PARCIAL 2 -Programación 1

Rojo no sirve o ya estna hechos
Amarillo mejorar o poner ejemplo
Verde esta bien

1. Agudelo De Arce Andres Felipe

2. Agudelo

3. Maycol Becerra Castro

   Ejercicio 1
   Torre de Hanoi: Implementar una función que resuelva el problema de la torre de Hanoi para n discos.

   Ejemplo: Resolver el problema de la torre de hanoi para 3 discos, moviendo los discos de la torre A a la torre C utilizando la torre B
   Para este ejemplo, tenemos que mover tres discos desde la torre A a la torre C, utilizando la torre B como torre auxiliar. El objetivo es mover los discos de la torre A a la torre C, uno a uno, asegurándose de que nunca se coloque un disco más grande encima de uno más pequeño.

   Ejercicio 2
   Fibonacci: implementar una función que calcule el n-ésimo término de la serie de Fibonacci.

   Ejemplo: Calcular el décimo término de la serie de Fibonacci, que es 55.

   Ejercicio 3

4. Juan Esteban Becerra Valencia

5. Juan Esteban Agudelo Escobar

   Ejercicio 1

   Hallar el maximo comun divisor de dos numeros

   Ejemplo: Para hallar el maximo comun divisor de forma recursiva se puede usar el algoritmo de euclides
   El cual dice que tenemos a y b, si b es cero el mcd seria igual a a , de caso contrario el mcd seria igual al mcd de b entre el residuo de la division de a y b

   Ejercicio 2

   Implementa una funcion recursiva para calcular la suma de los elementos de un
   Arbol binario

   Ejemplo:

   Ejercicio 3

   Implementa una funcion recursiva que muestre todos los numeros de la sucesión de fibonacci hasta n

   Explicacion: la sucesion de fibonacci es igual a 0+1= 1 1+1=2 1+2=3 2+3=5 3+5= 8 y etc entonces tendrias que hacer una funcion que muestre todos la sucession de fibonacci hasta algun valor n por ejemplo puedes hacer un contador el cual sea igual a n pare la sucession

   Ejercicio 4

   Hallar la suma de 4 digitos de forma recursiva

   Explicacion: se puede usar la funcion de la suma de forma recursiva que hicimos en clase y eso anidarlo cogiendo de a 2 numeros los cuales tendrias 2 diferentes resultados y esos resultados se meterian de nuevo en la funcion para que esta nos de la suma

   Ejercicio 5

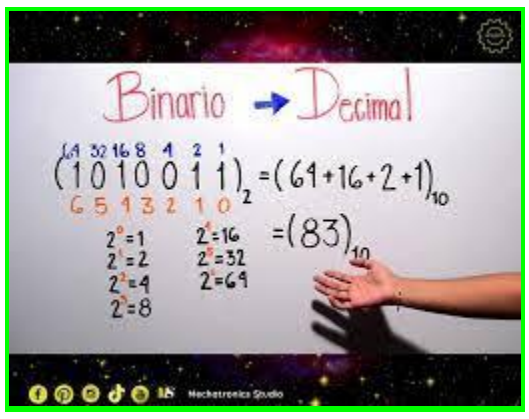
   Hallar el minimo comun divisor de dos numeros

   Explicacion : el minimo comun divisor de dos numeros es la multiplicación de estos dos numeros dividido entra el maximo comun divisor de estos dos numeros aca su puede usar la funcion que ya creamos antes del maximo comun divisor

6. Bernardo Castaño Silva
   Ejercicio 1
   Hacer un programa que lea números enteros por teclado hasta que el número ingresado cumpla con las siguientes condiciones:
   -Que el número sea mayor que 40
   -Que el número sea positivo
   -Que sea divisible entre 9
   Ejemplo:
   -Ingresar el número 45: Cumple con las condiciones

-Ingresar el número 27: No es mayor que 40
-Ingresar el número -36: Debe ser positivo

Ejercicio 2
Cree un programa que determine si un número es binario. Un número binario es aquel que está compuesto solamente de 1 y 0, de lo contrario mostrar “El número no es binario” y lo pase a sus respectivo decimal en base 10 como se ve en el siguiente ejemplo:



(1010011) base 2 -> (83)base 10

Ejercicio 3
Realizar la suma de dos cantidades sin el operador “+” y donde estos dos valores deben ser:
-Multiplos de 3
-Si están negativos, pasar a positivos
-Enteros, si son fracciones mostrar “Error”
Ejemplos:

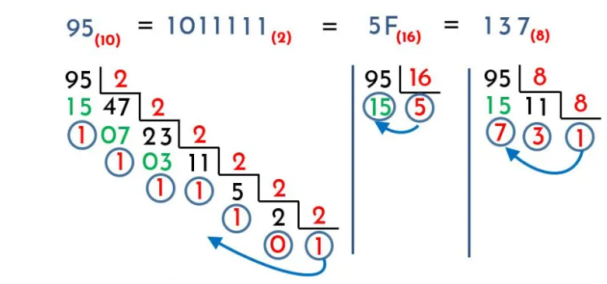
- -3 + 9 = 3+9 = 12
- 27+ 20 = “El 20 no es múltiplo de 3”
- 9/14 + 6 = “Error”

Ejercicio 4
Crear un programa que solucione la siguiente sumatoria, K(n, p) = p + 2 ∗ p + 3 ∗ p + 4 ∗ p + … + n ∗ p:

- El programa debe pedir al usuario que ingrese un número n, y un número p,
- Luego debe calcular el valor de K(n, p) usando una función recursiva,
- Debe imprimir el resultado de K(n, p)

Ejemplo:
Leer n=5 y p=2
Mostrar= 30

Ejercicio 5
Diseñe un programa que transforme un número entero positivo a notación binaria recursivamente (Tener en cuenta que la notación binaria es por medio de divisiones consecutivas sobre 2) **Por ejemplo:**



7. Kevin Castro Quintero
   Problema 1
   Realizar un programa que muestre por pantalla las tablas de multiplicar del 1 al 10 de cada número primo de 1 hasta n. (El n deberá estar en un rango de 1 a 20)
   Ejemplo:
   El usuario ingresa como n 10 asi que el programa deberá mostar

   2 x 1 = 2
   2 x 2 = 4
   2 x 3 = 6
   2 x 4 = 8
   2 x 5 = 10
   ……
   ……
   ……
   7 x 8 = 56
   7 x 9 = 63
   7 x 10 = 70

   Problema 2

Realizar un programa que al ingresar un número n este muestre todas las posibles sumas de los números naturales menores a él (deberá valorar que el número ingresado por el usuario no sea negativo).
Ejemplo:
El usuario ingresa por teclado el valor de 5, asi el programa deberá mostrar por pantalla
1 + 4 = 5
2 + 3 = 5
3 + 2 = 5
4 + 1 = 5

Problema 3
Realizar un programa que muestre la siguiente pirámide

```
    A
   BBB
  CCCCC
 DDDDDDD
```

Nota: Usar la función integer->char y el código ASCII

Problema 4

Realizar un programa que genere un número de teléfono de 10 dígitos el número generado por el programa siempre debe iniciar en el dígito 3.
Ejemplo:
Se corre el programa y el programa genera y muestra por pantalla:
3467890412

Problema 5

Realizar un programa que muestre la sucesión de los números dados por la fórmula (2n-1/n^2) según el número de términos ingresados por el usuario.
Ejemplo:

El usuario ingresa por teclado el número 5, el programa deberá mostrar por pantalla

1, 3/4, 5/9, 7/16, 9/25,

Ejemplo 2:

El usuario ingresa por teclado el número 2, el programa deberá mostrar por pantalla
1 , 3/4 ,

8. John Kevin Correa Morales

Ejercicio 1

En un cuadrado cuyo lado es a, se unen los puntos medios de sus 4 lados, formándose otro cuadrado cuyos puntos medios se unen también formando otro cuadrado, y así sucesivamente. Calcular la suma de los perímetros de los n primeros cuadrados así formados

Ejercicio 2

Diseñe e implemente un algoritmo que imprima todas las posibles descomposiciones de un número natural como suma de números menores que él.

1= 1

2 = 1+1

3= 2 + 1

3= 1+1+1

4= 3+1

4= 2+1+1

4 = 1+1+1+1

4=2+2

4=2+1+1

4=1+1+1+1

N = (n-1) +1

N = (n-2) + 2 = (n-2) + 1 + 1

Ejercicio 3

Programar un algoritmo recursivo que permita resolver el cuadrado latino.

Ejemplo

0 0 0 0 1

0 0 0 1 2

0 0 1 2 3

0 1 2 3 4

1 2 3 4 5

Ejercicio 4

Las **Torres de Hanoi** son un problema matemático bastante famoso que consiste en mover unos discos de diferentes tamaños de un poste a otro, respetando ciertas reglas. Este desafío ha sido analizado por matemáticos y científicos de la computación debido a su relevancia en la **teoría de algoritmos** y su aplicación en la informática. Para poder resolver Las Torres de Hanoi se requiere de habilidades en lógica, razonamiento y pensamiento creativo.

Para resolver el problema de Las **Torres de Hanoi**, se deben seguir ciertas reglas:

1. Se tienen tres varillas y una pila de **discos** de diferentes tamaños en una de las varillas.
2. El objetivo es mover toda la pila de discos a otra de las varillas, utilizando la tercera varilla como auxiliar.
3. Solo se puede mover un disco a la vez y no se puede colocar un disco más grande sobre uno más pequeño.

La solución se puede lograr utilizando la **recursividad**, donde se mueve la pila de discos como si fuera una unidad, y se repite el proceso para una pila de discos más pequeña.

ejemplo:

El **algoritmo** básico para resolver Las Torres de Hanoi con N discos sería:

1. Mover la pila de discos superiores (N-1) de la varilla inicial a la varilla auxiliar.
2. Mover el disco más grande (N) de la varilla inicial a la varilla de destino.
3. Mover la pila de discos superiores (N-1) de la varilla auxiliar a la varilla de destino.

Ejercicio 5

Crea un programa donde ingreses 10 nombres de ciudades, y en un método recursivo muestra las ciudades al revés (es decir del último al primero)

Ejemplo entrada:

Bogota

Pereira

Cucuta

Cali

cartagena

Ejemplo salida:

Cartagena

Cali

Cucuta

Pereira

bogota

8.Jacobo ceballos
Ejercicio 1:
Imprime todas las combinaciones de números del 1 al `n` con suma `n`
Dado un entero positivo n, imprime todas las combinaciones de números entre 1 y n teniendo suma n.

```
For n = 5, the following combinations are possible:


{ 5 }
{ 1, 4 }
{ 2, 3 }
{ 1, 1, 3 }
{ 1, 2, 2 }
{ 1, 1, 1, 2 }
{ 1, 1, 1, 1, 1 }
```

Ejercicio 2:

Programe un método recursivo que transforme un número entero positivo a notación binaria.

Ejercicio 3:

Encuentre todas las rutas desde la primera celda hasta la última celda de una matriz

Dado un M × N matriz entera, encuentre todas las rutas desde la primera celda hasta la última celda. Solo podemos movernos hacia abajo o hacia la derecha desde la celda actual

```
Input:


[ 1  2  3 ]
[ 4  5  6 ]
[ 7  8  9 ]


Output:


1, 2, 3, 6, 9
1, 2, 5, 6, 9
1, 2, 5, 8, 9
1, 4, 5, 6, 9
1, 4, 5, 8, 9
1, 4, 7, 8, 9
```

Ejercicio 4:

Encuentra todas las combinaciones posibles de palabras formadas desde el teclado del móvilDada una secuencia de números entre 2 y 9, imprime todas las combinaciones posibles de palabras formadas desde el teclado del móvil que tiene alfabetos en inglés asociados a cada tecla.


```
Input: [2, 3, 4]


Output:


ADG BDG CDG AEG BEG CEG AFG BFG CFG ADH BDH CDH AEH BEH
CEH AFH BFH CFH ADI BDI CDI AEI BEI CEI AFI BFI CFI
```

Ejercicio 5:

Realizar un código que cumpla la función de una división mediante restas sucesivas en racket.

10.Jose Felipe Duarte Coronado

Ejercicio 1: Calcular la secuencia de números de Fibonacci modificada.

En este ejercicio, en lugar de calcular la secuencia de Fibonacci regular (0, 1, 1, 2, 3, 5, 8, ...), se calculará una secuencia modificada en la que cada número es la suma de los dos números anteriores más su posición en la secuencia (1-indexada). La secuencia comienza con 0 y 1 como los dos primeros elementos. Por ejemplo, el tercer elemento sería 0 + 1 + 3 = 4, el cuarto elemento sería 1 + 4 + 4 = 9, y así sucesivamente.

Ejemplo 1: Entrada: 5 Salida: 0, 1, 4, 9, 19

Ejemplo 2: Entrada: 7 Salida: 0, 1, 4, 9, 19, 37, 68

Ejercicio 2

Ejercicio: Generar todas las posibles combinaciones de pares de números válidos.

Dado un número entero n, genera todas las posibles combinaciones de pares de números válidos utilizando n pares de números, donde el primer número del par es menor o igual al segundo número. Utiliza recursividad para resolver este problema.

Ejemplo 1:

Entrada: 2

Salida: (1,1)(2,2), (1,2)(2,2)

Ejemplo 2:

Entrada: 3

Salida: (1,1)(2,2)(3,3), (1,1)(2,3)(3,3), (1,2)(2,2)(3,3), (1,2)(2,3)(3,3)

Este ejercicio requiere el uso de recursividad para generar las combinaciones válidas de pares de números. Puedes abordar el problema utilizando una función auxiliar recursiva que construya las combinaciones válidas de pares de números de manera incremental, manteniendo un conteo de los números disponibles en el rango de 1 a n, y asegurándote de que cada paso en la construcción mantenga la validez de la combinación de pares de números (es decir, el primer número en el par siempre debe ser menor o igual al segundo número).

Ejercicio 3
Calcular la cantidad de dígitos en un número entero.

Dado un número entero n, encuentra la cantidad de dígitos en el número utilizando recursividad. No se permite el uso de listas, cadenas de caracteres ni funciones de conversión de tipos de datos.

Ejemplo 1: Entrada: 123 Salida: 3

Ejemplo 2: Entrada: 56789 Salida: 5

Ejercicio 4
Ejercicio: Calcular la raíz cuadrada de un número utilizando el método de aproximación de Newton.

Dado un número positivo x y una tolerancia de error ε, calcula la raíz cuadrada de x utilizando el método de aproximación de Newton con recursión. No se permite el uso de listas ni funciones de biblioteca para calcular la raíz cuadrada directamente.

El método de aproximación de Newton para calcular la raíz cuadrada de un número x es el siguiente:

Comienza con una estimación inicial a (por ejemplo, a = x).

Calcula la siguiente estimación a partir de la estimación actual utilizando la fórmula: a' = 0.5 * (a + x / a).

Si la diferencia absoluta entre a y a' es menor o igual que ε, devuelve a'.

De lo contrario, repite el proceso con a' como la nueva estimación.

Ejemplo 1:

Entrada: x = 25, ε = 0.0001

Salida: 5.0000 (con precisión de 4 decimales)

Ejemplo 2:

Entrada: x = 2, ε = 0.0001

Salida: 1.4142 (con precisión de 4 decimales)

Este ejercicio requiere el uso de recursividad para calcular la raíz cuadrada de un número utilizando el método de aproximación de Newton. Puedes abordar el problema utilizando una función recursiva que realice las iteraciones del método de Newton, verificando la tolerancia de error en cada llamada recursiva.

Aquí tienes un ejemplo de cómo calcular la raíz cuadrada de 25 utilizando el método de aproximación de Newton. Usaremos una tolerancia de error ε = 0.0001.

Paso 1: Comienza con una estimación inicial a = x = 25.

Paso 2: Calcula la siguiente estimación a partir de la estimación actual utilizando la fórmula: a' = 0.5 * (a + x / a) = 0.5 * (25 + 25 / 25) = 0.5 * (25 + 1) = 0.5 * 26 = 13

Paso 3: Verifica si la diferencia absoluta entre a y a' es menor o igual que ε. |25 - 13| = 12 > 0.0001, por lo que debemos continuar con el proceso.

Paso 4: Repite el proceso con a' = 13 como la nueva estimación: a' = 0.5 * (a + x / a) = 0.5 * (13 + 25 / 13) ≈ 0.5 * (13 + 1.92308) ≈ 0.5 * 14.92308 ≈ 7.46154

Paso 5: Verifica si la diferencia absoluta entre a y a' es menor o igual que ε. |13 - 7.46154| ≈ 5.53846 > 0.0001, por lo que debemos continuar con el proceso.

Paso 6: Repite el proceso con a' ≈ 7.46154 como la nueva estimación: a' = 0.5 * (a + x / a) ≈ 0.5 * (7.46154 + 25 / 7.46154) ≈ 0.5 * (7.46154 + 3.34961) ≈ 0.5 * 10.81115 ≈ 5.40558

Paso 7: Verifica si la diferencia absoluta entre a y a' es menor o igual que ε. |7.46154 - 5.40558| ≈ 2.05596 > 0.0001, por lo que debemos continuar con el proceso.

Paso 8: Repite el proceso con a' ≈ 5.40558 como la nueva estimación: a' = 0.5 * (a + x / a) ≈ 0.5 * (5.40558 + 25 / 5.40558) ≈ 0.5 * (5.40558 + 4.62317) ≈ 0.5 * 10.02875 ≈ 5.01438

Paso 9: Verifica si la diferencia absoluta entre a y a' es menor o igual que ε. |5.40558 - 5.01438| ≈ 0.39120 > 0.0001, por lo que debemos continuar con el proceso.

Paso 10: Repite el proceso con a' ≈ 5.01438 como la nueva estimación: a' = 0.5 * (a + x / a) ≈ 0.5 * (5.01438 + 25 / 5.01438) ≈ 0.5 * (5.01438 + 4.98567) ≈ 0.5 * 10.00005 ≈ 5.00003

Paso 11: Verifica si la diferencia absoluta entre a y a' es menor o igual que ε.

|5.01438 - 5.00003| ≈ 0.01435 > 0.0001, por lo que debemos continuar con el proceso.

Paso 12: Repite el proceso con a' ≈ 5.00003 como la nueva estimación:

a' = 0.5 * (a + x / a) ≈ 0.5 * (5.00003 + 25 / 5.00003) ≈ 0.5 * (5.00003 + 4.99997) ≈ 0.5 * 10 ≈ 5.00000

Paso 13: Verifica si la diferencia absoluta entre a y a' es menor o igual que ε.

|5.00003 - 5.00000| ≈ 0.00003 <= 0.0001, por lo que podemos detener el proceso.

El resultado es aproximadamente 5.00000 con una precisión de 5 decimales. Por lo tanto, la raíz cuadrada de 25 usando el método de aproximación de Newton es aproximadamente 5.00000.

Ejercicio 5

Generar un patrón de triángulo de Sierpinski utilizando recursión.

Dado un nivel de profundidad n, imprime un patrón de triángulo de Sierpinski utilizando recursión. El triángulo de Sierpinski es un fractal que se crea dividiendo un triángulo equilátero en 4 triángulos equiláteros más pequeños, y luego haciendo lo mismo con los 3 triángulos equiláteros resultantes (sin el triángulo central) de forma recursiva. Puedes representar el triángulo de Sierpinski utilizando caracteres de texto, como asteriscos (*) o cualquier otro símbolo.

```
Ejemplo 1 (n = 1):

 *
* *

Ejemplo 2 (n = 2):

   *
  * *
 *   *
* * * *

Ejemplo 3 (n = 3):

       *
      * *
     *   *
    * * * *
   *       *
  * *     * *
 *   *   *   *
* * * * * * * *
```

Este ejercicio requiere el uso de recursión para crear un patrón de triángulo de Sierpinski de n niveles de profundidad. Puedes abordar el problema utilizando una función recursiva que divida el triángulo en triángulos más pequeños, imprimiendo cada triángulo equilátero en función de su posición y nivel de profundidad.

11. Luis Miguel Echeverri Vélez

Ejercicio numero 1

La novia de Juan es apasionada por la programación en racket, y para darle una sorpresa, este quiere diseñar un programa que muestre la primera letra de su nombre (Isabella) y mostrarla en una pantalla de fondo para sorprenderla.

Existe un pequeño problema, Juan no sabe de qué tamaño será la pantalla, pero el mínimo tamaño para que se vea bien debe ser 25 pulgadas, y el maximo 50.

Teniendo en cuenta que para 25 la letra I grande debe estar formada por 10 filas de I's formadas por 3 cada una.
A partir de este tamaño, por cada pulgada el numero de filas aumentará en 1, así como por cada 3 pulgadas a partir de 25 se sumará una I a cada fila.

Ejemplo:

El dia de la sorpresa, Juan digita en pulgadas 25, por lo que en el programa saldría de la siguiente manera:

III
III
III
III
III
III
III
III
III
III

Ejemplo 2:
Si Juan digita 27, al correr el programa deberá salir lo siguiente:

IIII
IIII
IIII
IIII
IIII
IIII
IIII
IIII
IIII
IIII
IIII

FIN, si logras crear el programa, el plan maestro de Juan será un éxito. ¿ Que esperas para ayudarlo?
.

Ejercicio número 2

Elabore un programa que sirva para calcular el máximo común divisor (MCD) de dos números:

El Máximo Común Divisor o MCD se define como el mayor número que divide exactamente dos o más números, en el caso del ejercicio a plantear, dos.

Pd: Para la función recursiva utilizar el aritmético REMAINDER.

Ejemplo:

El usuario Ingresa 12 como N1 y 20 como N2
Resultado: la función debe dar como resultado 4, pues este es el número que divide a ambos sin dejar residuo.

El usuario Ingresa 15 como N1 y 45 como N2
Resultado: la función debe dar como resultado 15, pues este es el número que divide a ambos sin dejar residuo.

Ejercicio número 3

Hacer una funcion recursiva en racket que muestre los cuadrados de un número n y los de los números que se anteponen a este si es mayor a 0.

Ejemplo:
El Usuario le da el valor de 5 a n
Al correr la función esta debe dar los cuadrados de 1,2,3,4 y 5.

Siendo así, el resultado sería:
(1 4 9 16 25)

Ejercicio número 4

El profesor de Miguel necesita saber cuantos aprobaron el curso basándose en la nota del examen final, para ello, Miguel, que no lo presentó, es abordado por el profesor, quien le propone que diseñe un programa para ganar la nota, este se basa en lo siguiente:
El usuario ingresará el nombre del estudiante, al frente de este aparecerá APROBADO, PENDIENTE o REPROBADO.

Básese en lo siguiente:
- En el colegio de Miguel los exámenes se ganan con una calificación mayor a 3.5, AUNQUE, si la calificación es mayor a 3 pero menor a 3.5, el profesor podrá elegir si ayudarle o no, por eso la calificación quedaría en pendiente.
- Toda calificación menor a 3 será automáticamente reprobada.

Ejemplo:
El programa mostrará lo siguiente:

Felipe: APROBADO

Carlos: PENDIENTE

Manuel**:** REPROBADO

Ejercicio 5
El salario mínimo en Colombia se ubica en 1.000.000 Pesos colombianos, supongamos que de un sueldo n, hay que pagar un 20% de impuestos, hacer una función recursiva en racket que según el sueldo ingresado muestre este sin el 20% y abajo de este los numeros inferiores sin el 20% hasta llegar al sueldo mínimo menos el 20%, es decir 800.000.

Ejemplo:

El usuario digita 1500000,

Al correr el programa este debe mostrar lo siguiente:

1200000
1199999.2
1199998.4
1199997.6

Y así sucesivamente hasta que muestre el millón de pesos menos el 20% planteado

12. Miguel Angel Florian Gonzalez

Ejercicio1
Escribe una función recursiva en Racket llamada `sucesion_collatz` que reciba un número entero `n`. La función debe devolver el número de términos que tiene la sucesión de Collatz para `n`.

La regla de la sucesión de Collatz es la siguiente: si el número es par, se divide por dos, si es impar, se multiplica por tres y se suma uno. Por ejemplo, para `n=6`, la sucesión de Collatz sería: 6, 3, 10, 5, 16, 8, 4, 2, 1. En este caso, la función debería devolver el valor 9, ya que la sucesión tiene 9 términos.

Para resolver este problema, puedes utilizar una función recursiva que llame a sí misma con el número siguiente en la sucesión de Collatz, hasta llegar al número 1. Cada vez que se llama a la función, se incrementa un contador para llevar la cuenta del número de términos de la sucesión.

Ejercicio 2
Realizar un programa que le solicite al usuario que ingrese un número mayor de dos cifras y el programa debe devolver el numero invertido.
En este programa solo se debe realizar utilizando una sola función y sin utilizar let
Ejemplo
Si el usuario ingresa el numero 1234 el programa deberá devolver 4321

Ejercicio 3

Ejercicio: Permutaciones de un número

Descripción del problema

Escribe una función que encuentre todas las permutaciones posibles de un número dado. Por ejemplo, si se le proporciona el número "123", la función deberá generar todas las posibles permutaciones de ese número: “123", "132", "213", "231", "312" y "321".

Especificaciones de la función

La función debe llamarse permutaciones y debe tomar dos parámetros:

Elementos: un número que representa los elementos a permutar.

-Permutación Parcial: una cadena de números que representa una permutación parcial de los elementos.

La función debe imprimir todas las permutaciones posibles de los elementos. No se permiten listas ni arreglos, solo se puede utilizar recursividad y variables.

Ejemplo de uso

Si una persona ingresa 123 el resultado esperado será:

123

132

213

231

312

321

Ejercicio 4

Realizar un programa en racket, el cual solicite un número de tres cifras y un solo número, el programa deberá devolver en que

posición está el número indicado.

El programa también deberá de responder si el numero ingresado es menor de tres cifras o es mayor de tres cifras en este caso deberá responder "El número debe tener tres cifras" ya que el número a buscar solo cumple cuando hay tres cifras, y si la persona ingresa el numero a buscar menor que 0 el programa tendrá que responder: “el número debe ser mayor a 0” y también si la persona ingresa el numero a buscar mayor que 9 el programa tendrá que responder: “el número debe ser menor a 9”.

En tal caso de que por ejemplo el número de tres cifras ingresado es 320 y el número a buscar es 4 el programa tiene que responder “el número buscado no se encontró” .

Ejemplo: si el usuario ingresa el número “ 1 ” y el número “ 213 ” el programa deberá devolver 2, porque esta es la posición donde se encuentra el número 1.

Ejercicio 5

Escribe una función recursiva que calcule el mínimo común múltiplo (mcm) de dos números enteros positivos a y b. El mcm es el número más pequeño que sea múltiplo de ambos números.

La función debe tomar como entrada dos números enteros positivos a y b, y debe ser recursiva.

14. Sarita Giraldo Cardona

Programa 1

Realice un algoritmo que me permita calcular la multiplicación de los números impares terminados en 5 entre un rango de número de n hasta m.
ejemplo: suponiendo que n es 6 y m es 25
realizar la multiplicación de los números terminados en 5
en este caso 15 * 25= 375.

Programa 2
Realizar un programa que multiplique los números enteros impares y negativos
Comprendidos de n hasta -8
ejemplo
Suponiendo que n es 8, solo me va multiplicar -1*-3*-5*-7=105

Programa 3
Hacer un programa que reciba un número y cumpla con las siguientes condiciones

- Si el número es menor que 10: calcular el factorial

Ejemplo: si es 5 : 5*4*3*2*1=120

- Si el número es mayor que 10: ingresar otro argumento, que cumpla la funcion de ser el exponente del número n, y calcular la potencia.

Ejemplo: si n es 5 y el exponente 2= 5^2= 25 (SIN USAR EXP)

Programa 4
Realice un algoritmo con función recursiva que me pueda mostrar en pantalla la cantidad de dígitos que tiene un número.
EJEMPLO: ingresa el número 356, el programa me debe de mostrar el número 3, que son los dígitos que tiene 356.

Programa 5
Construir un programa que me permita hallar el resultado de la multiplicación entre los mismos dígitos.
EJEMPLO: ingresa el número 68, el resultado debería ser 48 (6*8=48)

15. Isabela Henao García

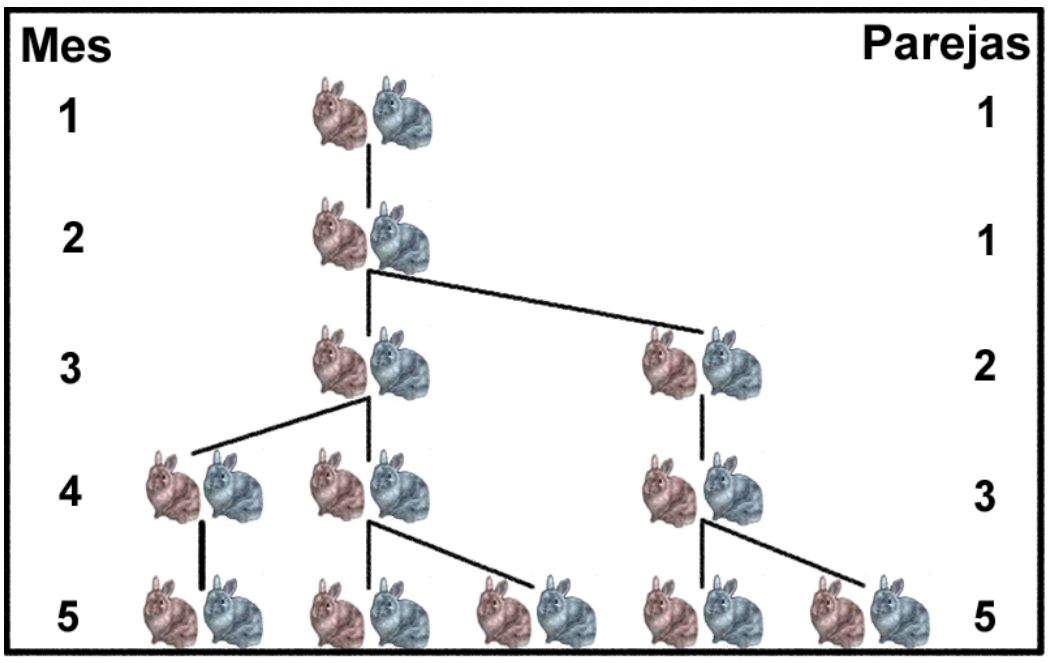
1.EJERCICIO.
Crear un programa recursivo que invierta un número ingresado y lo muestre
Ejemplos: entrada: 123 salida:321 (cualquier cantidad de números)

2.EJERCICIO:SUCESIÓN DE FIBONACCI

El ejercicio de Fibonacci pregunta cuántas parejas de conejos habrá en una granja luego de 12 meses, si se coloca inicialmente una sola pareja y se parte de las siguientes premisas:

1. **Los conejos alcanzan la madurez sexual a la edad de un mes.**
2. **En cuanto alcanzan la madurez sexual los conejos se aparean y siempre resulta preñada la hembra.**
3. **El periodo de gestación de los conejos es de un mes.**

4. Los conejos no mueren.
5. La hembra siempre da a luz una pareja de conejos de sexos opuestos.
6. Los conejos tienen una moral y un instinto de variedad genética muy relajados y se aparean entre parientes.



Ejemplo:para 12 meses se tendrán 144 parejas de conejos

CREAR UN PROGRAMA QUE RECIBA UN NÚMERO X DE MESES PARA CALCULAR CUÁNTAS PAREJAS DE CONEJOS HAY EN TOTAL.

3.EJERCICIO

Supóngase que una persona se mete a una piscina cuya profundidad es de 5 metros. Su intención es tocar el fondo de la piscina y después salir a la superficie,en el descenso se le va indicando la distancia desde la superficie (a cada metro).

Ejemplo:1 metro: 4 metros por debajo de la superficie.
2 metros: 3 metros por debajo de la superficie.
3 metros: 2 metros por debajo de la superficie.
4 metros: 1 metro por debajo de la superficie.
5 metros: toca el fondo de la piscina.
4 metros: 1 metro por debajo de la superficie.
3 metros: 2 metros por debajo de la superficie.
2 metros: 3 metros por debajo de la superficie.
Después de tocar el fondo de la piscina, la persona comienza su ascenso y se le indica la distancia a la superficie a la orilla:
1 metro: 4 metros por debajo de la superficie.
0 metros:llega a la superficie

CREAR UN PROGRAMA QUE RECIBA LA ESTATURA X DE LA PERSONA Y LE INDIQUE LA DISTANCIA DESDE LA SUPERFICIE (CADA METRO)

4.EJERCICIO
Combinaciones de n elementos tomados de m en m:
Escriba una función recursiva que tome dos números enteros positivos n y m como entrada y devuelva el número de combinaciones de n elementos tomados de m en m.
Ejemplo: Si n es 5 y m es 3, la función debería devolver 10, ya que hay 10 formas diferentes de elegir 3 elementos de un conjunto de 5 elementos.

5.EJERCICIO
Calcular el Máximo Común Divisor de dos números
Ejemplo: MCD: 123,345= 3

16. Daniel Felipe Holguin Castañeda

1) **Problema de sumatoria recursiva**

Escriba una función recursiva en Racket que tome un número entero positivo como entrada y calcule la sumatoria de todos los números enteros positivos desde 1 hasta el número de entrada.
Ejemplo:
Si se ingresa el número 5, la función debe calcular la sumatoria de los números 1+2+3+4+5, que es igual a 15.

2) **Problema de cálculo de factorial recursivo**

Escriba una función recursiva en Racket que tome un número entero positivo como entrada y calcule su factorial.
Ejemplo:
Si se ingresa el número 4, la función debe calcular el factorial de 4, que es igual a 4x3x2x1, es decir 24.

3) **Problema de búsqueda binaria recursiva:**

Escriba una función recursiva en Racket que tome una lista ordenada de números enteros y un número entero como entrada, y busque el número en la lista utilizando el algoritmo de búsqueda binaria recursiva.
Ejemplo:
Si se ingresa la lista [1, 3, 5, 7, 9] y el número 5, la función debe devolver el índice donde se encuentra el número en la lista, que en este caso es 2.

4) **Problema de Fibonacci recursivo:**

Escriba una función recursiva en Racket que tome un número entero positivo como entrada y calcule el número de Fibonacci correspondiente.
Ejemplo:
Si se ingresa el número 6, la función debe calcular el sexto número de Fibonacci, que es igual a 8.

5) **Problema de potencia recursiva:**

Escriba una función recursiva en Racket que tome dos números enteros positivos como entrada, base y exponente, y calcule la potencia de la base elevada al exponente.
Ejemplo:
Si se ingresa la base 2 y el exponente 3, la función debe calcular 2 elevado a la 3, que es igual a 8.

18. Angel David Luna Ospina

1. Plantear el algoritmo de euclides para hallar el MCD de forma recursiva:

$$Euclides(a,\ b) = \begin{cases} b & si\ \ b \leq a \quad \text{y} \quad (a\ mod\ b) = 0 \\ Euclides(b,\ a) & si\ \ a < b \\ Euclides(b,\ (a\ mod\ b)) & de\ \ otro\ \ modo \end{cases}$$

2. Crear un función que retorne #t si un número n es primo o #f si el número no es primo, tener en cuenta que un número primo solo es divisible entre 1 y sí mismo. Hacerlo de forma recursiva.
3. Escribe una función recursiva para sumar los elementos de una lista.
4. Escribir función recursiva que retorne el número de elementos de una lista, es decir que recorra la lista y cuente cuantos valores tiene.
5. Escribe una función recursiva para calcular el enésimo número triangular (es decir, la suma de los primeros n números enteros positivos).

19. Carlos Adrian Manzo Bañol

Problema 1:
Construir una función que reciba un número entero como argumento y retorna la suma de todos los números terminados en 6 comprendidos entre 1 y el número recibido como Argumento

Ejemplos:
Número recibido: 50
Devuelve: (+ 6 16 26 36 46)

Problema 2:
Construir una función que reciba un número entero como argumento y retorna la suma de sus dígitos primo

Ejemplos:
Número recibido: 20
Devuelve: (+ 2 3 5 7 11 13 17 19)

Problema 3:
Construir un programa que cuente la cantidad de números primos que hay entre 1 y un valor n leído

Ejemplo:
Número recibido: 25
Devuelve: 9

Problema 4:
Construir una función que determine si un número es perfecto. Se define un número perfecto como aquel que es igual a la suma de sus divisores exactos sin incluir el mismo número.

Ejemplo:
Número recibido: 6
Devuelve: Número perfecto

Número recibido: 30
Devuelve: Número Imperfecto.

Programa 5:
Realizar un programa que permita visualizar lo siguiente en pantalla:
Ejemplo
N=8

```
AAAAAAA
 AAAAA
  AAA
   A
  AAA
 AAAAA
AAAAAAA
```

20. Juan Esteban Ramírez Arias

Problema 1: Crea un método que imprima por pantalla un triangulo a partir de los valores de la base y la altura.
Ejemplo:
Se ingresa una base de 6, entonces se muestra la siguiente figura

```
     *
    * *
   *   *
  *     *
 * * * * *
```

Programa 2
Realizar un programa que multiplique los números negativos con números positivos de forma recursiva
EJEMPLO:
Suponiendo que n es 8 y b es -4
Su resultado es: -32

Programa 3
Hacer un programa reciba un número:
Si el número es mayor o igual a 100 debe imprimir los números del 12 al 22 con sus sumatorias.
Si el número es mayor que 30 imprimir los número del 2 al 40 con sus cuadrados
Ejemplo:
SUMATORIAS 5 = 5+4+3+2+1

Programa 4
Hacer un programa donde calcule cuántos numeros naturales hay en numeros ingresados con su sumatoria total
Ejemplo:
7 y 16
Existe 9 numeros: 8,9,10,11,12,13,14,15,16
Su sumatoria es: 108

Programa 5
Calcular el Máximo Común Divisor de 3 números
Ejemplo:
MCD: 12, 16 y 20
= mi MCD es 4

22. David Suárez Álvarez
Problema 1: Hacer un programa que lea por teclado una letra diferenciando mayuscula y minuscula y con ella dibuje un reloj de arena acostado de la siguiente forma:

```
A       A
AA     AA
AAA   AAA
AAAAAAA
AAA   AAA
AA     AA
```

A A
Ejemplo 1
Ingresa: B
Muestra:
B B
BB BB
BBB BBB
BBBBBBB
BBB BBB
BB BB
B B
Ejemplo 2
Ingresa: o
Muestra:
o o
oo oo
ooo ooo
ooooooo
ooo ooo
oo oo
o o

Problema 2:

Hacer un programa que dibuje un cuadrado de n x n con los dígitos de la serie de Fibonacci separados por tabulador.

Ejemplo 1
Ingresa: 4
Muestra:
1 1 2 3
5 8 13 21
34 55 89 144
233 377 610 987
Ejemplo 2
Ingresa: 2
Muestra:
1 1
2 3

Problema 3:

Hacer un programa que dibuje la vocal en mayúscula con 5 de altura ingresada por teclado mediante el número de 1 digito que el usuario elija

Ejemplo 1
Ingresa: A 7
Muestra:
7
7 7
77777
7 7
7 7
Ejemplo 2
Ingresa: e 2
Muestra:
2222
2
222
2
2222
Ejemplo 3
Ingresa: Z
Muestra:
Z no es una vocal
Ejemplo 4
Ingresa: I 13
Muestra:
13 tiene más de un dígito

Problema 4:

Hacer un programa que reciba por teclado letras hasta que el usuario ingrese un número, si ingresa una cantidad cuadrada muestra las letras en un cuadrado, si no las ordena alfabéticamente separando mayúsculas y minúsculas.

Ejemplo 1
Ingresa:
AeifQhQQa1
Muestra:
A e i
F Q h
Q Q a
Ejemplo 2
Ingresa: AaQWeBZz3
Muestra:
ABQWZaez

Problema 5:

Hacer un programa que reciba por teclado números hasta que el usuario ingrese una letra, el programa cuenta cuántos primos ingresó y hace una torre invertida con tantos pisos como primos haya ingresado usando el mayor de los primos.

Ejemplo 1
Ingresa:
1 2 3 4 5 6 7 8 b
Muestra:
7777777

77777
777
7
Ejemplo 2
Ingresa:
10 11 7 9 0 111 127 a
Muestra:
127127127127127
127127127
127

23. Daniel Rojas Groihs

Problema 1: Programe un método recursivo que transforme un número entero positivo a notación binaria.
Ejemplo: Al convertir el número 13 a binario, quedaría 1101.
Problema 2: Implemente una función recursiva que nos diga si un número de cualquier cantidad de dígitos es capicúa. Capicúa es un número que se lee igual de izquierda a derecha que de derecha a izquierda.
Ejemplo: El número 1441 es capicúa. El número 123 no es capicúa.
Problema 3: Escribir una función que reciba 2 enteros n y b, devuelva verdadero si n es potencia de b, o falso si no cumple con esto.
Ejemplo: n es 8, b es 2, entonces devuelve verdadero.
Problema 4: Crea un método que compruebe si un número está ordenado de forma decreciente o creciente de cualquier cantidad de dígitos.
Ejemplo: El número 1234 es creciente, el número 4321 es decreciente, el número 5147 no cumple con ninguna característica.
Problema 5: Programe un método recursivo que transforme un número binario a notación octal.
Ejemplo: Al convertir 110011, quedaría 63 en octal.

24. Johan Vargas

Problema 1
Hacer un programa que juegue al juego de cifras de “Cifras y Letras”. El juego consiste en obtener, a partir de 6 números, un número lo más cercano posible a un número de tres cifras realizando operaciones aritméticas con los 6 números.

Problema 2.
Invertir un numero de forma recursiva (no usar String)

Problema 3.
Invertir una palabra de forma recursiva

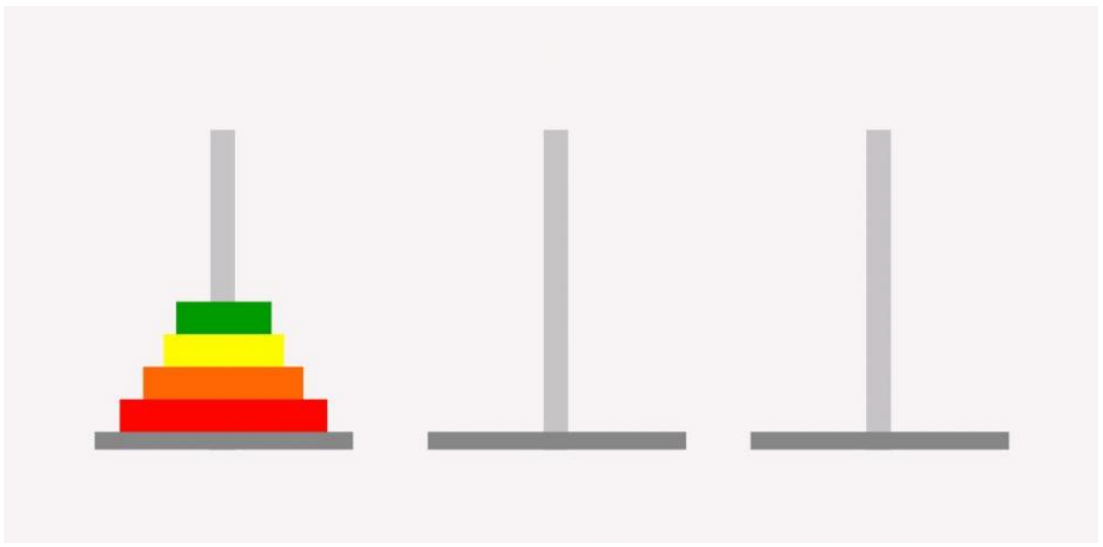
Problema 4.
Crea un método que dado un número, lo imprima invertido por pantalla

Problema 5
Crea un método que imprima por pantalla un Rectángulo a partir de los valores de la base y la altura

1.Ejercicio: TORRES DE HANOI

El juego, en su forma más tradicional, consiste en tres postes verticales. En uno de los postes se apila un número indeterminado de discos perforados por su centro (elaborados de madera), que determinará la complejidad de la solución. Los discos se apilan sobre uno de los postes en tamaño decreciente de abajo arriba. No hay dos discos iguales, y todos ellos están apilados de mayor a menor radio -desde la base del poste hacia arriba- en uno de los postes, quedando los otros dos postes vacíos. El juego consiste en pasar todos los discos desde el poste ocupado (es decir, el que posee la torre) a uno de los otros postes vacíos.
Para realizar este objetivo, es necesario seguir tres simples reglas:

1. **Solo se puede mover un disco cada vez y para mover otro los demás tienen que estar en postes.**
2. **Un disco de mayor tamaño no puede estar sobre uno más pequeño que él mismo.**
3. **Solo se puede desplazar el disco que se encuentre arriba en cada poste.**



La función debe tomar tres argumentos: el número de discos, la torre de origen, la torre auxiliar y la torre destino.
CREAR UN PROGRAMA QUE RECIBA UN X NÚMERO DE DISCOS Y DE LA SOLUCIÓN DICIENDO DE QUE POSTE A QUE POSTE VA HASTA LLEGAR A SU POSICIÓN FINAL.
(hanoi 4 'A 'B 'C);ejemplo

Mover disco de la varilla A a la varilla B
Mover disco de la varilla A a la varilla C

Mover disco de la varilla B a la varilla C
Mover disco de la varilla A a la varilla B
Mover disco de la varilla C a la varilla A
Mover disco de la varilla C a la varilla B
Mover disco de la varilla A a la varilla B
Mover disco de la varilla A a la varilla C
Mover disco de la varilla B a la varilla C
Mover disco de la varilla B a la varilla A
Mover disco de la varilla C a la varilla A
Mover disco de la varilla B a la varilla C
Mover disco de la varilla A a la varilla B
Mover disco de la varilla A a la varilla C
Mover disco de la varilla B a la varilla C