1. Describir la salida de los siguientes programas:

a)

```
#include <stdio.h>

int main (){
  int a = 5, *p;

  a = *p * a;
  if (a == *p)
    printf("a es igual a *p\n");
  else
    printf("a es diferente a *p\n");
}
```

b)

```
#include <stdio.h>

int main (){
  int a = 5, *p;

  *p = *p * a;
  if (a == *p)
    printf("a es igual a *p\n");
  else
    printf("a es diferente a *p\n");
}
```

c)

```
#include <stdio.h>

int main (){
  int a = 5, *p = &a;

  *p = *p * a;
  if (a == *p)
    printf("a es igual a *p\n");
  else
    printf("a es diferente a *p\n");
}
```

d)

```
#include <stdio.h>

int main (){
  int a = 5, *p = &a, **p2 = &p;

  **p2 = *p + (**p2 / a);
  *p = a+1;
  a = **p2 / 2;
  printf("a es igual a:%d\n", a);
}
```

e)

```
#include <stdio.h>
#include <stdlib.h>

int main (){
   int *p1, *p2;

   p1 = (int*) malloc(sizeof(int));
   *p1 = 42;
   p2 = p1;
   printf("%d y %d\n", *p1, *p2);

   *p2 = 53;
   printf("%d y %d\n", *p1, *p2);

   p1 = (int*) malloc(sizeof(int));
   *p1 = 88;
   printf("%d y %d\n", *p1, *p2);
}
```

2. Dadas las siguientes declaraciones

```
struct electrica {
   char corriente[30];
   int voltios;
};

struct electrica *p = (struct electrica*) malloc(sizeof(struct electrica));
struct electrica *q = (struct electrica*) malloc(sizeof(struct electrica));
```

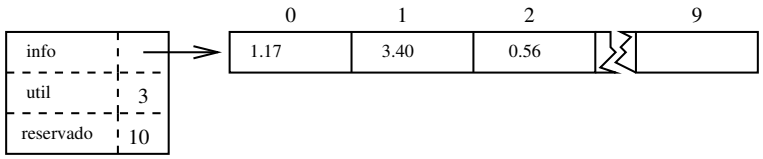
se pide averiguar qué hacen cada una de las siguientes sentencias. Hay alguna inválida?

```
a.  strcpy(p->corriente,"ALTERNA");   e.  strcpy(p->corriente,"ALTA");
b.  p->voltios = q->voltios;          f.  p->corriente = q->voltios;
c.  *p = *q;                          g.  p = 54;
d.  p = q;                            h.  *q = p;
```

3. Supongamos las siguientes definiciones de tipo de dato:

```
struct vectorSD {
   double *info;
   int util;
   int reservado;
};
```

donde `info` es un puntero que mantiene la dirección de una secuencia de reales, `util` indica el número de componentes de la secuencia y `reservado` indica el número de posiciones reservadas de la memoria dinámica para almacenar la secuencia de reales. La siguiente figura muestra un ejemplo de este tipo de representación.



a) Construir una función que inicialice una variable de tipo `struct vectorSD` reservando **n** casillas de memoria dinámica y ponga el número de componentes usadas a 0.

b) Construir una función que añada un elemento en una variable `struct vectorSD`. Considerar el caso de que la inclusión del nuevo valor sobrepase la reserva de memoria. En este caso, realojar el vector reservando el doble de posiciones.

c) Construir una función que realice una copia de una variable `struct vectorSD` en otra variable del mismo tipo. La copia debe reservar memoria para almacenar sólo las componentes usadas del vector.

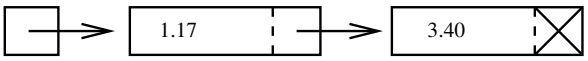
d) Construir una función que libere la memoria reservada por una variable de tipo `struct vectorSD`.

4. Dada la siguiente definición de tipo de dato:

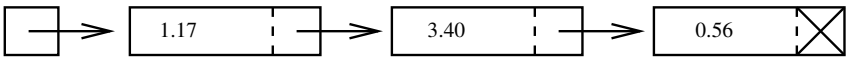
```
  struct celdaSD{
     double info;
     struct celdaSD *sig;
};
```

donde `info` es una variable `double` y `sig` es un puntero que apunta a una variable `struct celdaSD`.

a) Construir una función que permita añadir al final de una secuencia de celdas enlazadas una nueva celda. Por ejemplo, dada la siguiente estructura de celdas enlazadas



el resultado de añadir el valor `0.56` a la estructura es:

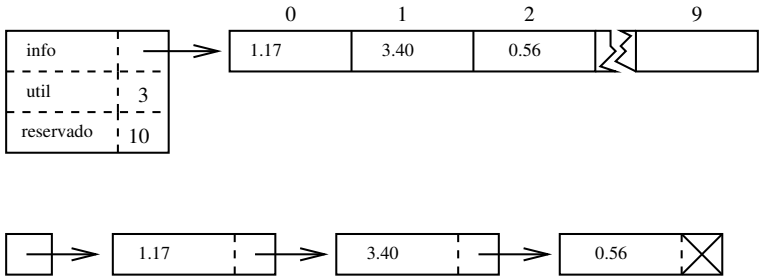


NOTA: El puntero de la última celda contiene un 0.

b) Construir una función que permita eliminar la última celda de una estructura de celdas encadenadas de tipo `struct celdaSD`.

c) Construir una función que elimine y libere toda la información contenida en una estructura de celdas enlazadas.

d) Construir una función que permita pasar una variable de tipo `struct vectorSD` (definida en el ejercicio anterior) a una estructura de celdas enlazadas de tipo `struct celdaSD`. Por ejemplo:



e) Construir una función que realice la operación inversa del ejercicio anterior, es decir, pasar un estructura de celdas enlazadas a una variable de tipo `struct vectorSD`.

f) Construir una función que inserte una nueva celda en la estructura anterior, detrás de una determinada celda (como parámetro de la función se indica la dirección de memoria de dicha celda).

f1) ¿Podría utilizar la función anterior para insertar una celda al principio de la estructura de celdas enlazadas? Si no es así, modificarlo para que sea posible.

f2) Implementar otra función que inserte la nueva celda delante de la celda indicada

g) Construir una función que dada la dirección de memoria de una celda, la elimine de la estructura de celdas enlazadas.

g1) ¿Sería posible utilizar la función anterior para eliminar la primera de las celdas de la estructura de celdas enlazadas? Si no es así, modificarlo para que sea posible.

h) Implementar el método de ordenación de la burbuja para ordenar una estructura de celdas enlazadas.

i) Suponiendo que partimos de una estructura de celdas enlazadas ordenadas ascendentemente, implementar una función que inserte ordenadamente una nueva celda en la estructura.

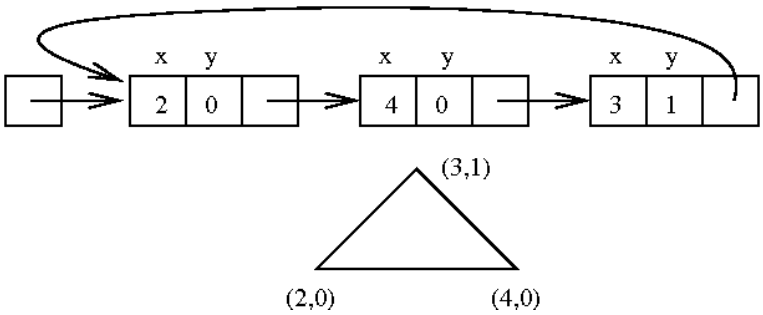
5. Se desea desarrollar una estructura de datos que permita representar de forma general diversas figuras poligonales. Cada figura poligonal se puede representar como un conjunto de puntos en el plano unidos por segmentos de rectas entre cada dos puntos adyacentes. Por esta razón se propone la siguiente representación:

```
struct Punto2D{
   double x;
   double y;
};

struct Nodo{
   struct Punto2D punto;
   struct Nodo *sigpunto;
};

// Define Poligono como un alias para Nodo*
typedef struct Nodo* Poligono;
```

Dada esta definición, un polígono se representa como una secuencia circular ordenada de nodos enlazados, por ejemplo, el triángulo de puntos (2,0),(4,0) y (3,1) se representa de la siguiente forma:



Teniendo en cuenta está representación, responder a las siguientes cuestiones:

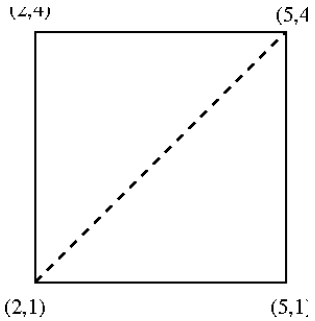
a) Construir una función que determine el número de lados que contiene la figura almacenada en una variable de tipo `Poligono`.

b) Suponiendo que existe una función llamada `PintaRecta (struct Punto2D p1, struct Punto2D p2)` que pinta una recta entre los 2 puntos que se le pasan como argumento, construir una función que permita pintar la figura que representa una determinada variable `Poligono`.

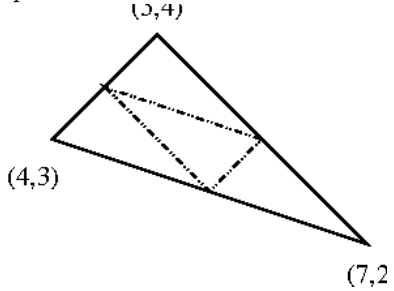
c) Implementar una función que permita crear en una variable de tipo `Poligono` un triángulo a partir de los tres puntos que lo definen.

d) Desarrollar una función que permita liberar la memoria reservada por una variable `Poligono`.

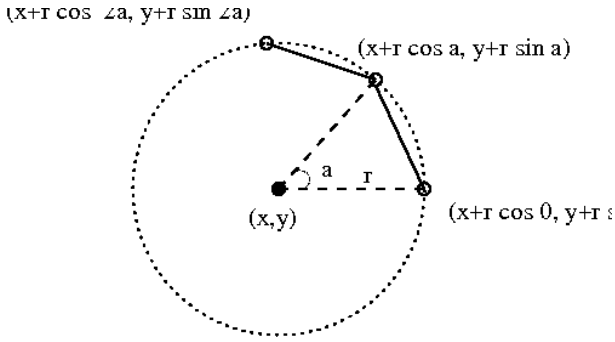
e) Sabiendo que una variable `Poligono` almacena un cuadrado, implementar una función que devuelva los dos triángulos que resultan de unir mediante una recta la esquina inferior izquierda del cuadrado con su esquina superior derecha.



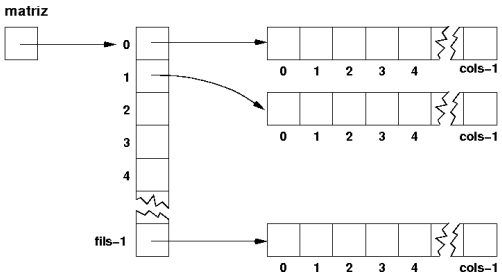
f) Construir una función que a partir de una variable `Poligono` que representa un triángulo devuelva el triángulo formado por los puntos medios de las rectas del triángulo original.



g) Desarrollar una función que permita construir un polígono regular de **n** lados inscrito en una circunferencia de radio **r** y centro **(x,y)**.



6. Supongamos que para definir matrices bidimensionales dinámicas usamos una estructura como la que aparece en la siguiente figura que denominaremos **Matriz2D_1**:



   a) Construir una función que escriba todas las componentes de una matriz bidimensional dinámica como la que se ha definido anteriormente.

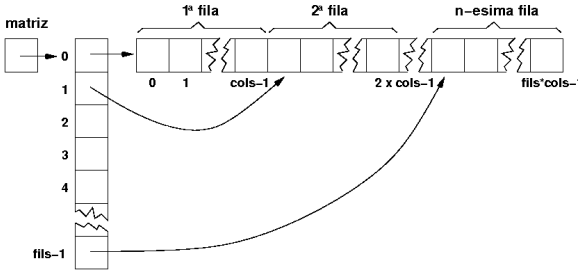
   b) Construir una función que dada una matriz de este tipo cree una copia.

   c) Implementar una función que extraiga una submatriz de una matriz bidimensional `Matriz2D_1`. Como argumento de la función se introduce desde qué fila y columna y hasta qué fila y columna se debe realizar la copia de la matriz original.

   d) Desarrollar una función que elimine una fila de una matriz bidimensional `Matriz2D_1`. Obviamente, la eliminación de una fila implica el desplazamiento hacia arriba del resto de las filas que se encuentran por debajo de la fila eliminada.

   e) Realizar una función como el anterior, pero que en vez de eliminar una fila, elimine una columna.

7. Supongamos que ahora decidimos utilizar una forma diferente para representar las matrices bidimensionales dinámicas a la que se propone en el ejercicio anterior. En este caso, usaremos una estructura semejante a la que aparece en la siguiente figura que denominaremos **Matriz2D_2**:



   a) Realizar los apartados a), b), c), d) y e) usando esta nueva representación de matrices bidimensionales dinámicas.

   b) Construir una función que dada una matriz bidimensional dinámica `Matriz2D_1` realice una copia de la misma en una matriz bidimensional dinámica `Matriz2D_2`.

   c) Desarrollar una función que realice el paso inverso al propuesto en el ejercicio anterior.