Preconcurso LLamaCup 2024-I Editorial

GPC UNJFSC

25 de Mayo, 2024

Problema A: Semana UP

1. Descripción del Problema

El problema nos pide escribir un programa que, dado un número entre 1 y 7, imprima el día de la semana correspondiente. La asignación de los días es la siguiente:

- 1: lunes
- 2: martes
- 3: miércoles
- 4: jueves
- 5: viernes
- 6: sábado
- 7: domingo

La salida debe ser en minúsculas y sin acentos.

2. Entrada

Un número entero n ($1 \leq n \leq 7$).

3. Salida

El día de la semana correspondiente al número dado.

4. Ejemplo

- Entrada: 1
- Salida: lunes

5. Solución

Para resolver este problema, usaremos una estructura de control `switch` que nos permitirá imprimir el día de la semana correspondiente según el valor de n.

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    switch(n) {
        case 1: cout << "lunes"; break;
        case 2: cout << "martes"; break;
        case 3: cout << "miercoles"; break;
        case 4: cout << "jueves"; break;
        case 5: cout << "viernes"; break;
        case 6: cout << "sabado"; break;
        case 7: cout << "domingo"; break;
    }
    return 0;
}
```

6. Explicación del Código

- Primero, incluimos la librería `iostream` para poder usar la entrada y salida estándar.
- Usamos el espacio de nombres `std` para simplificar el código.
- Declaramos una variable entera n para almacenar el número ingresado por el usuario.

- Leemos el número ingresado usando `cin` n.
  Utilizamos una estructura `switch` para evaluar el valor de n y ejecutar el código correspondiente.
  Cada caso imprime el día de la semana correspondiente.

Problema B: Tiendita de Anita

7. Descripción del Problema

Anita ha decidido poner su propia tiendita clandestina de dulces en la escuela. Necesita un programa que calcule sus ganancias totales. El programa debe leer un número t que representa la cantidad de tipos de bolsas de dulces vendidas por Anita, seguido de t líneas donde cada línea contiene dos números: la cantidad de bolsas vendidas y el precio por bolsa.

8. Entrada

La entrada consiste en $t+1$ líneas. La primera línea contiene el número entero t ($1 \leq t \leq 100$), y las siguientes t líneas contienen dos enteros x y y ($1 \leq x, y \leq 1000$), representando la cantidad de bolsas vendidas y el precio por bolsa, respectivamente.

9. Salida

Un único número que indica las ganancias totales de Anita.

10. Ejemplo

Entrada:

```
5
1 3
2 5
3 15
8 45
2 31
```

Salida:

```
480
```

11. Solución

Para calcular las ganancias totales de Anita, simplemente multiplicamos la cantidad de bolsas vendidas por el precio de cada bolsa y sumamos estos valores para todas las líneas de entrada.

```
#include <iostream>
using namespace std;

int main() {
    int t, ganancias = 0;
    cin >> t;

    while(t--) {
        int x, y;
        cin >> x >> y;
        ganancias += x * y;
    }

    cout << ganancias;
    return 0;
}
```

Problema C: Suma de Parejas

12. Descripción del Problema

El problema consiste en leer un número variable de datos. Primero se lee un número t, que representa la cantidad de parejas de números a sumar. Luego se leen t parejas de números, y se debe escribir la suma de cada pareja.

13. Entrada

La entrada comienza con un número entero t ($1 \leq t \leq 100$), que indica la cantidad de parejas de números a sumar. Luego siguen t líneas, cada una conteniendo dos números enteros x y y ($-10^9 \leq x, y \leq 10^9$), representando las t parejas de números a sumar.

14. Salida

Se debe imprimir t líneas, cada una conteniendo la suma de los dos números de cada pareja.

15. Ejemplo

Entrada:

```
5
7 4
9 3
1 1
10 10
4 5
```

Salida:

```
11
12
2
20
9
```

16. Solución

Para resolver este problema, simplemente se deben leer t parejas de números, sumar los números de cada pareja y escribir la suma correspondiente.

```
#include <iostream>
using namespace std;

int main() {
    int t;
    cin >> t;
    while (t--) {
        int x, y, sum;
        cin >> x >> y;
        sum = x + y;
        cout << sum << endl;
    }
    return 0;
}
```

Problema D: Ternas Pitagóricas

17. Descripción del Problema

Una terna pitagórica consiste de tres enteros positivos a, b, c que cumplen con la expresión $a^2 + b^2 = c^2$. Escribe un programa que lea los enteros a, b, c y determine si se trata de una terna pitagórica.

18. Entrada

Una línea con los enteros a, b, c, separados por un espacio.

19. Salida

Imprime "SI"(en mayúsculas) en caso de tratarse de una terna pitagórica o "NO"si no es el caso.

20. Ejemplo

Entrada:

```
3 4 5
```

Salida:

```
SI
```

Es una terna pitagórica porque $3^2 + 4^2 = 5^2$, es decir $9 + 16 = 25$.

21. Solución

Para resolver este problema, definimos una función `isTenaPit()` que toma tres enteros a, b, c y comprueba si forman una terna pitagórica. Luego, en la función `main()`, leemos los valores a, b, c y llamamos a la función `isTenaPit()` para determinar si es una terna pitagórica. Dependiendo del resultado, imprimimos "SI.º "NO".

```
#include <iostream>
using namespace std;

bool isTenaPit(int a, int b, int c) {
    int powa = pow(a, 2);
    int powb = pow(b, 2);
    int powc = pow(c, 2);
    int sum_pit = powa + powb;
    return (sum_pit == powc);
}

int main() {
    int a, b, c;
    cin >> a >> b >> c;
    if (isTenaPit(a, b, c)) {
        cout << "SI" << endl;
    } else {
        cout << "NO" << endl;
    }
    return 0;
}
```

Problema E: Menor y Mayor de Cinco Números

22. Descripción del Problema

Escribe un programa que lea cinco números enteros y que escriba el menor de ellos seguido del mayor de ellos.

23. Entrada

Consiste de cinco números enteros, a_1, a_2, a_3, a_4, a_5, separados por espacios, que tendrán un valor entre 1 y 10^9.

24. Salida

Consiste de dos números enteros, m y M, separados por un espacio, donde m es el menor de los cinco números y M es el mayor de los cinco números.

25. Ejemplo

Entrada:

```
3 1 4 1 5
```

Salida:

```
1 5
```

26. Solución

Para resolver este problema, podemos leer los cinco números enteros y almacenarlos en un vector. Luego, ordenamos el vector para obtener el menor y el mayor número. Finalmente, imprimimos el menor y el mayor número obtenidos.

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int main() {
    vector<int> nums(5);
    for (int i = 0; i < 5; i++) {
        cin >> nums[i];
    }
    sort(nums.begin(), nums.end());
    cout << nums[0] << " " << nums[4] << endl;
    return 0;
}
```

Problema F: Secuencia 3n+1

27. Descripción del Problema

Dado un entero positivo n, el problema 3n+1 consiste en determinar si es posible llegar a 1 aplicando repetidamente las siguientes operaciones:

1. Si n es par, divídelo entre 2.
2. Si n es impar, multiplícalo por 3 y súmale 1.

Por ejemplo, si n comienza en 26, entonces toma los valores 26, 13, 40, 20, 10, 5, 16, 8, 4, 2, 1. Se cree que siempre es posible llegar a 1 sin importar en qué entero positivo se comience. Escribe un programa que calcule el número de pasos que se necesitan para que n se vuelva 1 y el valor más grande que toma n durante el cálculo.

28. Entrada

Un entero n. Puedes suponer que $n > 0$.

29. Salida

Dos enteros que denoten el número de pasos que se necesitan para que n se vuelva 1 y el valor más grande que toma n durante el cálculo.

30. Ejemplo

Entrada:

```
26
```

Salida:

```
10 40
```

31. Solución

Para resolver este problema, simplemente simulamos el proceso descrito en la descripción del problema. Iniciamos con n y en cada iteración aplicamos las operaciones definidas hasta que n sea igual a 1. Mientras hacemos esto, llevamos la cuenta del número de pasos y actualizamos el valor más grande que toma n.

```
#include <iostream>
using namespace std;

int main() {
    int n, con = 0, mayor = 0;
    cin >> n;

    while (n > 1) {
        if (n % 2 == 0) {
            n = n / 2;
        } else {
            n = (3 * n) + 1;
        }
        con++;
        mayor = max(mayor, n);
    }

    cout << con << " " << mayor;
    return 0;
}
```

Problema G: Suma de Dígitos

32. Descripción del Problema

El problema consiste en escribir un programa que lea un entero positivo y regrese la suma de sus dígitos. Por ejemplo, si el entero es 123, el programa deberá producir 6, ya que $1 + 2 + 3 = 6$.

Nota: Si tu programa usa una función recursiva, vale 50 puntos; si no usa una función recursiva, vale 30 puntos.

33. Entrada

Una línea con un entero positivo.

34. Salida

Un entero que representa la suma de los dígitos del número de entrada.

35. Ejemplo

Entrada:

```
123456789
```

Salida:

```
45
```

36. Solución

La solución puede implementarse fácilmente recorriendo cada dígito del número de entrada y sumándolos. En el código provisto, se recorre cada dígito convirtiendo el carácter a un entero y sumándolo a la variable 'sum'.

```
#include <iostream>
using namespace std;

int main() {
    string s;
    cin >> s;
    int sum = 0;
    for (int i = 0; i < s.size(); i++) {
        sum += s[i] - '0';
    }
    cout << sum;
    return 0;
}
```

Problema H: Suma de Columnas de una Matriz

37. Descripción del Problema

El problema consiste en calcular la suma de cada columna de una matriz cuadrada de tamaño $n \times n$. Se te proporciona una matriz $n \times n$, donde cada entrada de la matriz representa un valor en la matriz. Se requiere calcular la suma de cada columna y producir una salida que muestre estas sumas en orden, donde la primera suma corresponde a la primera columna, la segunda suma a la segunda columna y así sucesivamente.

38. Entrada

La entrada comienza con un entero n, indicando el tamaño de la matriz cuadrada. En las siguientes n líneas, habrá n enteros separados por espacios, que representan los valores de cada fila de la matriz.

39. Salida

La salida consiste en una línea con n enteros separados por espacios. Cada entero representa la suma de una columna de la matriz, donde el primer entero corresponde a la suma de la primera columna, el segundo entero a la segunda columna, y así sucesivamente.

40. Ejemplo

Entrada:

```
3
1 2 3
3 5 6
7 8 9
```

Salida:

```
11 15 18
```

41. Solución

La solución al problema es bastante directa. Se lee la matriz de tamaño $n \times n$ y luego se calcula la suma de cada columna recorriendo la matriz columna por columna. La suma de cada columna se imprime a medida que se calcula.

El código proporcionado recorre la matriz de entrada y calcula la suma de cada columna. Luego, imprime estas sumas separadas por espacios en una sola línea.

```
#include <iostream>
using namespace std;

int main() {
    int n;
    cin >> n;
    int matriz[n][n];

    // Leer la matriz de entrada
    for (int fila = 0; fila < n; fila++) {
        for (int columna = 0; columna < n; columna++) {
            cin >> matriz[fila][columna];
        }
    }

    // Calcular la suma de cada columna y mostrarla
    for (int columna = 0; columna < n; columna++) {
        int suma_columna = 0;
        for (int fila = 0; fila < n; fila++) {
```

```
            suma_columna += matriz[fila][columna];
        }
        cout << suma_columna << " ";
    }

    return 0;
}
```

Problema I: Anagramas

42. Descripción del Problema

Se te da una palabra A y una palabra B, y se te pide determinar si son anagramas. Dos cadenas son anagramas si contienen las mismas letras en diferentes órdenes. Por ejemplo, roma$\overline{z}$.ªmor"son anagramas.

43. Entrada

La entrada comienza con un entero N, que indica el número de casos de prueba. Luego sigue N líneas, cada una con dos cadenas de caracteres separadas por un espacio en blanco.

44. Salida

Para cada caso de prueba, imprime "si"si las dos cadenas son anagramas y "no.$^{\text{en}}$ caso contrario.

45. Ejemplo

Entrada:

```
2
roma amor
qwe wer
```

Salida:

```
si
no
```

46. Solución

Para resolver este problema, puedes ordenar las letras de ambas cadenas y luego comparar si son iguales. Si las dos cadenas tienen las mismas letras en diferentes órdenes, entonces son anagramas.

El código proporcionado lee las dos cadenas, las ordena y luego las compara. Si las dos cadenas ordenadas son iguales, imprime "si"; de lo contrario, imprime "no".

```
#include <iostream>
#include <algorithm>
using namespace std;

int main() {
    int N;
    cin >> N;
    while (N--) {
        string A, B;
        cin >> A >> B;
        sort(A.begin(), A.end());
        sort(B.begin(), B.end());
        if (A == B) {
            cout << "si" << endl;
        } else {
            cout << "no" << endl;
        }
    }
    return 0;
}
```