**1a Guía de Estructura de Datos**

**Nombre: Grupo:**

**Responda Falso o Verdadero ( F / V ) 0.25 ptos**

|  |  |
| --- | --- |
| 1.-En lenguaje C un **casting** se escribe usando **(Tipo)** | ( ) |
| 2.-En lenguaje C el **casting** se usa para conversion de tipos | ( ) |
| 3.-En lenguaje C la **declaracion** de una función no tiene cuerpo | ( ) |
| 4.-En lenguaje C la **definicion** de una función no tiene cuerpo | ( ) |
| 5.-En lenguaje C la l**lamada** a una función es una expresion | ( ) |
| 6.-En lenguaje C los **parámetros formales** no tienen nombre\_\_\_\_\_\_\_\_\_ | ( ) |
| 7.-Los elementos de un **arreglo** pueden ser **de tipos diferentes** | ( ) |
| 8.-Los elementos de un **arreglo** se eligen usando un numero (conocido como índice) | ( ) |
| 9.-Los elementos de un **arreglo** son anónimos | ( ) |
| 10.-El **primer elemento** de un **arreglo** esta en la **posición cero** (indice 0) | ( ) |
| 11,-En lenguaje C se puede usar una variable sin haberla declarado | ( ) |
| 12.-El **ámbito** de una **variable** son todos aquellos lugares del código donde es valido usar la variable | ( ) |
| 13.-Las **variables locales** se pueden usar en todo el programa | ( ) |
| 14.-Las **variables globales** se pueden usar en todo el programa | ( ) |
| 15.-Un **apuntador** es una variable que almacena la dirección de una localidad de memoria | ( ) |
| 16.-El **operador unario &** no se puede aplicar a una constante (por ejemplo &25) | ( ) |
| 17.-Si se aplica el **operador unario \*** a una variable se obtiene la dirección de dicha variable | ( ) |
| 18.-el **operador unario \*** solo se puede aplicar a un **apuntador** | ( ) |
| 19.-Si se aplica el **operador unario &** a una variable se obtiene la dirección de dicha variable | ( ) |
| 20.-En el **paso** de parámetros **por valor** la función llamada recibe **copias** de los **parámetros reales**. | ( ) |
| 21.-Los **tipos básicos** en lenguaje C se pasan a las **funciones** por referencia | ( ) |
| 22.-El **nombre** de un **arreglo** proporciona la **dirección** del **primer elemento** del arreglo | ( ) |
| 23.-El **nombre** de un **arreglo** es un **apuntador** | ( ) |
| 24.-En lenguaje C no es valido **sumar o restar** un numero **entero** a un **apuntador** | ( ) |
| 25.-En lenguaje C no es valido **sumar o restar** un numero **flotante** a un **apuntador** | ( ) |
| 26.-Si pa apunta a un elemento en particular de un **arreglo** entonces pa+i apunta i elementos después de pa | ( ) |
| 27-Si apu es un apuntador a un bloque de memoria entonces **apu[i] y \*(apu+i) son equivalentes** | ( ) |
| 29.-Todos los miembros (elementos) de una **estructura** deben ser del **mismo tipo** | ( ) |
| 29.-Los miembros (elementos) de una **estructura** son **anónimos** | ( ) |
| 30.-Los miembros (elementos) de una **estructura** **tienen nombre** | ( ) |
| 31-En lenguaje C los **tipos básicos** se pasan a las **funciones** por valor | ( ) |
| 32.-Los **tipos básicos** en lenguaje C se pasan a las **funciones** por **referencia** | ( ) |
| 33-En lenguaje C se deben evitar las **variables globales** | ( ) |
| 34.-En lenguaje C los **arreglos** se pasan a las **funciones** por referencia | ( ) |
| 35.-En el **paso** de parámetros **por valor** la función llamada recibe **copias** de los **parámetros reales**. | ( ) |
| 36.-La declaración de función (prototipo) int strcmp(char s1[], char s2[]);  es equivalente a int strcmp(char \*s1, char \*s2); | ( ) |
| 37.-El tipo de la **dirección** de un **apuntador** (o dicho de otro modo el tipo de la expresión &apun (donde apun es un apuntador)) **es un apuntador a apuntador** | ( ) |
| 38.-Un procedimiento (funcion en lenguaje C) es un conjunto de instrucciones a las que se asocia un nombre | ( ) |
| 39.-El **nombre** de una **función** es un **apuntador** | ( ) |
| 40.-El **nombre** de una  **función** proporciona la **dirección** de la localidad de memoria donde esta almacenada la **primer instrucción** de la función. | ( ) |
| 41-Si apu\_est es un apuntador a estructura y miembro es un miembro de dicha  estructura entonces el fragmento de código **apu\_est->miembro** es valido | ( ) |
| 42.-Un **apuntador a void** no puede **apuntar a** **cualquier tipo** de dato | ( ) |
| 43.-Si apu\_est es un apuntador a estructura entonces \*(apu\_est)**.**miembro y  apu\_est->miembro son equivalentes | ( ) |
| 44,-Si el tipo de una **función** es **void** entonces la función **no retorna** un **valor** | ( ) |
| 45.-En lenguaje C cuando una función termina su ejecución se saca su **marco** de la pila de llamadas | ( ) |
| 46.-Los **tipos de datos compuestos** son los que proporciona el lenguaje de  programación | ( ) |
| 47.-En lenguaje C una cadena es un **arreglo** de caracteres terminado en caracter nulo | ( ) |
| 48.-Las **pilas** son estructuras de datos **estáticas** | ( ) |
| 49.-Las **pilas** se usan para almacenar datos de forma temporal | ( ) |
| 50.-En una **pila** el 1er elemento en **entrar** es el 1o en **salir** | ( ) |
| 51.-La operación **pop saca** de la pila **al ultimo que entro** a la pila | ( ) |
| 52,-Para que un contenedor sea abstracto debe poder contener datos de cualquier tipo. | ( ) |
| 53,-en la implementación estática de la pila la pila esta vacía cuando la variable tope es cero. | ( ) |
| 54,-El usuario de la pila debe conocer la implementación de esta | ( ) |
| 55,-La implementación de la pila debe estar oculta | ( ) |

1.-Consta de 2 partes un **conjunto de operaciones** **y** una **representación**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a) pila | b) tipo de datos | c) objeto | d) ninguno de los anteriores | ( ) |

2.-En un (a) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_el 1er elemento en entrar es el ultimo en salir

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a) cola | b) lista ordenada | c) pila | d) tabla hash | ( ) |

3.-En un(a) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_el 1er elemento en entrar es el primero en salir

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| a) cola | b) árbol | c) pila | d) tabla hash | ( ) |

4-¿Cual de las siguientes **definiciones** de **función** es correcta? (Haga un análisis de los tipos en los parámetros y asignaciones)

|  |  |  |
| --- | --- | --- |
| a) void convie(int r,int a, int real, int ima){  real=r\*cos(a); ima=r\*sen(a);  } | b) void convie(int r,int a, int \*real, int \*ima){  \*real=r\*cos(a); \*ima=r\*sen(a);  } | ( ) |
| c) void convie(int r,int a, int \*real, int \*ima){  real=r\*cos(a); ima=r\*sen(a);  } | d) void convie(int r,int a, int real, int ima){  \*real=r\*cos(a); \*ima=r\*sen(a);  } |  |

5.-Cual de las siguientes formulas no es recursiva

|  |  |  |
| --- | --- | --- |
| a) cn, m=cn-1, m-1+cn-1, m | c) pn(x)=(1+x)\*pn-1(x) | ( ) |
| b) Sn=n+(n-1)+(n-2)+ ...+4+3+2+1 | d) todas son recursivas |  |

6.-Cual de las siguientes formulas no es recursiva

|  |  |  |
| --- | --- | --- |
| a) n!=n\*(n-1)\*(n-2)\* ...\*7\*6\*5\*4\*3\*2\*1 | c) n!=n\*(n-1)! | ( ) |
| b) 2n=2\*2(n-1) | d) fn=f(n-1)+f(n-2) |  |

7.-Cuando se **implementa** una **pila** usando un **arreglo y** una **variable entera tope** el valor almacenado en dicha variable indica.

|  |  |  |
| --- | --- | --- |
| a) cual es la posición del 1er elemento | b ) cual es la posición del ultimo elemento | ( ) |
| c) cual es la siguiente posición disponible | d) si la pila esta vacía |  |

8.-Una **pre-condicion** para la función **pop** en una pila finita es que la pila no debe estar

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A ) vacia | B ) llena | c) no vacia | d) ninguna de las anteriores | ( ) |

9.-Si elementos es un arreglo global de enteros , tope una variable global entera y cuando la pila esta vacia tope vale cero cual de las siguientes funciones push es la correcta?

|  |  |  |
| --- | --- | --- |
| A ) int push(int dato){  elementos[tope]=dato; tope++; } | B ) int push(int dato){  tope++; elementos[tope]=dato;} | ( ) |
| c) int push(int dato){  elementos[tope]=dato; } | d) int push(int dato){  elementos[tope]=dato tope--; } |  |

Suponga que los nodos de una lista simplemente ligada de enteros se definen en lenguaje C del siguiente modo:

|  |  |
| --- | --- |
|  | struct nodoL { int info; struct nodoL \*sig; };  typedef struct nodoL NodoL; |

10.-Si p y inicio son apuntadores a NodoL e inicio apunta al primer elemento de la lista.

Cual for se debería usar para **recorrer una lista simplemente ligada**?

|  |  |  |
| --- | --- | --- |
| a) for(p= inicio; p ; p->sig) | c) for(p= inicio; p ; p=p->sig) | ( ) |
| b) for(p= inicio; p ; p=sig) | d) for(p= inicio; p ; p=p[sig]) |  |

Si se tiene el siguiente código

NodoL \*nvo;

nvo=(NodoL \*)malloc(sizeof(NodoL));

11.-Cual de las siguientes instrucciones es correcta

|  |  |
| --- | --- |
| a) info=5; b) info->nvo=5; c) nvo->info=5; d) nvo->int info=5; | ( ) |

**Problema 1**.-Dada la estructura

struct libro { char titulo[100], autor[100], editorial[70];

int precio; };

typedef struct libro Libro;

Declare un variable de tipo apuntador a Libro, reserve memoria para dicha variable y guarde la siguiente información “Lolita”, “Nabokov”, “Anaya”, 160.00. Escriba la función sirva para imprimir los datos de un solo libro almacenado en una lista simplemente enlazada (que puede almacenar datos de cualquier tipo) y que dicha función se pueda pasar como 2o parámetro a la función que imprime la información de todos los nodos de esa lista.

**Listas**

Escribe una función que dadas dos listas simplemente enlazadas A y B, determine si son idénticas o

no.

Escribe un procedimiento que dada una lista simplemente enlazada A, obtenga una copia B de la misma.

-Escribe un procedimiento que dada una lista simplemente enlazada A, realice una copia simétrica B de la misma (es decir invertida).

-Escribe un procedimiento que dada una lista simplemente enlazada A, la almacene en un archivo.

-Escribe una función para sacar el promedio de los valores de una lista simplemente enlazada A

-Escribe una función para sacar el máximo de los valores de una lista simplemente enlazada A

-Escribe una función recursiva que encuentre el número de nodos de una lista simplemente enlazada A.

-Escribe una función recursiva que remueva duplicados de una lista ordenada.

-Escriba un programa para realizar suma de polinomios en una variable con listas enlazadas.

-Escriba un programa para realizar suma de polinomios en tres variables con listas enlazadas.

-Insertar en orden dada una lista enlazada de enteros ordenados del mas pequeño (at the head end) to largest, y un apuntador a un simple nodo conteniendo un entero, inserte el nodo en la lista so that it

sigue ordenada.

**Recursividad**

1-Escriba una función **recursiva** que devuelva **la suma de los primeros N enteros**

2.-a) La sucesión de fibonacci se define como

F0=1 , F1=1 y Fn=Fn-1+Fn-2 -Hacer una función recursiva que calcule el valor Fn para un n dado.

b) codifique, tanto de forma recursiva como de forma iterativa, una función que nos devuelva el máximo común divisor de dos números enteros utilizando el algoritmo de Euclides.

ALGORITMO DE EUCLIDES

Dados dos números enteros positivos m y n, tal que m > n, para encontrar su máximo común

divisor (es decir, el mayor entero positivo que divide a ambos):

- Dividir m por n para obtener el resto r (0 ≤ r < n)

- Si r = 0, el MCD es n.

-Si no, el máximo común divisor es MCD(n,r).

3-Implemente, tanto de forma recursiva como de forma iterativa, una función que le dé la invierta a una cadena de caracteres.

-Programar una **función recursiva** que imprima los dígitos de un número decimal en orden

inverso.

4.-Escriba una **función recursiva** que dado un numero decimal retorne cuantos dígitos tiene

5-Escriba una **función recursiva** que transforme un número entero decimal positivo a notación binaria.

6- Escriba una **función** **recursiva** que permita **sumar los dígitos de un númer**o.

Ejemplo: Entrada: *123* Resultado: *6*

*7*- Escriba una función **recursiva** que transforme un número expresado en notación binaria a un número entero decimal.

8.-Escriba una función recursiva para evaluar un polinomio ( sin usar pow)

9.-Escriba una función recursiva que imprima un arreglo

10.-Escriba una función recursivaque sume los elementos de un arreglo

11.-Escriba una función recursiva que **busque un valor** en un arreglo **.**

12.-Escriba una función recursiva que encuentre el máximo de un arreglo

13.-Escriba una función recursiva cuenta, que devuelve el número de veces que un valor ocurre en un arreglo.

14.-Codifique una función recursiva que **busque un nodo** en **un lista.**

15.-Escriba una función recursiva PrimeraPosicion que devuelve la primera posición en la que aparece un valor en un arreglo.

16.-Codifique una función recursiva que devuelva la **cuenta del numero de nodos de una lista simplemente ligada.**

17.-Escriba una función recursiva cuenta, que devuelve el número de veces que un valor ocurre en una lista.

18.-Codifique función **busca un nodo** en **una lista simplemente ligada.**

19.-Escriba una función recursiva PrimeraPosicion que devuelve la primera posición en la que aparece un valor en una lista.

|  |  |
| --- | --- |
| Prob6. Dado el código de la sig. función que imprime de forma iterativa los dígitos un número decimal en orden inverso escriba una versiónrecursivade esta función. | Prob7. Dado el código de la siguiente función que imprime una lista de forma recursiva modifíquelo para que retornecuantoselementoshay **en la lista** o que sume los elementos en la lista |
| void digitos(long int dato){  int residuo=dato%10;  while(residuo > 0){  printf("%d=", residuo);  dato=dato/10;  residuo=dato%10;  }  } | void imprime(NodoL \*cab){  if(!cab) return ;  printf(“%d ”, cab->info);  imprime(cab->sig);  } |

-Programe un método recursivo que transforme un número expresado en notación binaria a un número entero.

-Dado un arreglo de enteros, programar algoritmos recursivos que calculen :

-Programar un algoritmo recursivo que determine si un número es par.

-Sabiendo que 0 es par, es decir,

EsPar(0) = true

EsImpar(0) = false

y que la paridad de cualquier otro entero positivo es la opuesta que la del entero anterior,

desarrolle las funciones lógicas, mutuamente recursivas, EsPar y EsImpar, que se

complementen a la hora de averiguar la paridad de un entero positivo.

|  |  |
| --- | --- |
| I.-Cual es la función de búsqueda binaria? ( ) II.-Cual es la función de búsqueda secuencial? ( ) | |
| int busca(int key, int tab[], int tam){ A)  int i;  for(i=0;i < tam; i++)  if(key == tab[mid])  return i;  return -1;  } | int busca(int key, int tab[], int tam){ B)  int low, high, mid;  low = 0; high = tam - 1;  while (low <= high) {  mid = (low+high) / 2;  if (key < tab[mid])  high = mid - 1;  else if ( key > tab[mid])  low = mid + 1;  else  return mid;  }  return -1; } |

Responda las preguntas en base al siguiente código

|  |  |
| --- | --- |
| #include<stdio.h>  #include<conio.h>  void intercambia(char \*a, char \*b){  char tmp;  tmp = \*a;  \*a = \*b;  \*b = tmp;  }  void ordenshell(char a[],int n)  {  int i, j, h;  char v;  for (h = 1; h <= n; h = 3\*h+1)  ;  h=h/3;  for ( ; h > 0; h /= 3)  {  for (i = h; i < n; i ++)  {  v = a[i];  j = i;  while (j >= h && a[j-h] > v)  {  a[j] = a[j-h];  j=j-h;  }  a[j] = v;  }  }  } | void seleccion(char a[], int n)  {  int i, j, indmin;  char min;  for(i=0;i < n;i++){  min=a[i];  indmin=i;  for(j=i+1;j < n;j++)  if(a[j] < min){  min=a[j];  indmin=j;  }  intercambia(&a[i], &a[indmin]);  }  }  void insercion(char a[], int n)  {  int i,j;  char v;  for(i=2;i < n;i++){  v=a[i];  j=i;  while(a[j-1] > v)  {  a[j]=a[j-1];  j--;  }  a[j]=v;  }  }  void burbuja(char a[], int n)  {  int i, j;  for(i=n;i>=0;i--)  for(j=2;j<i;j++)  if(a[j-1] > a[j]){  intercambia(&a[j-1],&a[j]);  }  } |

|  |  |
| --- | --- |
| ordenrapido(char a[], int izq, int der){  int i, j;  char v;  if (der > izq){  v = a[der];  i = izq-1;  j = der;  while (1)  {  while (a[++i] < v)  ;  while (a[--j] > v)  ;  if (i >= j)  break;  intercambia(&a[i], &a[j]);  }  intercambia(&a[i], &a[der]);  ordenrapido(a,izq, i-1);  ordenrapido(a,i+1, der);  }  }  void main(){  char a[]="EJEMPLOAORDENAR";  clrscr();  puts(a);  seleccion(a,strlen(a));  insercion(a, strlen(a));  burbuja(a,strlen(a));  ordenshell(a, strlen(a));  ordenrapido(a,0, strlen(a)-1);  puts(a);  } | El for interno en la función **selección** que hace?  for(j=i+1;j < n;j++)  if(a[j] < min){  min=a[j];  indmin=j;  }  El ciclo interno en la función **inserción** que hace?  while(a[j-1] > v)  {  a[j]=a[j-1];  j--;  }  El for interno del la función **burbuja** que hace?  for(j=2;j<i;j++)  if(a[j-1] > a[j]){  intercambia(&a[j-1],&a[j]);  }  Cuando el valor de h es igual a 1 el ciclo interno en la función **ordenshell** que hace?  for (i = h; i < n; i ++){  v = a[i];  j = i;  while (j >= h && a[j-h] > v){  a[j] = a[j-h];  j=j-h;  }  a[j] = v;  }  Que hace cada uno de los 2 ciclos internos de **ordenrapido**?  while (a[++i] < v)  ;  while (a[--j] > v)  ; |

-Cual es la principal desventaja del método de la **burbuja**?

-Cual de las funciones de ordenación es recursiva

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| a) inserción | b) selección | c) burbuja | d) ordenshell | e) ordenrapido | ( ) |

-Haga la prueba de escritorio para la cadena

char a[]=”EJEMPLO”; y la llamada **ordenrapido**(a, 0, 6)

Analice el código de las siguientes funciones para responder las preguntas de abajo

|  |  |  |
| --- | --- | --- |
| A) int funcion(char \*s, char \*t){  int i;  for (i = 0; s[i] == t[i] ; i++)  if ( !s[i] )  return 0;  return s[i] - t[i];  } | B) int funcion(char \*s){  char \*p = s;  while (\*p )  p=p+1;  return p - s;  } | |
| C) void funcion(char \*f, char \*d){  while( \*d=\*f ){  d++; f++;  }  } | D) void funcion(void \*arr[], int n, void (\*f)(void \*)){  int i;  for(i = 0; i < n ; i = i+1)  f( arr[i] );  } | |
| E) void \*funcion(void \*arr[], int n, int (\*d)(void \*, void \*)){  void \*m=arr[0];  int i;  for(i = 0; i < n ; i = i+1)  if( d( m, arr[i] ) < 0 )  m=arr[i];  return m;  } | | F) int intercambia(int v[], int i, int j){ int tmp;  tmp = v[i]; v[i] = v[j]; v[j] = tmp;  }  void funcion(int arr[], int n){  int i, m=n/2;  for(i = 0 ; i < m ; i++)  intercambia(arr, i, n-1-i);  } |

Cual función encuentra el máximo de un arreglo? \_\_\_\_\_\_\_\_\_\_Cual función invierte un arreglo? \_\_\_\_\_\_\_\_\_\_

Cual función obtiene la longitud de una cadena? \_\_\_\_\_\_\_\_\_\_Cual función imprime un arreglo? \_\_\_\_\_\_\_\_\_\_

Cual función copia una cadena? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Cual función compara dos cadenas?\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| I.-void func1(void \*arr[], int n, void (\*f)(void \*)){  int i;  for(i=0;i < n; i=i+1) f(arr[i]);  } | II.-void func2(NodoL \*inicio, void (\*f)(void \*)){  NodoL \*p;  if( !inicio ) return ;  for(p=inicio;p;p=p->sig) (\*f)(p->dato); } |
| III.-void func3(void \*arre1[], void \*arre2[], int n,  void \*(\*f)(void \*)){  int i;  for(i=0 ; i < n; i = i+1)  arre2[i]=f(arre1[i]);  } | IV.-void func4(NodoL \*inicio1, void \*(\*f)(void \*)){  NodoL \*p, \*inicio2=(NodoL \*)NULL;  if( !inicio1 ) return ;  for( p= inicio1 ; p ; p = p->sig)  inicio2=creaNodoL(f(p->dato), inicio2);  return inicio2;  } |
| V.-void \*func5(void \*arr[], int n, void \*ap,  int (\*f)(void \*, void \*)){  int i;  for(i=0;i < n; i=i+1)  if(! (f)(arr[i], ap) )  return arr[i];  return NULL;  } | VI.-void \*func6(NodoL \*inicio, void \*ap,  int (\*f)(void \*, void \*)) {  NodoL \*p=(NodoL \*)NULL;  for(p = inicio; p ; p = p->sig)  if(! (\*f)(p->dato, ap) )  return p->dato;  return (void \*)NULL;  } |

De las funciones de arriba cual es para

1.-copia profunda de una lista ?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2.-buscar un dato en un arreglo ?\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3.-copia profunda de la lista un arreglo ?\_\_\_\_\_\_\_ 4.-imprimir una lista ?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5.-buscar un dato en una lista ?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_6.-imprimir un arreglo ?\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |
| --- | --- |
| A) int f(int n) {  if(n < 10) return 1;  return 1 + f(n / 10);  } | B) int e(int arr[],int n,int g,int x){  if(n==0)  return arr[g];  return e(arr, n-1, g, x)\*x+arr[g-n];  } |
| C) int t(int n,int m){  if(m > n) return 0;  if(m == 0 || m == n) return 1;  return t(n-1,m-1)+t(n-1,m);  } | D) int f(int a, int b){  if(b==0) return 1;  return f(a,b-1) \* a;  } |
| E) int r(int arr[], int n) {  int mR;  if (n == 1) return arr[0];  mR = r(arr, n-1);  if(mR < arr[n-1])  return arr[n-1];  return mR;  } | void f(struct Node \*n) {  if (!n ) {  f(n->l);  printf("%d,", n->d);  f(n->r);  }  } |

De las funciones de arriba cual es para

encontrar el máximo de un arreglo? \_\_\_\_\_\_\_\_ calcular los numeros triangulo de pascal?\_\_\_\_

eleva un numero a una potencia?\_\_\_\_\_\_\_\_\_\_\_ evaluar un polinomio?\_\_\_\_\_\_\_\_\_\_\_\_\_

contar los digitos de un numero?\_\_\_\_\_\_\_\_\_\_\_ Recorrido inorden de un arbol binario? \_\_\_\_\_\_

|  |  |
| --- | --- |
| A) void func(NodoL \*inicio, void (\*af)(void \*)){  NodoL \*p;    if(!inicio) return ;  for(p=inicio;p;p=p->sig)  af(p->info);  } | B) NodoL \*func(NodoL \*inicio, void \*key, int (\*af)(void \*, void \*)){  NodoL \*p;  for(p=inicio;p;p=p->sig)  if(!af(p->info, key))  return p;  return p;  } |
| C) void func(NodoL \*inicio1, NodoL \*\*inicio2,void \*(\*af)(void \*)){  NodoL \*p;    if(!inicio1) return ;  \*inicio2=(NodoL \*)NULL;  for(p=inicio1;p;p=p->sig){  \*inicio2=creaNodoL((\*af)(p->info), \*inicio2);  }  } |  |

1.-Que función sirve para buscar en la lista? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2.-Que función sirve para imprimir la lista? \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3.-Que función sirve para realizar la copia profunda de la lista\_\_\_\_\_\_\_\_\_\_\_