day<=28){  
 printf(**"YES"**);  
 } **else**{  
 printf(**"NO"**);  
 }  
 **break**;  
 }  
 **case** 3:{*//mar* **if**(day>=1 && day<=31){  
 printf(**"YES"**);  
 } **else**{  
 printf(**"NO"**);  
 }  
 **break**;  
 }  
 **case** 4:{*//apr* **if**(day>=1 && day<=30){  
 printf(**"YES"**);  
 } **else**{  
 printf(**"NO"**);  
 }  
 **break**;  
 }  
 **case** 5:{*//may* **if**(day>=1 && day<=31){  
 printf(**"YES"**);  
 } **else**{  
 printf(**"NO"**);  
 }  
 **break**;  
 }  
 **case** 6:{*//june* **if**(day>=1 && day<=30){  
 printf(**"YES"**);  
 } **else**{  
 printf(**"NO"**);  
 }  
 **break**;  
 }  
 **case** 7:{*//july* **if**(day>=1 && day<=31){  
 printf(**"YES"**);  
 } **else**{  
 printf(**"NO"**);  
 }  
 **break**;  
 }  
 **case** 8:{*//aug* **if**(day>=1 && day<=31){  
 printf(**"YES"**);  
 } **else**{  
 printf(**"NO"**);  
 }  
 **break**;  
 }  
 **case** 9:{*//sep* **if**(day>=1 && day<=30){  
 printf(**"YES"**);  
 } **else**{  
 printf(**"NO"**);  
 }  
 **break**;  
 }  
 **case** 10:{*//oct* **if**(day>=1 && day<=31){  
 printf(**"YES"**);  
 } **else**{  
 printf(**"NO"**);  
 }  
 **break**;  
 }  
 **case** 11:{*//nov* **if**(day>=1 && day<=30){  
 printf(**"YES"**);  
 } **else**{  
 printf(**"NO"**);  
 }  
 **break**;  
 }  
 **case** 12:{*//dec* **if**(day>=1 && day<=31){  
 printf(**"YES"**);  
 } **else**{  
 printf(**"NO"**);  
 }  
 **break**;  
 }  
 **default**:  
 printf(**"NO"**);  
 }  
 **return** 0;  
}

# 3.

Write a program that for a three digit number read from SI will check if it is palindrome or not, and will print out an appropriate message. The message is "Palindrome" if it is, and "Not palindrome" if it's not. If the number is not three digit number, write the message "Wrong input".

Solution

#include **<stdio.h>  
  
int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **if**(n/100>9){  
 printf(**"Wrong input"**);  
 **return** 0;  
 }  
 **if**(n/100==n%10){  
 printf(**"Palindrome"**);  
 } **else**{  
 printf(**"Not Palindrome"**);  
 }  
 **return** 0;  
}

LAB 4

# 1.

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.

Solution

#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;  
}

# 2.

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

Solution

#include **<stdio.h>  
  
int** main(){  
 **int** N;  
 scanf(**"%d"**,&N);  
 **int** PM10,PM2\_5;  
 **int** sum10=0,sum2\_5=0,counter=0;  
 **for**(**int** i=0 ; i<N ; i++){  
 scanf(**"%d%d"**,&PM10,&PM2\_5);  
 **if**(PM10==-1 && PM2\_5==-1){  
 **continue**;  
 } **else**{  
 sum10+=PM10;  
 sum2\_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**)sum10/counter,(**float**)sum2\_5/counter);  
 }  
 **return** 0;  
}

# 3.

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).

Solution

#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;  
}

# 4.

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.

Solution

#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;  
}

LAB 5

# 1.

A rectangle located in the first quadrant is given. The rectangle is defined by the coordinates (x,y) of the two opposite corners of the rectangle: A (bottom left) and C (top right). The coordinates are integers and are read from SI. After reading the coordinates of the points A and C, the program reads coordinates of two other points (P1 and P2) also from SI. Write a program that, for each of the two points P1 and P2, will print YES if the point lays over the edge of the rectangle (on the sides of the rectangle) or NO otherwise (lays in or outside the rectangle).

Solution

#include **<stdio.h>  
  
int** main (){  
 **int** x\_A,x\_C,x\_P1,x\_P2,y\_A,y\_C,y\_P1,y\_P2;  
 scanf(**"%d%d"**,&x\_A,&y\_A);  
 scanf(**"%d%d"**,&x\_C,&y\_C);  
 scanf(**"%d%d"**,&x\_P1,&y\_P1);  
 scanf(**"%d%d"**,&x\_P2,&y\_P2);  
 **if**(x\_P1==x\_A && (y\_P1>=y\_A && y\_P1<=y\_C)){  
 printf(**"YES\n"**);  
 } **else if**(x\_P1==x\_C && (y\_P1>=y\_A && y\_P1<=y\_C)){  
 printf(**"YES\n"**);  
 } **else if**(y\_P1==y\_A && (x\_P1>=x\_A && x\_P1<=x\_C)){  
 printf(**"YES\n"**);  
 } **else if**(y\_P1==y\_C && (x\_P1>=x\_A && x\_P1<=x\_C)){  
 printf(**"YES\n"**);  
 } **else**{  
 printf(**"NO\n"**);  
 }  
 **if**(x\_P2==x\_A && (y\_P2>=y\_A && y\_P2<=y\_C)){  
 printf(**"YES\n"**);  
 } **else if**(x\_P2==x\_C && (y\_P2>=y\_A && y\_P2<=y\_C)){  
 printf(**"YES\n"**);  
 } **else if**(y\_P2==y\_A && (x\_P2>=x\_A && x\_P2<=x\_C)){  
 printf(**"YES\n"**);  
 } **else if**(y\_P2==y\_C && (x\_P2>=x\_A && x\_P2<=x\_C)){  
 printf(**"YES\n"**);  
 } **else**{  
 printf(**"NO\n"**);  
 }  
 **return** 0;  
}

# 2.

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.

Solution

#include **<stdio.h>  
  
int** countOfDigits (**int** number){  
 **int** count=0;  
 **while**(number){  
 number/=10;  
 count++;  
 }  
 **return** count;  
}  
  
**int** sortedNumber(**int** number){  
 **int** count=1;  
 **while**(number){  
 **int** last=number%10;  
 number/=10;  
 **int** slast=number%10;  
 **if**(slast>last){  
 count++;  
 }  
 }  
 **return** count;  
}  
  
**int** main(){  
 **int** N;  
 scanf(**"%d"**,&N);  
 **int** min=0,firstFlag=1;  
 **for**(**int** i=0 ; i<N ; i++){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** numOfDig= countOfDigits(n);  
 **int** sortedN= sortedNumber(n);  
 **if**(numOfDig==sortedN){  
 **if**(numOfDig==1){  
 **continue**;  
 }  
 printf(**"%d\n"**,n);  
 **int** tmp=n;  
 **if**(firstFlag){  
 min=tmp;  
 firstFlag=0;  
 } **else if**(min>tmp){  
 min=tmp;  
 }  
 }  
 }  
 **if**(min){  
 printf(**"%d"**,min);  
 }**else**{  
 printf(**"-1"**);  
 }  
 **return** 0;  
}

LAB 6

# 1.

A number N and N elements of an array are read from SI.

The array should be transformed in a new array where all the even elements should be increased by 1, and all the odd elements decreased by 1.

The transformed array should be printed to standard output and all the elements of the array should be separated with space between them.

Solution

#include **<stdio.h>  
  
int** main(){  
 **int** array[100];  
 **int** N;  
 scanf(**"%d"**,&N);  
 **for**(**int** i=0; i<N ; i++){  
 scanf(**"%d"**,&array[i]);  
 }  
 **for**(**int** i=0 ; i<N ; i++){  
 **if**(array[i]%2==0){  
 array[i]+=1;  
 }  
 **else**{  
 array[i]-=1;  
 }  
 }  
 **for**(**int** i=0; i<N ; i++){  
 printf(**"%d "**,array[i]);  
 }  
 **return** 0;  
}

# 2.

An array of integers A with length N (N<=100) is read from SI. Also, a integer K is read from SI.

Your task is to transform the array into a new array where the elements less than K will be first, and after them the elements greater or equal than K should follow. The order of the elements should not be changed.

Print the elements of the transformed array on standard output. All elements should be separated with a space between them.

Example:

A[] = {1, 3, 2, 5, 9, 0, 8, 10}, K=6

The transformed array will be:

A[] = {1, 3, 2, 5, 0, 9, 8, 10}

A[] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0}, K=6

The transformed array will be:

A[] = {5, 4, 3, 2, 1, 0, 10, 9, 8, 7, 6}

A[]={2, 4, 6, 8, 10}, K = 5

The transformed array will be:

A[] = {2, 4, 6, 8, 10}

A[]={1, 3, 5, 7, 9} K = 5

The transformed array will be:

A[] = {1, 3, 5, 7, 9}

Solution

#include **<stdio.h>  
  
int** main(){  
 **int** A[100];  
 **int** N;  
 scanf(**"%d"**,&N);  
  
 **for**(**int** i=0 ; i<N ; i++){  
 scanf(**"%d"**,&A[i]);  
 }  
 **int** K;  
 scanf(**"%d"**,&K);  
  
 **int** smallerK[100],countS=0,greaterK[100],countG=0;  
  
 **for**(**int** i=0 ; i<N ; i++){  
 **if**(A[i]<K){  
 smallerK[countS]=A[i];  
 countS++;  
 }**else**{  
 greaterK[countG]=A[i];  
 countG++;  
 }  
 }  
 **for**(**int** i=0 ; i<countS ; i++){  
 printf(**"%d "**,smallerK[i]);  
 }  
  
 **for**(**int** i=0 ; i<countG ; i++){  
 printf(**"%d "**,greaterK[i]);  
 }  
 **return** 0;  
}

# 3.

An integer N is read from SI. Afterwards N arrays of integers (with maximum length of 100 elements) are read from SI. For each array, firstly the size of the array is read, and then the elements of the array.

Your task is to write a program which for each array read from SI will print the percentage of **equal symmetric numbers** in the array. Symmetric elements in an array are the first and the last, the second and element before the last, etc.

If the array has odd number of elements, then the middle element is considered both symmetric and equal.

Input:

3

5 7 2 5 2 8

6 1 2 3 3 1 1

8 8 4 8 3 2 1 4 8

Output and explanation:

60.00% (2,2,5 are symmetrical and equal -> 3/5 elements -> 60%)

66.67% (1,1,3,3 are symmetrical and equal -> 4/6 elements -> 66.67%)

50.00% (8,8,4,4 are symmetrical and equal -> 4/8 elements -> 50.00%)

Solution

#include **<stdio.h>  
  
int** main(){  
 **int** N;  
 scanf(**"%d"**,&N);  
  
 **for**(**int** i=0 ; i<N ; i++){  
  
 **int** size;  
 scanf(**"%d"**,&size);  
  
  
 **int** A[100];  
 **for**(**int** j=0 ; j<size ; j++){  
 scanf(**"%d"**,&A[j]);  
 }  
  
 **int** counter=0;  
 **for**(**int** j=0,k=size-1 ; j<size ; j++,k--){  
 **if**(A[j]==A[k]){  
 counter++;  
 }  
 }  
 printf(**"%.2f%%\n"**,counter\*100.0/size);  
  
  
 }  
 **return** 0;  
}

LAB 7

# 1.

A squared matrix is read from SI. First, the number of rows and columns N is read, and then the N\*N elements of the matrix.

Change the sign of the elements located on the main diagonal (the positive numbers should become negative and reverse).

Print the transformed matrix on the screen (each element is printed with 3 places using %3d).

Solution

#include **<stdio.h>  
  
int** main() {  
 **int** N;  
 scanf(**"%d"**,&N);  
 **int** M[100][100];  
  
 **for**(**int** i=0 ; i<N ; i++){  
 **for**(**int** j=0 ; j<N ; j++){  
 scanf(**"%d"**,&M[i][j]);  
 }  
 }  
  
 **for**(**int** i=0 ; i<N ; i++){  
 M[i][i]\*=-1;  
 }  
  
 **for**(**int** i=0 ; i<N ; i++){  
 **for**(**int** j=0 ; j<N ; j++){  
 printf(**"%3d "**,M[i][j]);  
 }  
 printf(**"\n"**);  
 }  
 **return** 0;  
}

# 2.

A matrix with m rows and n columns is read. Firstly, the dimensions m and n are read, followed by the matrix elements.

Perform a min-max normalization of each column in the matrix i.e. each element in every column should be replaced with the value   where x is an element in a given column. Max and min are correspondingly the maximum and minimum elements in the corresponding column.

Print the transformed matrix on SO.

Explanation of the example:

The elements on the  0-th column are 1,6,11,16,21. Min is 1, max is 21.

Each element in the column should be transformed in the following way:

(1-1)/(21-1) = 0/20 = 0.00

(6-1)/(21-1) = 5/20 = 0.25

(11-1)/(21-1) = 10/20 = 0.50

(16-1)/(21-1) = 15/20 = 0.75

(21-1)/(21-1) = 20/20 = 1.00

Solution

include <stdio.h>  
  
**int** main()  
{  
 **int** m,n;  
 scanf(**"%d%d"**,&m,&n);  
  
 **int** M[100][100];  
 **double** N[100][100],min,max;  
  
 **for**(**int** i=0;i<m;i++)  
 {  
 **for**(**int** j=0;j<n;j++)  
 {  
 scanf(**"%d"**,&M[i][j]);  
 }  
 }  
  
 **for**(**int** i=0;i<n;i++){  
 min=max=M[0][i];  
 **for**(**int** j=0;j<m;j++){  
 **if**(M[j][i] < min){  
 min = M[j][i];  
 }  
 **if**(M[j][i] > max){  
 max = M[j][i];  
 }  
 }  
  
 **for**(**int** j=0;j<m;j++){  
 N[j][i] = (M[j][i] - min) / (max - min);  
 }  
 }  
  
 **for**(**int** i=0;i<m;i++){  
 **for**(**int** j=0;j<n;j++){  
 printf(**"%.2lf "**, N[i][j]);  
 }  
 printf(**"\n"**);  
 }  
  
 **return** 0;  
}

# 3.

A squared matrix with dimension n is read from SI. Firstly the number n is read, followed by the matrix elements.

If n is an odd number, the message ERROR should be printed.

Otherwise, you need to **fold** the matrix as demonstrated in the figure below. When folding the matrix the elements in the matching positions are summed.

Solution

#include **<stdio.h>  
  
int** main() {  
 **int** n;  
 scanf(**"%d"**, &n);  
  
 **if**(n % 2 != 0) {  
 printf(**"ERROR"**);  
 **return** 0;  
 }  
  
 **int** a[100][100];  
  
 **for**(**int** i = 0; i < n; i++) {  
 **for**(**int** j = 0; j < n; j++) {  
 scanf(**"%d"**, &a[i][j]);  
 }  
 }  
  
 **int** countR =0, countC = 0;  
 **for**(**int** i = n - 1; i >= n / 2; i--) {  
 **for**(**int** j = 0; j < n; j++) {  
 a[countR][countC] += a[i][j];  
 countC++;  
 }  
 countR++;  
 countC = 0;  
 }  
  
 countR = countC = 0;  
 **for**(**int** i = 0; i < n / 2; i++) {  
 **for**(**int** j = n - 1; j >= n / 2; j--) {  
 a[countR][countC] += a[i][j];  
 countC++;  
 }  
 countR++;  
 countC = 0;  
 }  
  
 **for**(**int** i = 0; i < n / 2; i++) {  
 **for**(**int** j = 0; j < n / 2; j++) {  
 printf(**"%d "**, a[i][j]);  
 }  
 printf(**"\n"**);  
 }  
  
 **return** 0;  
}

LAB 8

# 1.

Implement the functions:

* divisibleByK (int number, int k) - which returns 1 if k is a divisor of number, and 0 otherwise
* nextDivisibleByK (int number, int k) - which will return the first number larger than number that is divisible by k

Do not change the main function!

Explanation of the example: In each line, left of the array there is a number from the interval [10,20], while right from the array there is the first number larger than the number left from the array which is divisible by k (7).

**Bonus: Implement nextDivisibleByK with recursion.**

Solution

#include **<stdio.h>  
  
int** divisibleByK(**int** number, **int** k){  
 **return** number%k==0;  
}  
  
**int** nextDivisibleByK (**int** number, **int** k){  
 **if**((number+1)%k==0){  
 **return** number+1;  
 } **else**{  
 **return** nextDivisibleByK(number+1,k);  
 }  
}  
  
**int** main(){  
 **int** a, b, k;  
 scanf(**"%d %d %d"**, &a, &b, &k);  
  
 **for** (**int** i = a; i <= b; i++) {  
 printf(**"%d -> %d\n"**, i, nextDivisibleByK(i, k));  
 }  
 **return** 0;  
}

# 2.

The numbers A and B are read from SI. Print on SO all the numbers in the interval [A,B] which are palindromes and are contained only from the digits 0,1,2,3 and 4.   
  
**Plan to solve:**

* Implement a function reverseNumber (int number) which will calculate the reverse number of the number
* Implement a function isPalindrome (int number) which will return 1 if the number is palindrome and 0 otherwise.
* Implement a function containsDigits (int number) which will return 1 if all the digits in the number are 0,1,2,3 or 4, and 0 otherwise
* Use the functions isPalindrome and containsDigits in the main function

Extra: Make the function  containsDigits recursive.

Solution

#include **<stdio.h>**

**int** reverseNumber(**int** number){  
 **int** reverse=0;  
 **while** (number){  
 **int** ld=number%10;  
 reverse=10\*reverse+ld;  
 number/=10;  
 }  
 **return** reverse;  
}  
  
**int** isPalindrome(**int** number){  
 **return** number== reverseNumber(number);  
}  
  
**int** containsDigits(**int** number){  
 **int** ld=number%10;  
 **if**(number==0){  
 **return** 1;  
 }  
 **if**(ld>=0 && ld<5){  
 containsDigits(number/10);  
 } **else**{  
 **return** 0;  
 }  
}  
 **int** main(){  
 **int** a,b;  
 scanf(**"%d%d"**,&a,&b);  
 **for**(**int** i=a ; i<=b ; i++){  
 **if**(isPalindrome(i) && containsDigits(i)){  
 printf(**"%d\n"**,i);  
 }  
 }  
 **return** 0;  
}

# 3.

A number N is read from SI. Write a function form (int n) that will print a form (like in the example) based on the value of N.

**In order to achieve max points, you need to solve the task recursively. A non-recursive solution will be graded with 50% of the points.**

Solution

#include **<stdio.h>  
  
void** print(**int** n){  
 **if**(n==0){  
 printf(**"\n"**);  
 } **else**{  
 printf(**"\*"**);  
 print(n-1);  
 }  
}  
  
**void** stars(**int** n){  
 **if**(n==0){  
 **return**;  
 } **else**{  
 print(n);  
 stars(n-1);  
 }  
}  
  
**int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 stars(n);  
 **return** 0;  
}

LAB 9

# 1.

Write a function void sort(int x, int y, int z) which will sort the numbers a,b,c in decrising order i.e a will be the largest and c will be the smallest.

Do not change the main function.

Solution

#include **<stdio.h>  
  
void** sort(**int** \*x, **int** \*y, **int** \*z){  
 **if**(\*x<\*y){  
 **int** tmp=\*x;  
 \*x=\*y;  
 \*y=tmp;  
 }  
 **if**(\*x<\*z){  
 **int** tmp=\*x;  
 \*x=\*z;  
 \*z=tmp;  
 }  
 **if**(\*y<\*z){  
 **int** tmp=\*y;  
 \*y=\*z;  
 \*z=tmp;  
 }  
}  
  
**int** main() {  
 **int** a,b,c;  
 scanf(**"%d%d%d"**,&a,&b,&c);  
 sort(&a,&b,&c);  
 printf(**"%d %d %d"**,a,b,c);  
 **return** 0;  
}

# 2.

Write a fully recursive function triangle (n) which given the input argument n, prints a triangle from the numbers from 1 to n. Additional functions can be used, but no loops should be used.

Example for n=4.

1

12

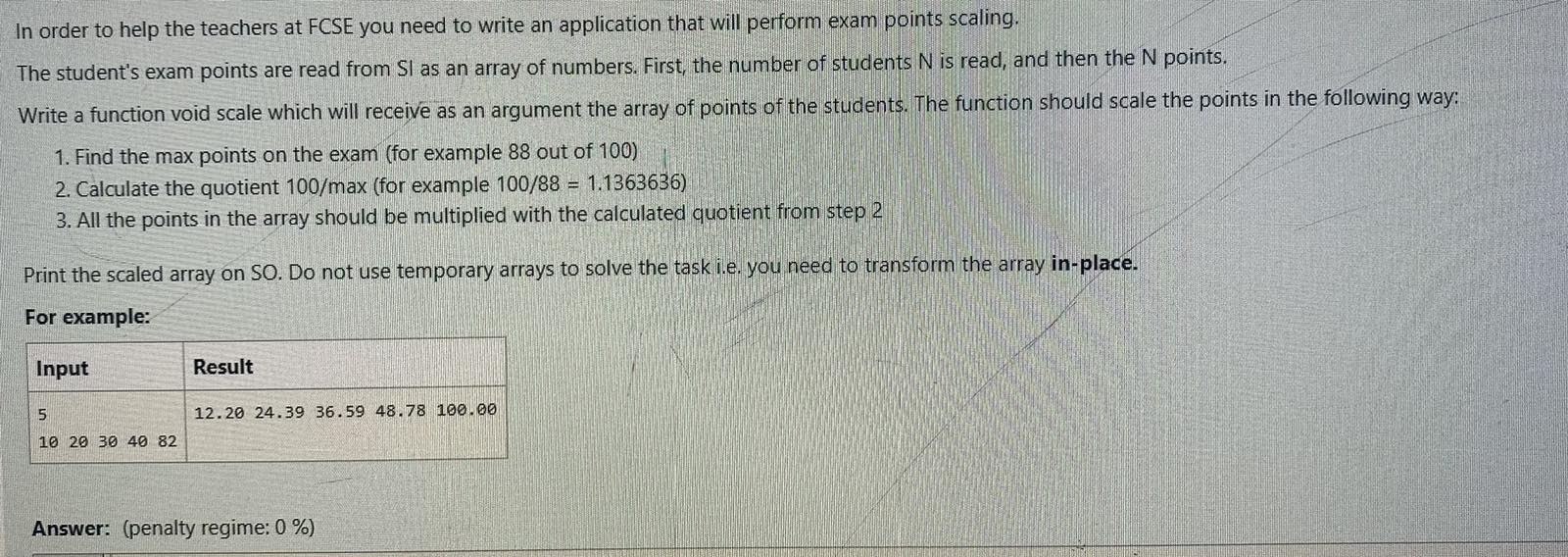
123

1234

Solution

#include **<stdio.h>  
  
void** numbers(**int** n, **int** m){  
 **if**(n==0){  
 **return**;  
 } **else**{  
 printf(**"%d"**,m-n+1);  
 numbers(n-1,m);  
 }  
}  
  
**void** triangle(**int** n, **int** m){  
 **if**(n==0){  
 **return**;  
 } **else**{  
 numbers(m-n+1,m-n+1);  
 printf(**"\n"**);  
 triangle(n-1,m);  
 }  
}  
  
**int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 triangle(n,n);  
 **return** 0;  
}

# 3.



Solution

#include **<stdio.h>  
  
void** scale (**double** \*a, **int** n){  
 **double** max=\*a;  
 **for**(**int** i=1; i<n ; i++){  
 **if**(\*(a+i)>max){  
 max=\*(a+i);  
 }  
 }  
 **double** quotient=100/max;  
 **for**(**int** i= 0 ; i<n ; i++){  
 \*(a+i)\*=quotient;  
 }  
}  
  
**int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **double** a[100];  
 **for**(**int** i=0 ; i<n ; i++){  
 scanf(**"%lf"**,a+i);  
 }  
 scale(a,n);  
 **for**(**int** i=0 ; i<n ; i++){  
 printf(**"%.2lf "**,\*(a+i));  
 }  
 **return** 0;  
}

LAB 10

# 1.

One string (with a max of 100 chars) is read from SI. Transform the string so the left and the right half of the string will switch their places.

Note: All test cases will have an even number of chars.

**For example:**

| Input | Result |
| --- | --- |
| Stefan | fanSte |
| Strukturno | turnoStruk |

Solution

#include **<stdio.h>**#include **<string.h>  
  
void** transform1(**char** \*str){  
 **int** len= strlen(str);  
 **char** tmp[100];  
 strncpy(tmp,str,len/2);  
 str=strcat(str+(len/2),tmp);  
 printf(**"%s"**,str);  
}  
  
**void** transform2(**char** \*str){  
 **char** tmp[100];  
 **for**(**int** i=0; i< strlen(str)/2 ; i++){  
 tmp[i]=str[i];  
 str[i]=str[i+strlen(str)/2];  
 str[i+strlen(str)/2]=tmp[i];  
 }  
}  
  
**int** main(){  
 **char** str[100];  
 fgets(str,**sizeof**(str),**stdin**);  
 str[strlen(str)-1]=**'\0'**;  
 transform1(str);  
 **return** 0;  
}

# 2.

A number N, number K and char C are read from SI. After than, N strings are read from SI (each of them in a new line)   
Print on SO all the strings which contain the char C exactly K times (ignore the case of the letters).  
If there are no such strings, print a message NONE.

Solution

#include **<stdio.h>**#include **<string.h>**#include **<ctype.h>  
  
int** containsC(**char** \*str,**char** C, **int** K){  
 **int** counter=0;  
 **for**(**int** i=0 ; i< strlen(str) ; i++){  
 **if**(tolower(str[i])== tolower(C)){  
 counter++;  
 }  
 }  
 **return** counter==K;  
}  
  
**int** main(){  
 **int** N,K;  
 scanf(**"%d%d\n"**,&N,&K);  
 **char** C;  
 scanf(**"%c\n"**,&C);  
 **for**(**int** i=0 ; i<N ; i++){  
 **char** str[100];  
 fgets(str,**sizeof**(str),**stdin**);  
 str[strlen(str)-1]=**'\0'**;  
 **if**(containsC(str,C,K)){  
 printf(**"%s"**,str);  
 }  
 }  
 **return** 0;  
}

# 3.

Write a function double relativeFrequency (char \* fileName, char letter) that will determine the relative frequency of a letter in a text file named 'fileName'!.

A relative frequency of a letter is calculated as the quotient between the count of occurrences of the letter and the total number of letters in the text. The case of the letter should be ignored (consider all letters as lower or upper letters!)

After that, write a function void printRelativeFrequencies(char \* fileName) that will utilize the previous function, and will calculate the relative frequencies of all the letters from the alphabet and then it will print them on standard output.

Solution

#include **<stdio.h>**#include **<ctype.h>  
  
double** relativeFrequency (**char** \* fileName, **char** letter){  
 FILE \*f= fopen(fileName,**"r"**);  
 **int** totalLetters=0, countL=0;  
 **char** c;  
 **while**((c=fgetc(f))!=**EOF**){  
 **if**(isalpha(c)){  
 totalLetters++;  
 }  
 **if**(tolower(letter)== tolower(c)){  
 countL++;  
 }  
 }  
 **return** (countL\*1.0)/totalLetters;  
}  
  
**void** printRelativeFrequencies(**char** \* fileName){  
 **for**(**char** c=**'A'** ; c<=**'Z'** ; c++){  
 printf(**"%c -> %.3lf\n"**,c, relativeFrequency(fileName,c));  
 }  
}  
  
**int** main(){  
 printRelativeFrequencies(**"C:\\Users\\user\\Desktop\\CCC\\Exams\\input.txt"**);  
 **return** 0;  
}