Algoritmos y Programación I (95.11) – Curso Kuhn – 4to parcialito – 10/06/2019

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

1. Teniendo el TDA Conjunto de enteros el cual se representa sobre `typedef struct {int *e; size_t n;} conjunto_t;` en donde `e` es un vector **ordenado** de longitud `n` con los elementos contenidos en el conjunto se pide:

   a. Implementar la primitiva

   ```
   conjunto_t *conjunto_interseccion(const conjunto_t *a, const conjunto_t *b);
   ```

   que compute la intersección entre dos conjuntos. (Asumir que se tiene la primitiva `conjunto_t *conjunto_crear();` y la función auxiliar `static bool _agregar_al_final(conjunto_t *c, int e);` que agrega un elemento al final sin chequear si esa es su ubicación correcta. **Para simplificar:** No hace falta chequear fallas de memoria.)

   Aclaración: La función debe resolver el problema eficientemente sin llamar a funciones de ordenamiento ni implementar ningún método de ordenamiento.

   b. Si implementáramos la primitiva

   ```
   bool conjunto_agregar(conjunto_t *c, int e);
   ```

   con una llamada a `_agregar_al_final(c, e)`, ¿con qué método nos convendría ordenar el arreglo después?, justificar brevemente.

2. Teniendo una lista **de enteros** en la cual la lista se representa `typedef struct {struct nodo *prim;} lista_t;` y el nodo se representa como `struct nodo {struct nodo *sig; int dato;};` implementar la primitiva

   ```
   void lista_enteros_borrar(lista_t *l, int e);
   ```

   que borre todas las ocurrencias del dato `e` en la lista.

3. Se representa a un color RGB de 24 bits como un entero sin signo `0xAABBCC` donde el byte `0xAA` representa el rojo, `0xBB` el verde y `0xCC` el azul. Se pide:

   a. Implementar la función `unsigned char get_rojo(unsigned int color);` que devueva el valor de la componente roja.

   b. Implementar la función `void set_verde(unsigned int *color, unsigned char verde);` que cargue la componente de `verde` en el `color` recibido.

¡Suerte! :)

Algoritmos y Programación I (95.11) – Curso Kuhn – 4to parcialito – 10/06/2019

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

1. Teniendo el TDA Conjunto de enteros el cual se representa sobre `typedef struct {int *e; size_t n;} conjunto_t;` en donde `e` es un vector **ordenado** de longitud `n` con los elementos contenidos en el conjunto se pide:

   a. Implementar la primitiva

   ```
   conjunto_t *conjunto_interseccion(const conjunto_t *a, const conjunto_t *b);
   ```

   que compute la intersección entre dos conjuntos. (Asumir que se tiene la primitiva `conjunto_t *conjunto_crear();` y la función auxiliar `static bool _agregar_al_final(conjunto_t *c, int e);` que agrega un elemento al final sin chequear si esa es su ubicación correcta. **Para simplificar:** No hace falta chequear fallas de memoria.)

   Aclaración: La función debe resolver el problema eficientemente sin llamar a funciones de ordenamiento ni implementar ningún método de ordenamiento.

   b. Si implementáramos la primitiva

   ```
   bool conjunto_agregar(conjunto_t *c, int e);
   ```

   con una llamada a `_agregar_al_final(c, e)`, ¿con qué método nos convendría ordenar el arreglo después?, justificar brevemente.

2. Teniendo una lista **de enteros** en la cual la lista se representa `typedef struct {struct nodo *prim;} lista_t;` y el nodo se representa como `struct nodo {struct nodo *sig; int dato;};` implementar la primitiva

   ```
   void lista_enteros_borrar(lista_t *l, int e);
   ```

   que borre todas las ocurrencias del dato `e` en la lista.

3. Se representa a un color RGB de 24 bits como un entero sin signo `0xAABBCC` donde el byte `0xAA` representa el rojo, `0xBB` el verde y `0xCC` el azul. Se pide:

   a. Implementar la función `unsigned char get_rojo(unsigned int color);` que devueva el valor de la componente roja.

   b. Implementar la función `void set_verde(unsigned int *color, unsigned char verde);` que cargue la componente de `verde` en el `color` recibido.

¡Suerte! :)