UNIVERSIDAD AUTONOMA DE MADRID
ESCUELA POLITÉCNICA SUPERIOR
ESTRUCTURAS DE DATOS Y ALGORITMOS

Curso 2008-09 **Examen parcial**

APELLIDOS:

NOMBRE:

1. **(0.5 puntos)** Enumera las características de un algoritmo

1. Especificación precisa entrada y salida
2. Número finito de pasos
3. Debe terminar

2. **(0.5 puntos)** ¿Cuáles son las ventajas de utilizar una lista enlazada en vez de una tabla para representar el TAD Lista?

1. No se desperdicia espacio
2. Borrados e inserciones más rápidos

3. **(0.5 puntos)** ¿Qué solución se adopta en el TAD Lista para que la inserción al comienzo de la lista y el borrado del primer elemento no sean casos especiales?

Nodo cabecera

Para las preguntas 4 a 6, suponer la siguiente definición de un TAD Lista con nodo cabecera.

```
typedef struct Nodo *PNodo;
typedef struct Nodo {
   int dato;
   PNodo siguiente;
} Nodo;
typedef struct Nodo *Lista;
```

4. **(0.5 puntos)** Dada la siguiente implementación para la función crearLista

```
void crearLista(Lista L)
{
     L->siguiente = NULL;
}
```

completar la implementación de la función `main` para que funcione correctamente.

```
void main()
{
     Lista L;
```

L = (PNodo) malloc(sizeof(Nodo));

```
     crearLista(L);
}
```

5. **(0.5 puntos)** Completar la implementación de la función `insertar` en una lista para que funcione correctamente, es decir, inserte un nuevo nodo con dato X después del nodo apuntado por P.

```
int insertar(int X, PNodo P){
  PNodo NodoTemp;

  NodoTemp = (PNodo) malloc(sizeof(Nodo);
  if (NodoTemp == NULL){
    printf("Error de asignación: No hay espacio");
    return 1;
  }
  NodoTemp->dato = X;
  NodoTemp->siguiente = P->siguiente;
```

P->siguiente = NodoTemp;

```
  return 0;
}
```

6. **(0.5 puntos)** Completar la implementación de la función esUltimo para que devuelva 1 si P apunta al último nodo de una lista.

```
int EsUltimo(PNodo P){
   return
```

P -> siguiente == NULL;

```
}
```

7. **(0.5 puntos)** Cuando se declara un puntero a una estructura no siempre es necesario reservar espacio para la estructura. Pon un ejemplo de un caso en el que no es necesario reservar dicho espacio.

Si el puntero sólo lo queremos para recorrer una lista de estructuras

8. **(0.5 puntos)** Indica una ventaja de las listas doblemente enlazadas con respecto a las listas enlazadas.

Simplifican el borrado porque hay un puntero al elemento anterior

9. **(0.5 puntos)** ¿Qué restricción tiene el TAD Pila con respecto al TAD Lista en relación a la operación de borrado?

Sólo se puede borrar el primer elemento de la pila

10. **(0.5 puntos)** ¿Por qué se llama a las pilas listas LIFO?

Last In First Out (Último en entrar, primero en salir)

11. **(0.5 puntos)** Suponer la siguiente definición de un TAD Pila con nodo cabecera.

```
typedef struct Nodo *PNodo;
typedef struct Nodo {
   int dato;
   PNodo siguiente;
} Nodo;
typedef struct Nodo *Pila;
```

completar la implementación de la función cima para que funcione correctamente.

```
int cima(Pila P)
{
     if (!esVacia(P))
          return
```

P -> siguiente -> dato;

```
     printf("Pila vacia\n");
     return 0;
}
```

12. **(0.5 puntos)** Suponer la siguiente definición de un TAD Bicola implementado como una lista doblemente enlazada

```
typedef struct Nodo *PNodo;
typedef struct Nodo {
```

```
   int dato;
   PNodo siguiente;
   PNodo anterior;
} Nodo;
typedef struct {
   PNodo izquierdo;
   PNodo derecho;
} Bicola;
```

completar la implementación de la función borrar_izquierdo para que borre el primer elemento de la bicola, es decir, el elemento al que apunta el puntero izquierdo.

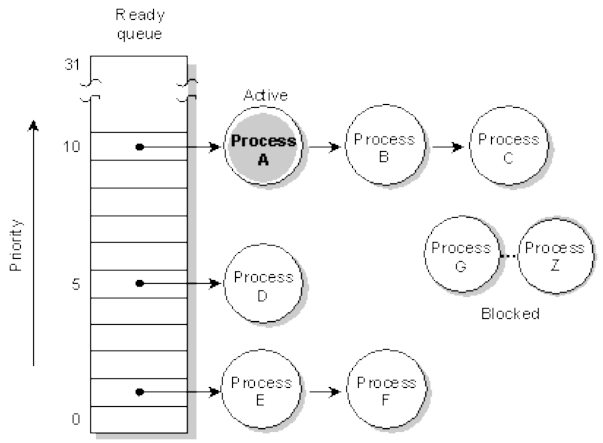
```
void borrar_izquierdo(Bicola *B)
{
   PNodo Primero;

   if (esVacia(*B))
     printf("Cola vacia\n"),
   else {
     Primero = B->izquierdo;
     if (B->izquierdo == B->derecho) {
        B->izquierdo = NULL;
        B->derecho = NULL;
     }
     else {
```

B -> izquierdo = Primero -> siguiente;
B -> izquierdo -> anterior = NULL;

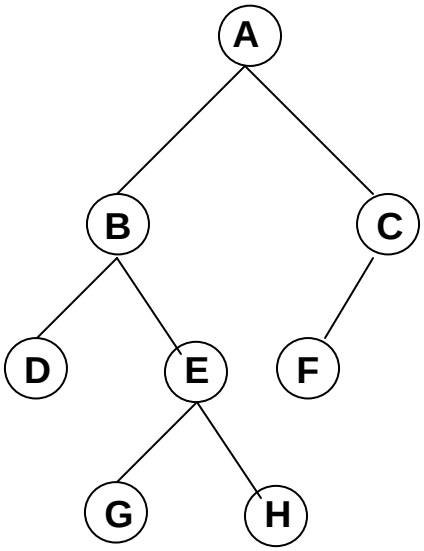
```
     }
     Primero->siguiente = NULL;
     free(Primero);
   }
}
```

13. **(0.5 puntos)** ¿Qué es una cola de prioridad? Representa gráficamente una posible forma de implementar una cola de prioridad.

Tabla con punteros a colas. Cada elemento de la tabla representa una prioridad.



14. **(0.5 puntos)** Para el siguiente árbol, ¿cuál es su recorrido en preorden?



A, B, D, E, G, H, C, F

15. **(0.5 puntos)** ¿Qué es un árbol binario de búsqueda?

Árbol binario en el que el valor de cualquier nodo es superior al valor de los nodos de su subárbol izquierdo e inferior al valor de los nodos de su subárbol derecho

16. **(0.5 puntos)** Para llevar a cabo una operación de borrado en un árbol binario de búsqueda, si el nodo que se desea borrar tiene dos hijos, ¿por qué otro nodo del árbol hay que sustituirle?

El nodo más pequeño de su subárbol derecho

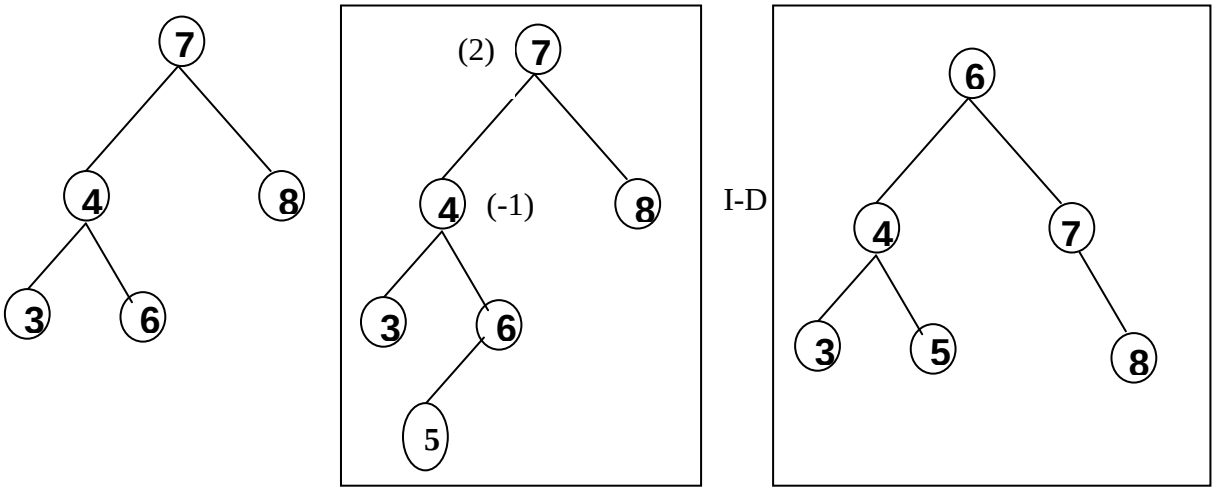
17. **(0.5 puntos)** ¿Cómo se localiza el nodo de valor más pequeño en un árbol binario de búsqueda?

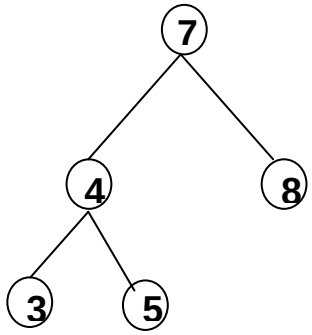
Ultimo nodo de la rama izquierda

18. **(0.5 puntos)** ¿Cuál es la principal ventaja de los árboles AVL?

El número máximo de comparaciones para búsqueda es <= log(n), donde n es el número de nodos del árbol

19. **(0.5 puntos)** Insertar el elemento 5 en el siguiente árbol AVL, indicando razonadamente los pasos seguidos.



20. **(0.5 puntos)** Sobre el AVL de la pregunta anterior y considerándolo exclusivamente como un árbol binario de búsqueda, indicar razonadamente su estado tras la extracción del elemento 6.

Como el elemento 6 tiene 2 hijos, se sustituye por el nodo de valor más pequeño de su subárbol derecho, es decir el 7