Algoritmos y Programación I (95.11) – Curso Kuhn – 2do parcialito, 1er recuperatorio – 24/06/2019

Resolver los siguientes problemas en forma clara y legible en código ISO-C99.

1. Implementar la función `char *join(char *strs[], size_t n, char delim);` que reciba en `strs` un arreglo de `n` cadenas de caracteres y devuelva una cadena que contenga la concatenación de todas ellas usando el carácter `delim` como delimitador.

   Por ejemplo, si se llama a `join({"hola", "que", "tal"}, 3, ' ')` debe devolverse `"hola que tal"`.

2. a. Implementar una función `char **leer_lineas(size_t *n);` que lea líneas de `stdin` hasta que se termine la entrada estándar y que devuelva un arreglo de cadenas que contenga cada una de esas líneas. Debe devolverse la cantidad de líneas leídas a través de `n`.

   b. Implementar la función `void liberar_lineas(char **lineas, size_t n);` que reciba un arreglo dinámico de cadenas `lineas` de longitud `n` y libere la memoria asociada.

3. Se quiere modelar el contacto en una agenda telefónica.

   Un número de teléfono consta de una *referencia* que indica qué representa ese número y que es un texto de no más de `MAX_CADENA` elementos y de un entero largo sin signo que representa al *número*.

   Un contacto en una agenda tiene el *nombre* del contacto, una cadena de no más de `MAX_CADENA` elementos y de un vector de n (nunca más de `MAX_NUMEROS`) *números* telefónicos.

   a. Declarar las estructuras `struct numero` y `struct contacto` que representen un número telefónico y un contacto respectivamete.

   b. Definir los tipos `numero_t` y `contacto_t` en base a las estructuras anteriores.

   c. Escribir una función

      ```
      bool numero_en_contacto(const contacto_t *c, unsigned long numero);
      ```

      que retorne si `numero` es uno de los números del contacto `c`.

¡Suerte! :)

Algoritmos y Programación I (95.11) – Curso Kuhn – 2do parcialito, 1er recuperatorio – 24/06/2019

Resolver los siguientes problemas en forma clara y legible en código ISO-C99.

1. Implementar la función `char *join(char *strs[], size_t n, char delim);` que reciba en `strs` un arreglo de `n` cadenas de caracteres y devuelva una cadena que contenga la concatenación de todas ellas usando el carácter `delim` como delimitador.

   Por ejemplo, si se llama a `join({"hola", "que", "tal"}, 3, ' ')` debe devolverse `"hola que tal"`.

2. a. Implementar una función `char **leer_lineas(size_t *n);` que lea líneas de `stdin` hasta que se termine la entrada estándar y que devuelva un arreglo de cadenas que contenga cada una de esas líneas. Debe devolverse la cantidad de líneas leídas a través de `n`.

   b. Implementar la función `void liberar_lineas(char **lineas, size_t n);` que reciba un arreglo dinámico de cadenas `lineas` de longitud `n` y libere la memoria asociada.

3. Se quiere modelar el contacto en una agenda telefónica.

   Un número de teléfono consta de una *referencia* que indica qué representa ese número y que es un texto de no más de `MAX_CADENA` elementos y de un entero largo sin signo que representa al *número*.

   Un contacto en una agenda tiene el *nombre* del contacto, una cadena de no más de `MAX_CADENA` elementos y de un vector de n (nunca más de `MAX_NUMEROS`) *números* telefónicos.

   a. Declarar las estructuras `struct numero` y `struct contacto` que representen un número telefónico y un contacto respectivamete.

   b. Definir los tipos `numero_t` y `contacto_t` en base a las estructuras anteriores.

   c. Escribir una función

      ```
      bool numero_en_contacto(const contacto_t *c, unsigned long numero);
      ```

      que retorne si `numero` es uno de los números del contacto `c`.

¡Suerte! :)