UNLaM
Dto. Ingeniería e Investigaciones Tecnológicas

E. A. Nro. 1 – Primera Evaluación de Aprendizaje.

Debe resolver la siguiente problemática eligiendo la mejor estrategia.

1.- Determinar la cantidad de ocurrencias de una sub cadena en una cadena. Hay dos modos. Encontrar ANA en anana se puede decir que está una vez o dos veces (según quién lo mire), así que hay que darles el gusto a ambas formas de verlo.

- En este ejercicio se espera que encare una estrategia inicial (si lo prefiere, haciendo uso de funciones de biblioteca). Pero que llegue, por una cuestión de resolver la algoritmia y demostrando que usted lo puede hacer, a una solución en que en una sola función resuelva cada problema. Si en su estrategia inicial usó alguna función de biblioteca, piense si la puede resolver, p. ej., con un macro reemplazo. Ambas soluciones (con repetición y sin repetición) se pueden resolver en unas 50 líneas de código.

- El otro punto a considerar es que bajo ningún concepto use subíndices ni puntero + desplazamiento (p. ej.: `*(punt + desp)`), sólo aritmética de punteros.

2.- Dado un array bidimensional de enteros, rotarlo 180 grados (¡y no pregunte si a la izquierda o a la derecha!), y luego debe mostrar los elementos que pertenecen a la diagonal secundaria además de los que están por debajo de la misma. Elija una buena estrategia para hacer ambas cosas, porque debe funcionar no solo con matrices cuadradas (la forma de mostrar es la que usted verá ejecutando el programa).

Puede probar el proyecto con una matriz rectangular.

- Note que dado que los enteros de la matriz son de a lo sumo tres dígitos se los mostró de modo de tener una salida ordenada, prolija, visualizable.

- Esperamos que elija la mejor estrategia para "*rotar*" la matriz. Distribución del tiempo: 90 % estrategia, 2 % solución, 8 % prueba.

- Esperamos que elija la mejor solución para mostrar la "triangular secundaria inferior". Sólo hay que *mirar* los elementos de la matriz a mostrar, pero antes mostrar (de una sola vez) los n espacios en blanco que correspondan.

3.- Se tiene un archivo de texto y valiéndose de dos pilas, hay que ordenarlo y generar un nuevo archivo ordenado. Si no se pudiera generar el archivo de salida, se debe mostrar por pantalla. Esta función ordenará de menor a mayor bajo el criterio de que primero se compara por cantidad de palabras, si la cantidad de palabras coincide se tiene en cuenta la longitud de la palabra más larga.

Puede simular que no hay archivo a ordenar. También puede simular que no se puede crear el archivo ordenado (con lo que muestra por pantalla en lugar de crear el archivo).

Las primitivas del TDA PILA involucradas son las de implementación estática de memoria.

- En cuanto al último ítem (implementación estática) se espera la misma estrategia de implementación del material subido en contenidos, el cual tiene el código fuente de la implementación de todas las primitivas.

- la implementación de la estrategia de ordenamiento con dos pilas ya fue planteada en algún ejercicio de la guía de T.P., con lo que queremos ver si ha incorporado esta estrategia.

- también queremos evaluar su estrategia de solución de la función de comparación. Si no lo hace inicialmente (pudo ser su solución inicial a perfeccionar), es esperable que no genere cadenas de caracteres auxiliares. Y que al igual que con el primer ejercicio, no utilice subíndices para contar palabras ni para determinar cuál es la más larga. ¡Haga una implementación que supere sus propias expectativas y las nuestras!

- Note que lo que debe entregar: "**`nnnnnnnn_APELLIDO_Nombre_EA_1.zip`**", en un archivo comprimido: donde '**`nnnnnnnn`**' es su D. N. I., '**`APELLIDO_Nombre`**' son su apellido

UNLaM
Dto. Ingeniería e Investigaciones Tecnológicas

completo y nombre completo. Ese archivo comprimido debe contener los archivos "**`main.c`**", "**`main.h`**", "**`funciones.c`**", "**`funciones.h`**", "**`lib-EA_1_R_bib.a`**", "**`EA-1.cbp`**" y "**`EA-1.layout`**".

Para su información: lo único que usaremos para comprobar que todo anda como debe es el archivo "**`funciones.c`**", que probaremos en nuestro proyecto dónde tendremos habilitadas las funciones terminadas en "**`_MIO`**" que usted debe resolver.

Cómo envía su solución: por medio de [MIeL] en [Tutoría] / [Resumen de prácticas].

Recibirá corrección y las observaciones, si las hubiera, por el mismo medio.

Los archivos del proyecto quedarán como documentación.
- "main.c" que utiliza las versiones de sus funciones (por ejemplo, "**`ordenarArchivo_MIO`**").
- "**`funciones.c`**".en que define sus funciones (además de cualquier otra función o macroreemplazo que le resulte necesario)
- "**`funciones.h`**" donde habilitó los prototipos de las funciones y prototipos terminadas en "**`_mio`**".
- "**`EA-1.cbp`**" archive del proyecto.
- "**`EA-1.layout`**" diseño del proyecto.
- "**`lib-EA_1_R_bib.a`**" biblioteca orecompilada.