PROBLEMS

# 1.

Compute the sum: 1! + (1+2)! + (1+2+3)! + … + (1+2+…+n)! by using recursive functions.

# 2.

Write a program that for a given natural number will compute the difference between that number and the following prime number. The program should use recursive function for checking if a number is prime.

For the number 573, the program should print 577 – 573 = 4

# 3.

Write a recursive function that will compute the sum of the digits of a given number.

# 4.

Given a non-negative int n, compute recursively (no loops) the count of the occurrences of 8 as a digit, except that an 8 with another 8 immediately to its left counts double, so 8818 yields 4.

# 5.

Write a program that for given array of integers (read from SI) will print the greatest common divisor (GCD) of its elements. GCD should be computed using recursive function.

# 6.

Write a program that for given array of natural numbers (read from SI) will find and print the least common denominator (LCD) of its elements. The program should use recursive function for computing LCD of two numbers.

# 7.

Write a program that fro given array of integers (read from SI) will print the smallest element. The program should use recursive function for finding the smallest element of an array.

# 8.

Write a program that for given array of natural numbers (read from SI) will compute the sum. The program should use recursive function for computing the sum of array of integers.

# 9.

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.**

**For example:**

| **Input** | **Result** |
| --- | --- |
| 10 20 7 | 10 -> 14  11 -> 14  12 -> 14  13 -> 14  14 -> 21  15 -> 21  16 -> 21  17 -> 21  18 -> 21  19 -> 21  20 -> 21 |

# 10.

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.

# 11.

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.**

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5 | \*\*\*\*\*  \*\*\*\*  \*\*\*  \*\*  \* |

PROBLEMS

# 1.

*/\*Compute the sum: 1! + (1+2)! + (1+2+3)! + … + (1+2+…+n)! by using recursive functions.\*/*#include **<stdio.h>  
  
 int** sum ( **int** n){  
 **if**(n==1){  
 **return** 1;  
 } **else**{  
 **return** n+sum(n-1);  
 }  
}  
  
 **int** factorial(**int** n){  
 **if**(n==1){  
 **return** 1;  
 } **else**{  
 **return** n\* factorial(n-1);  
 }  
}  
  
 **int** sumFact(**int** n){  
 **if**(n==1){  
 **return** 1;  
 }**else**{  
 **return** factorial(sum(n))+ sumFact(n-1);  
 }  
}  
  
**int** main(){  
 **int** number;  
 scanf(**"%d"**,&number);  
 printf(**"%d"**, sumFact(number));  
 **return** 0;  
}

# 2.

*/\*Write a program that for a given natural number will compute the difference between that number and the following prime number.  
 \* The program should use recursive function for checking if a number is prime.  
For the number 573, the program should print 577 – 573 = 4  
\*/*#include **<stdio.h>  
  
 int** isPrime(**int** n, **int** divisor){  
 **if**(divisor==1){  
 **return** 1;  
 }  
 **if**(n%divisor==0){  
 **return** 0;  
 }  
 **return** isPrime(n, divisor - 1);  
 }  
  
 **int** findNextPrime(**int** n){  
 **if**(isPrime(n,n-1)){  
 **return** n;  
 } **else**{  
 **return** findNextPrime(n+1);  
 }  
 }  
  
**int** main(){  
 **int** number;  
 scanf(**"%d"**,&number);  
 printf(**"%d - %d = %d"**, findNextPrime(number+1),number, findNextPrime(number+1)-number);  
 **return** 0;  
}

# 3.

*/\*Write a recursive function that will compute the sum of the digits of a given number.\*/*#include **<stdio.h>  
  
 int** sum(**int** n){  
 **if**(n==0){  
 **return** 0;  
 } **else**{  
 **return** n%10 + sum(n/10);  
 }  
 }  
  
**int** main(){  
 **int** number;  
 scanf(**"%d"**,&number);  
 printf(**"%d"**, sum(number));  
 **return** 0;  
}

# 4.

*/\*Given a non-negative int n, compute recursively (no loops) the count of the occurrences of 8 as a digit,  
except that an 8 with another 8 immediately to its left counts double, so 8818 yields 4.\*/*#include **<stdio.h>  
  
int** countOccurrencesOf8(**int** n){  
 **if**(n==0){  
 **return** 0;  
 }  
 **int** ld=n%10;  
 **int** sl=n/10%10;  
  
 **if**(ld!=8){  
 **return** countOccurrencesOf8(n/10);  
 } **else**{  
 **if**(sl==8){  
 **return** 2+ countOccurrencesOf8(n/10);  
 } **else**{  
 **return** 1+ countOccurrencesOf8(n/10);  
 }  
 }  
}  
  
**int** main(){  
 **int** number;  
 scanf(**"%d"**,&number);  
 printf(**"%d"**, countOccurrencesOf8(number));  
 **return** 0;  
}

# 5.

*/\*Write a program that for given array of integers (read from SI) will print the greatest  
common divisor (GCD) of its elements. GCD should be computed using recursive function.\*/*#include **<stdio.h>  
  
int** gcdEuclidian(**int** m, **int** n){  
 **if**(m%n==0){  
 **return** n;  
 } **else**{  
 **return** gcdEuclidian(n,m%n);  
 }  
}  
  
**int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
  
 **int** array[100];  
  
 **for**(**int** i=0 ; i<n ; i++){  
 scanf(**"%d"**,&array[i]);  
 }  
 **int** gcd=gcdEuclidian(array[0],array[1]);  
 **for**(**int** i=2 ; i<n ; i++){  
 gcd=gcdEuclidian(gcd,array[i]);  
 }  
 printf(**"%d"**,gcd);  
 **return** 0;  
}

# 6.

*/\*Write a program that for given array of natural numbers (read from SI) will find and print the least  
common denominator (LCD) of its elements. The program should use recursive function for computing LCD of two numbers.\*/*#include **<stdio.h>  
  
int** lcd(**int** m, **int** n, **int** divisor){  
 **if**(m==1 && n==1){  
 **return** 1;  
 } **else**{  
 **if**(m%divisor==0 || n%divisor==0){  
 **if**(m%divisor==0){  
 m/=divisor;  
 }  
 **if**(n%divisor==0){  
 n/=divisor;  
 }  
 **return** divisor\* lcd(m,n,divisor);  
 } **else**{  
 **return** lcd(m,n,divisor+1);  
 }  
 }  
}  
  
**int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
  
 **int** array[100];  
  
 **for**(**int** i=0 ; i<n ; i++){  
 scanf(**"%d"**,&array[i]);  
 }  
 **int** lcdResult=lcd(array[0], array[1],2);  
 **for**(**int** i=2 ; i<n ; i++){  
 lcdResult=lcd(lcdResult, array[i],2);  
 }  
 printf(**"%d"**, lcdResult);  
 **return** 0;  
}

# 7.

*/\*Write a program that fro given array of integers (read from SI) will print the smallest element.  
The program should use recursive function for finding the smallest element of an array.\*/*#include **<stdio.h>**#include **<ctype.h>**#include **<string.h>  
  
int** minimum(**int** a[], **int** n, **int** min){  
 **if**(n<0){  
 **return** min;  
 } **else**{  
 **if**(a[n]<min){  
 min=a[n];  
 }  
 **return** minimum(a,n-1,min);  
 }  
}  
  
**int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** array[100];  
 **for**(**int** i=0 ; i<n ; i++){  
 scanf(**"%d"**,&array[i]);  
 }  
 **int** min=array[n-1];  
 min=minimum(array,n-1,min);  
 printf(**"%d"**,min);  
 **return** 0;  
}

# 8.

*/\*Write a program that for given array of natural numbers (read from SI) will compute the sum.  
The program should use recursive function for computing the sum of array of integers.\*/*#include **<stdio.h>**#include **<ctype.h>**#include **<string.h>  
  
int** sum(**int** a[], **int** n){  
 **if**(n==0){  
 **return** a[0];  
 } **else**{  
 **return** a[n]+ sum(a,n-1);  
 }  
}  
  
**int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** array[100];  
 **for**(**int** i=0 ; i<n ; i++){  
 scanf(**"%d"**,&array[i]);  
 }  
 printf(**"%d"**, sum(array,n-1));  
 **return** 0;  
}

# 9.

*/\*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.  
\*/*#include **<stdio.h>**#include **<ctype.h>**#include **<string.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;  
}

# 10.

*/\*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.  
  
\*/*

#include **<stdio.h>**#include **<ctype.h>**#include **<string.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;  
}

# 11.

*/\*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.  
\*/*#include **<stdio.h>**#include **<ctype.h>**#include **<string.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;  
}

# 12.