Ejercicios con punteros y memoria dinámica

1. Completa el código de la siguiente función para que intercambie los valores de las variables apuntadas por `a` y `b`.

   ```
   void intercambia( int * a, int * b )
   ```

2. Completa el código de la siguiente función para que devuelva la dirección en memoria dinámica de un nuevo array con los `n` valores contenidos en el array `datos`.

   ```
   int * copia( int datos[], int n )
   ```

3. Completa el código de la siguiente función para que devuelva la dirección en memoria dinámica de un nuevo array con `n` enteros generados aleatoriamente en el intervalo cerrado `[a,b]`.

   ```
   int * genera( int n, int a, int b )
   ```

4. Completa el código de la siguiente función para que devuelva la dirección en memoria dinámica de un nuevo array con los `n` valores contenidos en el array `datos` en el orden inverso.

   ```
   int * invierte( int datos[], int n )
   ```

5. Completa el código de la siguiente función para que devuelva la dirección en memoria dinámica de un nuevo array cuyos elementos sean el resultado de sumar los `n` valores contenidos en los arrays `a` y `b`.

   ```
   int * suma( int a[], int b[], int n )
   ```

6. Completa el código de la siguiente función para que devuelva la dirección en memoria dinámica de una nueva cadena de caracteres idéntica a la recibida como parámetro.

   ```
   char * duplica( char * cadena )
   ```

7. Completa el código de la siguiente función para que devuelva la dirección en memoria dinámica de una nueva cadena de caracteres con la inversión de la recibida como parámetro.

   ```
   char * invierte( char * cadena )
   ```

8. Completa el código de la siguiente función para que devuelva la dirección en memoria dinámica de una nueva cadena de caracteres que contenga `n` letras elegidas aleatoriamente de entre las incluidos en la cadena `letras`.

   ```
   char * genera( char * letras, int n )
   ```

9. Completa el código de la siguiente función para que devuelva la dirección en memoria dinámica de una nueva cadena de caracteres que contenga la cadena formada por la concatenación de las cadenas `a` y `b`. Por ejemplo, si a="hola" y b="mundo" el resultado será la cadena "holamundo".

   ```
   char * concatena( char * a, char * b )
   ```

10. Completa el código de la siguiente función para que devuelva la dirección en memoria dinámica de un nuevo array con `n` puntos cuyas coordenadas estén generadas aleatoriamente.

```
struct PuntoRep {
  int x;
  int y;
};

struct PuntoRep * genera( int n )
```

11. Completa el código de la siguiente función para que devuelva la dirección en memoria dinámica de un nuevo array con todos los valores positivos que haya en el array `datos`, sabiendo que éste contiene `n` números enteros. Coloca un -1 como último elemento del array devuelto a modo de marca de fin.

```
int * positivos( int datos[], int n )
```

12. Completa el código de la siguiente función para que devuelva la dirección en memoria dinámica de un nuevo array con `n+1` punteros a carácter. Cada uno apuntará a una nueva cadena de caracteres almacenada en memoria dinámica. El contenido de las primeras `n` cadena será cada una de las `n` direcciones de correo electrónico encontradas en la cadena recibida como parámetro. La última cadena será la cadena de caracteres vacía. Las direcciones de correo electrónico aparecerán separadas por el carácter ';' dentro de la cadena `direcciones`.

```
char ** divide( char * direcciones )
```

13. Completa el código de la siguiente función para que devuelva la dirección en memoria dinámica de un nuevo array con todos los valores positivos que haya en el array `datos`, sabiendo que éste contiene `n` números enteros. Pero esta vez no deben aparecer números repetidos. Coloca un -1 como último elemento del array devuelto a modo de marca de fin. Por ejemplo, si datos = {1,2,3,1,2} hay que devolver {1,2,3, -1}.

```
int * positivos( int datos[], int n )
```

14. Completa el código de la siguiente función para que devuelva la dirección en memoria dinámica de un nuevo array con las coordenadas correspondientes a las celdas del array bidimensional `datos`, de `f` filas y `c` columnas, en las que haya un valor negativo. El número de valores negativos encontrados, que se corresponderá con el tamaño del nuevo array generado en memoria dinámica, se devolverá a través del parámetro `n`.

```
struct CoordenadaRep {
  int x;
  int y;
};

struct CoordenadaRep * encuentra( int f, int c, int datos[f][c], int * n )
```

15. Completa el código de la siguiente función para que devuelva la dirección en memoria dinámica de una matriz (array de dos dimensiones) con `n` filas y `n` columnas conteniendo la diferencia en valor absoluto entre cada pareja de elementos de los arrays `a` y `b`., sabiendo que ambos contienen `n` elementos.

```
int ** genera( int a[], int b[], int n )
```