

Problema 1:

La suma de los dígitos del número $10^n - 1$ es 3798. Hallar n .

Solución: El número $10^n - 1$ es de la forma 999...99 en donde hay n nueves. Tenemos entonces que: $9 * n = 3798$. Lo cual da $n = 422$.

```
#include<stdio.h>
int main ()
{
    printf("422");
}
```

Problema 2:

Determinar para cuántas evaluaciones de variables lógicas (a, b, c) la siguiente expresión es verdadera:

$$(a \vee b) \rightarrow (a \wedge c)$$

Solución: La expresión es verdadera cuando $(a \vee b)$ es falso o cuando $(a \wedge c)$ es verdadero.

Para que $(a \wedge c)$ sea verdadero se debe de tener necesariamente que las variables a y c son verdaderas. Por ende $a = V$ y $c = V$. Entonces tenemos dos opciones en este caso: (a, b, c) es (V, F, V) o (V, V, V) .

Para que $(a \vee b)$ sea falso se debe de tener necesariamente que las variables a y b son falsas. Por ende $a = F$ y $b = F$. Entonces tenemos dos opciones en este caso: (a, b, c) es (F, F, V) o (F, F, F) . Juntando las soluciones para cada caso se tiene que hay 4 evaluaciones diferentes en la cuales la expresión es verdadera.

```
#include <stdio.h>
int main ()
{
    printf("4");
}
```

Problema 3:

Se denomina como *binaria* a una lista de unos y ceros. ¿Cuántas listas binarias de n números existen que contengan al menos dos unos?

Solución: Para obtener la cantidad de listas binarias que poseen al menos dos unos vamos a obtener la cantidad total de listas binarias y a esa cantidad le restamos la cantidad de listas binarias que no cumplen la condición del problema, es decir, aquellas que poseen exactamente 0 y 1 unos. Sabemos que hay en total 2^n listas binarias, que existen n con 1 uno y solo una con 0 unos. Entonces el la cantidad de listas binarias que buscamos va a ser de $2^n - (n + 1)$.

```
#include <stdio.h>
#include <math.h>
int main()
{
    int n;
    scanf("%d", &n);
```

```
int ans = pow(2,n) - (n + 1);
printf("%d",ans);
}
```

Problema 4:

Halla el perímetro de un triángulo rectángulo que tiene lados $n, x, x + 1$ en función de n sabiendo que el lado $x + 1$ es la hipotenusa del triángulo.

Solución: Dado que $x + 1$ es la hipotenusa del triángulo, sabemos que

$$(x + 1)^2 - x^2 = n^2 \implies 2x + 1 = n^2.$$

Por lo tanto, el perímetro del triángulo es

$$n + x + (x + 1) = n + 2x + 1 = n + n^2.$$

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

Problema 5:

Considera la sucesión:

$$\begin{aligned} a_1 &= 9 \\ a_2 &= 99 \\ a_n &= \underbrace{99 \dots 99}_{n \text{ dígitos}} \end{aligned}$$

Halla la suma de la sucesión

Solución: Si queremos añadir un 9 al final de un número, simplemente lo multiplicamos por 10 y le sumamos un 9 para completar el dígito de las unidades. Por lo tanto, para calcular $a_n = \underbrace{99 \dots 99}_{n \text{ dígitos}}$, podemos empezar con

el valor de 0 y realizar dicha operación n veces. Al mismo tiempo, sumamos todos los a_n .

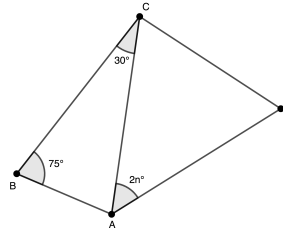
```
#include<stdio.h>
long long nterm (int n)
{
    long long num = 0;
    for (int i = 0; i < n; i++)
    {
        num = num * 10 + 9;
    }
    return num;
}
int main ()
```

```
{
    long long n;
    scanf("%lld", &n);
    long long sum = 0;
    for (int i = 1; i <= n; i++)
    {
        sum += nterm(i);
    }
    printf("%lld", sum);
}
```

También se puede utilizar `math.pow` para realizar las potencias y acortar la solución.

Problema 6:

En la figura se muestra un cuadrilátero $ABCD$. El ángulo $\angle CAD$ tiene el valor de $2n^\circ$ grados, donde n es un número entero positivo. Si $BC = AD$, ¿cuánto mide el ángulo $\angle ADC$ en grados en función de n ?



Solución: Se sabe que la suma de ángulos internos de un triángulo es 180° . Entonces

$$\angle ACB + \angle CBA + \angle BAC = 180^\circ$$

$$30^\circ + 75^\circ + \angle BAC = 180^\circ$$

$$\angle BAC = 180^\circ - 30^\circ - 75^\circ = 75^\circ.$$

Como $\angle CBA = \angle BAC = 75^\circ$, tenemos que $AC = BC$. Por dato $BC = AD$, entonces $AC = AD$. Como el triángulo DAC es isósceles, los ángulos $\angle ADC$ y $\angle ACD$ son iguales:

$$\angle ADC + \angle ACD + \angle CAD = 180^\circ$$

$$2\angle ADC + 2n^\circ = 180^\circ$$

$$2\angle ADC = 180^\circ - 2n^\circ$$

$$\angle ADC = \frac{180^\circ - 2n^\circ}{2} = (90 - n)^\circ.$$

```
#include<iostream>
using namespace std;
int main ()
{
```

```
int n;  
cin >> n;  
int resp = 90 - n;  
cout << resp;  
}
```

Problema 7:

Escribir un programa que imprima los enteros del 1 al 100 inclusive, bajo las siguientes condiciones:

- Para múltiplos de tres, imprimir Olimpiada en lugar del número.
- Para múltiplos de cinco, imprimir Informatica en lugar del número.
- Para múltiplos de cinco y tres, imprimir OlimpiadaInformatica en lugar del número.
- Para números que no cumplan ninguna de las condiciones anteriores, imprimir dicho número.

Solución:

```
int main () {  
    for (int i = 1; i <= 100; i++)  
    {  
        if (i % 3 == 0 && i % 5 == 0)  
        {  
            printf("OlimpiadaInformatica\n");  
        }  
        else if (i % 3 == 0)  
        {  
            printf("Olimpiada\n");  
        }  
        else if (i % 5 == 0)  
        {  
            printf("Informatica\n");  
        }  
        else  
        {  
            printf("%d\n", i);  
        }  
    }  
}
```

Problema 8:

Crea un programa que recibe un entero no negativo n y que imprime la cantidad de parejas de enteros no negativos (a, b) tales que $a^2 + b^2 = n$.

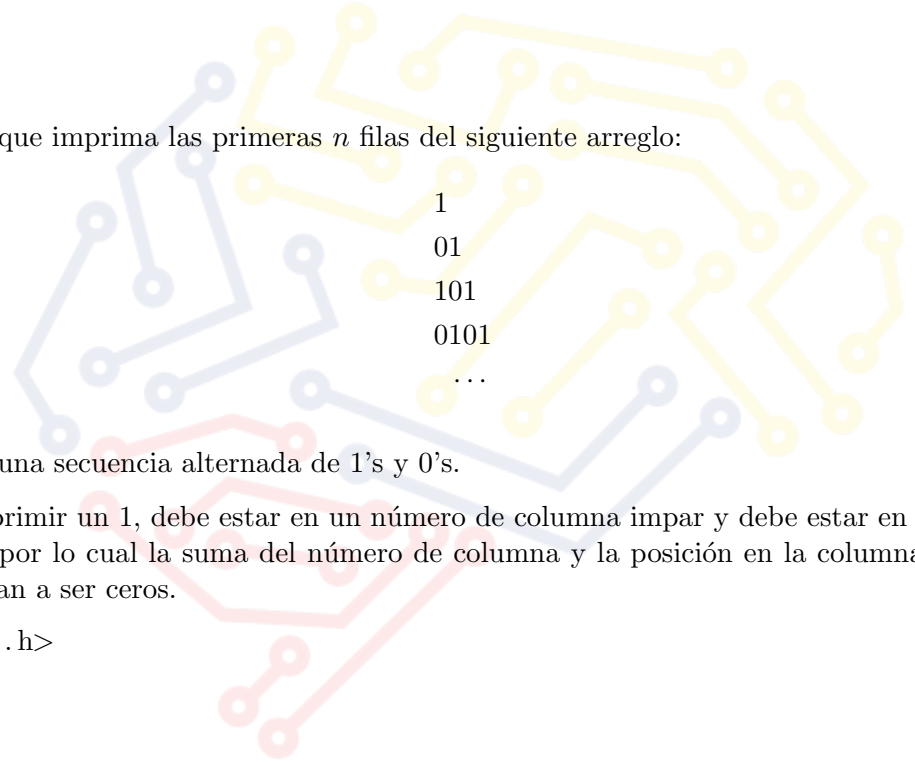
Solución:

```
#include<stdio.h>  
int main ()  
{  
    int n;  
    scanf("%d", &n);
```

```
int count = 0;
for (int i = 0; i <= n; i++)
{
    for (int j = 0; j <= n; j++)
    {
        if (i*i + j*j == n)
        {
            count++;
        }
    }
}
printf("%d", count);
}
```

Problema 9:

Crea un programa que imprima las primeras n filas del siguiente arreglo:



```
1
01
101
0101
...
```

Donde cada fila es una secuencia alternada de 1's y 0's.

Solución: Para imprimir un 1, debe estar en un número de columna impar y debe estar en una posición impar en dicha columna, por lo cual la suma del número de columna y la posición en la columna debe ser par, y el resto de números van a ser ceros.

```
#include<stdio.h>
```

```
int main(){
    int n;
    scanf("%d", &n);
    for(int i = 1; i <= n; i++){
        for(int j = 1; j <= i; j++){
            if((i + j) % 2 == 0)
                printf("1");
            else
                printf("0");
        }
        printf("\n");
    }
}
```

Problema 10:

Dadas dos secuencias de caracteres s_1 y s_2 con longitudes l_1 y l_2 , crea un programa que verifique si es posible permutar los caracteres de s_1 y obtener s_2 .

Solución: Primero verificamos que s_1 y s_2 tengan la misma longitud. De no ser así no existe ninguna permutación. Luego podemos ordenar lexicográficamente ambas cadenas y comparar si el resultado es idéntico. La complejidad de esta solución es $O(n \log n)$ en tiempo donde n es la longitud de ambas cadenas y $O(1)$ en espacio si el ordenamiento no usa espacio adicional.

```
#include<algorithm>
#include<iostream>
using namespace std;
int main () {
    std::string s1;
    std::string s2;
    cin >> s1;
    cin >> s2;
    int ans = 0;
    if (s1.size() != s2.size())
    {
        cout << ans;
    }
    std::sort(s1.begin(), s1.end());
    std::sort(s2.begin(), s2.end());
    if(s1 == s2)
    {
        ans = 1;
    }
    cout << ans;
}
```

Otra solución consiste en contar los caracteres de cada arreglo y verificar que son iguales. Esta solución tiene complejidad de $O(n)$ en expectancia si se usa una tabla hash para el conteo pero usa espacio $O(n)$.