Resumen de Ejercicios

Nivel 1 – (manejo de cadenas, bucles y operaciones sencillas)

1. **first_word**

   Objetivo: Imprimir la primera palabra de una cadena.

   Conceptos: Recorrer una cadena, identificar espacios, gestionar índices.

2. **fizzbuzz**

   Objetivo: Imprimir números del 1 al N, sustituyendo los múltiplos de 3 por “fizz”, de 5 por “buzz” y de ambos por “fizzbuzz”.

   Conceptos: Bucles, condicionales y módulo.

3. **ft_strcpy**

   Objetivo: Copiar el contenido de una cadena fuente a otra (sin usar la función estándar).

   Conceptos: Manejo de arrays, punteros, terminación con `'\0'`.

4. **ft_strlen**

   Objetivo: Calcular la longitud de una cadena (número de caracteres hasta `'\0'`).

   Conceptos: Recorrido de arrays y contadores.

5. **ft_swap**

   Objetivo: Intercambiar los valores de dos variables enteras.

   Conceptos: Punteros y paso por referencia.

6. **putstr**

   Objetivo: Imprimir una cadena por la salida estándar.

   Conceptos: Funciones de escritura (por ejemplo, `write` en Unix), bucles.

7. **repeat_alpha**

   Objetivo: Imprimir cada carácter de la cadena, repitiendo las letras un número de veces igual a su posición en el alfabeto (por ejemplo, ‘a’ se imprime 1 vez, ‘b’ 2 veces, etc.).

Conceptos: Conversión de caracteres, aritmética con códigos ASCII y bucles.

8. **rev_print**

   Objetivo: Imprimir la cadena en orden inverso.

   Conceptos: Recorrido inverso de arrays.

9. **rot_13**

   Objetivo: Aplicar el cifrado ROT13 a una cadena (rotar cada letra 13 posiciones en el alfabeto).

   Conceptos: Manejo de mayúsculas y minúsculas, aritmética modular.

10. **rotone**

    Objetivo: Rotar cada carácter al siguiente (por ejemplo, 'a' se convierte en 'b', y 'z' en 'a').

    Conceptos: Similar a rot_13 pero con desplazamiento 1, manejo de límites en el alfabeto.

11. **search_and_replace**

    Objetivo: Buscar un carácter en una cadena y reemplazarlo por otro, imprimiendo el resultado.

    Conceptos: Recorrido de cadenas, condicionales y sustitución.

12. **ulstr**

    Objetivo: Convertir las letras minúsculas en mayúsculas y viceversa en una cadena.

    Conceptos: Manipulación de caracteres, funciones de conversión (sin usar funciones estándar como `toupper` / `tolower` si es requerido).

Nivel 2 – (operaciones con cadenas, conversión de formatos y manejo básico de memoria)

1. **alpha_mirror**

   Objetivo: Para cada letra, imprimir su "espejo" en el alfabeto (por ejemplo, 'a' ↔ 'z', 'b' ↔ 'y', etc.).

   Conceptos: Mapear caracteres usando aritmética con ASCII.

2. **camel_to_snake**

Objetivo: Convertir una cadena en formato *camelCase* a *snake_case*.

Conceptos: Recorrer la cadena, detectar letras mayúsculas e insertar un guión bajo antes, luego pasar a minúscula.

3. **do_op**

   Objetivo: Realizar una operación aritmética simple (suma, resta, multiplicación, división, módulo) a partir de argumentos dados.

   Conceptos: Conversión de cadena a entero, condicionales y validación de operaciones.

4. **ft_atoi**

   Objetivo: Convertir una cadena numérica en un entero (implementación propia de `atoi`).

   Conceptos: Manejo de espacios, signos y dígitos.

5. **ft_strcmp**

   Objetivo: Comparar dos cadenas (similar a la función estándar `strcmp`).

   Conceptos: Recorrido paralelo de dos cadenas y diferencia de códigos ASCII.

6. **ft_strcspn**

   Objetivo: Contar la cantidad de caracteres iniciales en una cadena que no contengan ningún carácter de un “set” dado (imitando la función de la libc).

   Conceptos: Bucles anidados, búsqueda de caracteres.

7. **ft_strdup**

   Objetivo: Duplicar una cadena, asignando memoria dinámica para la copia.

   Conceptos: Uso de `malloc`, recuento de caracteres y copia.

8. **ft_strpbrk**

   Objetivo: Buscar la primera aparición en una cadena de cualquiera de los caracteres de un conjunto dado (similar a la función estándar).

   Conceptos: Recorridos anidados y búsqueda.

9. **ft_strrev**

   Objetivo: Invertir una cadena (retornar o imprimir la cadena en orden inverso).

Conceptos: Recorrido y manejo de arrays.

10. **ft_strspn**

    Objetivo: Contar la cantidad inicial de caracteres de una cadena que se encuentran en un “set” dado.

    Conceptos: Recorrido y verificación de pertenencia.

11. **inter**

    Objetivo: Imprimir, sin repetir, los caracteres comunes a dos cadenas, en el orden en que aparecen en la primera.

    Conceptos: Comparación de cadenas, eliminación de duplicados.

12. **is_power_of_2**

    Objetivo: Determinar si un número es potencia de 2.

    Conceptos: Operaciones bit a bit y divisiones sucesivas.

13. **last_word**

    Objetivo: Imprimir la última palabra de una cadena (ignorando espacios extra).

    Conceptos: Recorrer la cadena desde el final, identificar límites de palabras.

14. **max**

    Objetivo: Devolver el valor máximo de un array de enteros.

    Conceptos: Recorrido de arrays y comparación.

15. **print_bits**

    Objetivo: Imprimir la representación binaria (bits) de un número.

    Conceptos: Operaciones bit a bit, máscaras y bucles.

16. **reverse_bits**

    Objetivo: Invertir el orden de los bits de un número (por ejemplo, de 8 bits, el bit de posición 0 pasa a la 7, etc.).

    Conceptos: Operaciones bit a bit, ciclos de longitud fija.

17. **snake_to_camel**

    Objetivo: Convertir una cadena en *snake_case* a *camelCase*.

Conceptos: Identificar guiones bajos y convertir la letra siguiente a mayúscula.

18. **swap_bits**

    Objetivo: Intercambiar partes de los bits de un byte (por ejemplo, cambiar el nibble alto por el bajo).

    Conceptos: Operaciones bit a bit, máscaras y desplazamientos.

19. **union**

    Objetivo: Imprimir la unión de los caracteres de dos cadenas, sin repeticiones y en el orden en que aparecen.

    Conceptos: Comparación de caracteres, evitar duplicados y manejo de arrays.

20. **wdmatch**

    Objetivo: Comprobar si todos los caracteres de la primera palabra aparecen, en orden, en la segunda.

    Conceptos: Recorrido de cadenas y verificación secuencial.

Nivel 3 – (algoritmos, manejo de listas, conversión en bases y operaciones matemáticas)

1. **add_prime_sum**

   Objetivo: Sumar todos los números primos menores o iguales a un número dado.

   Conceptos: Comprobación de primos (bucle y divisibilidad) y acumulación.

2. **epur_str**

   Objetivo: Eliminar espacios extra en una cadena, dejando solo un espacio entre palabras.

   Conceptos: Recorrido de la cadena, detección y manejo de espacios.

3. **expand_str**

   Objetivo: “Expandir” una cadena según un patrón; por ejemplo, repetir caracteres basándose en un dígito precedente.

   Conceptos: Conversión de caracteres a números, bucles anidados y manejo de la salida.

4. **ft_atoi_base**

   Objetivo: Convertir una cadena a entero, considerando una base numérica (por ejemplo, binario, hexadecimal, etc.).

   Conceptos: Validación de la base, manejo de dígitos y letras, conversión aritmética.

5. **ft_list_size**

   Objetivo: Calcular el número de elementos en una lista enlazada.

   Conceptos: Estructuras de datos (listas enlazadas) y bucles.

6. **ft_range**

   Objetivo: Crear un array de enteros que contenga todos los valores entre dos números dados (mínimo y máximo).

   Conceptos: Manejo de arrays y memoria dinámica.

7. **ft_rrange**

   Objetivo: Similar a ft_range, pero el array se llena en orden inverso.

   Conceptos: Igual que ft_range con lógica de decremento.

8. **hidenp**

   Objetivo: Verificar si todos los caracteres de la primera cadena aparecen en la segunda, en orden (no necesariamente de forma contigua).

   Conceptos: Recorrido secuencial, similar a wdmatch pero con posibles matices.

9. **lcm**

   Objetivo: Calcular el mínimo común múltiplo (LCM) de dos números.

   Conceptos: Algoritmos matemáticos, uso del GCD (pgcd) y aritmética.

10. **paramsum**

    Objetivo: Sumar los números pasados como argumentos al programa.

    Conceptos: Manejo de argumentos en `main`, conversión de cadena a entero y acumuladores.

11. **pgcd**

    Objetivo: Calcular el máximo común divisor (GCD) de dos números (por ejemplo, usando el algoritmo de Euclides).

Conceptos: Recursión o bucles y operaciones aritméticas.

12. **print_hex**

    Objetivo: Imprimir un número en su representación hexadecimal.

    Conceptos: Conversión de números, división y manejo de caracteres (0-9, a-f).

13. **rstr_capitalizer**

    Objetivo: Capitalizar (poner en mayúscula) la última letra de cada palabra y poner en minúscula el resto.

    Conceptos: Manejo de cadenas, detección de límites de palabra y conversión de caracteres.

14. **str_capitalizer**

    Objetivo: Capitalizar la primera letra de cada palabra y poner en minúscula el resto (el clásico “title case”).

    Conceptos: Similar al anterior pero con enfoque en el primer carácter.

15. **tab_mult**

    Objetivo: Imprimir la tabla de multiplicar de un número dado.

    Conceptos: Bucles, formato de salida y operaciones aritméticas.

Nivel 4 – (uso de estructuras de datos, manipulación dinámica, algoritmos de ordenación y transformación de cadenas)

1. **flood_fill**

   Objetivo: Implementar un algoritmo de “relleno” (como el de la herramienta “bucket” en pintura) en una matriz (2D array).

   Conceptos: Recursión o uso de pila/cola, manejo de matrices, comprobación de límites.

2. **fprime**

   Objetivo: Mostrar todos los factores primos de un número.

   Conceptos: Comprobación de primos, bucles y lógica de factorización.

3. **ft_itoa**

   Objetivo: Convertir un entero en una cadena de caracteres.

Conceptos: Manejo de memoria dinámica, cálculo del número de dígitos y manejo de números negativos.

4. **ft_list_foreach**

   Objetivo: Aplicar una función a cada elemento de una lista enlazada.

   Conceptos: Recorrer estructuras enlazadas y función de callback.

5. **ft_list_remove_if**

   Objetivo: Eliminar de una lista enlazada los nodos que cumplan una condición dada.

   Conceptos: Manipulación de listas, gestión de memoria y comparaciones.

6. **ft_split**

   Objetivo: Dividir una cadena en un array de cadenas usando un delimitador (por lo general, espacios u otros caracteres).

   Conceptos: Manejo de cadenas, asignación dinámica y estructuras de arrays.

7. **rev_wstr**

   Objetivo: Imprimir las palabras de una cadena en orden inverso, manteniendo el orden interno de cada palabra.

   Conceptos: Separación de palabras, manejo de arrays y recuento de palabras.

8. **rostring**

   Objetivo: Mover la primera palabra al final de la cadena y ajustar los espacios.

   Conceptos: Procesamiento de cadenas, detección de palabras y reensamblado de la cadena.

9. **sort_int_tab**

   Objetivo: Ordenar un array de enteros (por ejemplo, usando burbuja o cualquier otro método sencillo).

   Conceptos: Algoritmos de ordenación y manejo de arrays.

10. **sort_list**

    Objetivo: Ordenar una lista enlazada (por ejemplo, en orden alfabético o numérico, según se especifique).

Conceptos: Algoritmos de ordenación aplicados a estructuras enlazadas y manejo de punteros.