Algoritmos y Programación II (75.41) Cátedra Lic. Gustavo Carolo			Integrador 18/07/2005
Tema 1		**Ejercicio**	**Puntaje**
Apellido y Nombre			
Padrón	**Cuat. Cursada:**		
Nota Final			
Corrigió			

Práctica

1. Dado el siguiente TDA MENSAJERO compuesto por:

- Arbol usuarios TDA AB
  - ID usuario (clave de ordenamiento del arbol)
  - Alias
  - Cantidad mensajes recibidos
  - Cantidad mensajes entregados
  - Cola mensajes entrada
    - ID emisor
    - ID receptor
    - Mensaje
  - Cola mensajes salida
    - ID emisor
    - ID receptor
    - Mensaje

Se pide:

- Definir todos los elementos de la estructura dada.
- Desarrollar una función abstracta **Enviar Mensajes**, que dada la estructura descripta, **un ID usuario desde y un ID usuario hasta,** envie los mensajes que se encuentran en la cola de salida de los usuarios que están dentro de rango dado. Si un mensaje no pudo ser recibido por un destinatario es reenviado a la cola de entrada de su emisor. Devolver el resultado de la operación con los errores que crea conveniente.

Nota: *Los procedimientos de búsqueda, recorrido, actualización y/o inserción en el árbol deberán ser realizados en forma recursiva. Tener en cuenta que la estructuras ordenadas no deben recorrerse demás en la búsqueda por la clave.*

2. Dado un TDA AB, desarrollar la primitiva **AB RAMA MAYOR PESO**, que reciba como parámetro una variable del mencionado TDA, una variable del tipo entero, y devuelva en dicha variable el número de nodos que tiene la rama mas pesada (es decir rama con mas nodos). Dicha primitiva deberá devolver el resultado de la operación (**-1** si la rama **izquierda** es **mayor**, **1** si la rama **derecha** es **mayor**, o **0** si **ambas** ramas son **iguales**).No se pueden utilizar estructuras auxiliares, como así tampoco otras primitivas.

Entregar la resolución de la Teoría y la Práctica en hojas separadas.

1)

```
typedef enum Tmovim { raiz, padre, izq, der }

typedef struct {
                 int ID_emisor;
                 int ID_receptor;
                 char Mensaje [255];
                } TelemCola;

typedef struct {
                 int ID_usuario;
                 char Alias[10];
                 int Cantidad_recibidos;
                 int Cantidad_enviados;
                 Tcola Cola_entrada;
                 Tcola Cola_salida;
                } TelemAB;

typedef struct {
                 TAB ab;
                } TDA_Mensajero;


int Enviar_Mensajes ( TDA_Mensajero *M, int IDD, int IDH )

{
  if Ab_Vacio(M→ab)
     return 1;
  else
       {
         Tmovim Mov=raiz;
         recorrido_en_rango ( M→ab, &M→ab, IDD, IDH, Mov );
        }
 return 0;
}

void recorrido_en_rango ( TAB ab, TAB *pab, int IDD, int IDH, Tmovim Mov )

{
 TelemAB eab;
 int error;
 Ab_MoverCte ( &ab, Mov, &error );
 if !( error )
    {
      Ab_ElemCte ( ab, &eab);
      if ( eab.ID_usuario >= IDD ) && ( eab.ID_usuario <=IDH )
         procesar ( eab, pab );
      if ( eab.ID_usuario > IDD )
         recorrido_en_rango ( ab, pab, IDD, IDH, izq );
      if ( eab.ID_usuario < IDH )
         recorrido_en_rango ( ab, pab, IDD, IDH, der );
    }
}
```

```
void procesar ( TelemAB eab,  TAB *pab )

{
 TelemCola ec;
 TelemAB eaux;
 int enc;
 int error;
 while !( C_Vacia( eab.Cola_salida ) )
   {
       C_Sacar ( &eab.Cola_Salida, &ec );
       enc=busqueda ( pab, ec.ID_receptor, raiz );
       if (Enc)
         {
          Ab_ElemCte ( pab, &eaux );
          error=C_Agregar ( &eaux.Cola_entrada )
          if !( Error )
             {
              ( eaux.Cantidad_recibidos )++;
              Ab_ModifCte ( pab, &eaux );
             }
         }
      if (error) || !(enc)
        {
         error=C_Agregar ( &eab.Cola_Entrada, ec );
         ( eab.Cantidad_recibidos )++;
        }
     ( eab.Cantidad_enviados )++;
      enc=busqueda ( pab, eab.ID_usuario, raiz );
  }
}

int busqueda( TAB *pab, int ID, Tmovim *mov )

{
 TelemAB eab;
 Terror error;
  AB_MoverCte( pab,mov, &error )
  if (error)
     return 1;
  else
     {
      AB_ElemCte ( pab, &eab );
      if ( eab.ID_usuario==ID )
         return 0;
      else
       {
        if (eab.clave > clave)
          mov=izq;
        else
          mov=der;
       return busqueda(pab,clave,&mov);
   }
 }
```

2)

```
int AB_RAMA_MAYOR_PESO (TAB ab, int *cantidad)

{
  return  Compara_ Pesos (Peso(ab.raiz→izq), Peso(ab.raiz→der),cantidad);
}

int Peso (TNODO_AB *p)

{
  if !(p)
     return 0;
  else
     return (1+Peso(p→izq)+Peso(p→der));
}

int Compara_Pesos (int a, int b,int *cant)

{
 cant=a
 if (a=b)
      return 0;
  else
      if (a>b)
         return -1;
      else
          {
           cant=b;
           return 1;
          }
}
```