Proyecto 1
primera parte

Algoritmos y Estructuras de Datos II
Laboratorio

10 de marzo de 2010

Hacer los siguientes funciones separadas junto con una interface para el usuario que maneje la entrada/salida.

1. Programar una función

   ```
   void swap(int *p, int *q)
   ```

   que intecambie los valores apuntados por `p` y `q`.

2. Hacer una función para obtener un puntero al máximo de tres variables ingresadas por el usuario. El ingreso de las variables debe hacerse desde la interface (función `main`) a variables de tipos `int`. A partir de este puntero devuelto por la función mostrar despues el resultado desde esta misma interface. Al compilar el programa utilice la librería *electric fence* (utilizar el parámetro `-lefence` de gcc[1]).

   Pregunta: ¿Se puede programar la función con el siguiente prototipo?

   ```
   int * mayorp(int a, int b, int c);
   ```

   tal que `mayorp(a,b,c)` me devuelva un puntero que apunte al mayor valor almacenado en `a`, `b`, `c`?

   Si se puede prográmela. Si no se puede explique porque.

3. Hacer un programa que pida un entero mayor que cero al usuario, cree un arreglo (de enteros) dinámico de esa longitud, lo llene con números al azar [2] y llame a la función:

[1] Leer man page de efence.
[2] Leer la man page de la función de librería rand.

```
int * mayorap(int a[], int n)
```

(`n` en la longitud del arreglo `a`)

que devuelva un puntero al máximo elemento del arreglo.

Verificar con el programa *valgrind* que toda la memoria fue liberada al finalizar el programa.

Pregunta: ¿El valor que devuelve la función denota un arreglo? ¿Qué arreglo?

Una vez finalizada la llamada a esta función, liberar la memoria asignada.

4. Hacer un programa que pida un entero `n` mayor que cero al usuario, cree un arreglo dinámico `a` de **punteros a enteros** de longitud `n`, pida otro entero `m` al usuario y asigne

$$\texttt{a[i]} = \begin{cases} \texttt{\&m}, \text{ si } \texttt{i+1} \text{ divide a } \texttt{m} \\ \texttt{NULL}, \text{ si } \texttt{i+1} \text{ no divide a } \texttt{m} \end{cases}$$

en cada posición `i` del arreglo, con `0 <= i < n`.

Recuerde liberar la memoria al finalizar.

Verificar con el programa *valgrind* que toda la memoria fue liberada al finalizar el programa.