Programación 1
Segundo parcial 2012
Solución

Ejercicio 1

```
function PotenciaRapida(base,exponente: integer): integer;
var
   factor,cociente,resultado : integer;
begin
   { inicialización }
   factor:= base;           { los Bs de la letra }
   cociente:= exponente;  { los ns de la letra }
   resultado:= 1;         { acumulador del resultado }

   while cociente > 1 do
   begin
      { si es impar se acumula }
      if cociente mod 2 = 1 then
         resultado:= resultado * factor;

      { avanzar siguientes de las dos secuencias }
      factor:= factor * factor;
      cociente:= cociente div 2;
   end;

   { el último B  (para cociente=1) se acumula siempre }
   PotenciaRapida:= resultado * factor;

end; { PotenciaRapida }
```

Ejercicio 2

Parte a)

```
{ función auxiliar no pedida }
function valor_num(num : Numero):  real;
begin
   case num.tipo of
     tentero : valor_num:= num.ivalor;
     treal   : valor_num:= num.rvalor;
   end; { case }
end;

function MenorNumero(a,b: Numero) : boolean;
begin
   MenorNumero:= valor_num(a) < valor_num(b)
end; { MenorNumero }
```

Parte b)

```
procedure obtener_region_b(    num       : Numero;
                               arreglo   : ArregloNumero;
                           var region_b  : ArregloConTope);
var
   i : integer;
begin
   { inicializo array con tope}
   region_b.tope:= 0;

   { busco comienzo de región B }
   i:= 1;
   while (i <= MAX) and not MenorNumero(arreglo[i],num) do
      i:= i + 1;

   { recorro región B (posiblemente vacía) y copio  }
   while (i <= MAX) and MenorNumero(arreglo[i],num) do
   begin
      { agrego celda al array con tope }
      region_b.tope:= region_b.tope + 1;
      region_b.info[region_b.tope]:= valor_num(arreglo[i]);

      i:= i + 1;
   end;
end; { obtener_region_b }
```

Ejercicio 3 (Puede resolverse de varias formas diferentes)

Solución 1. Sin modificar contenido de las celdas, solo ajustando los punteros:

```
procedure rotacion(var l: Lista);
var
   ultimo,primero: Lista;
begin

   if (l <> NIL) and (l^.sig <> nil) then
   begin
      { busco último }
      ultimo:= l;
      repeat
         ultimo := ultimo^.sig
      until ultimo^.sig =nil;

      { desengancho el primero }
      primero:= l;
      l:= l^.sig;

      { engancho primero al final}
      ultimo^.sig :=  primero;
      primero^.sig := NIL;
   end

end;
```

Solución 2. Sin cambiar los punteros, corriendo el contenido (elem):

```
procedure rotacion2(var l: Lista);
var
   q    : Lista;
   temp : integer;
begin

   if (l <> NIL) and (l^.sig <> nil) then
   begin

      { guardo valor del primero }
      temp:= l^.elem;

      { recorro lista corriendo todos los elementos una celda hacia adelante}
      q:= l;
      repeat
         { corro elemento un lugar hacia adelante }
         q^.elem := q^.sig^.elem;
         q:= q^.sig;
      until q^.sig = nil;

      { pongo el valor del primero en última celda }
      q^.elem:= temp;

   end

end; { rotacion2 }
```

Solución 3. Creando una celda nueva y borrando la primera. Esta solución puede ser menos eficiente que las anteriores por invocar new y dispose:

```
procedure rotacion3(var l: Lista);
var
   ultimo,primero : Lista;
   temp           : integer;
begin

   if (l <> NIL) and (l^.sig <> nil) then
   begin
      primero:= l;
      { busco último }
      ultimo:= l;
      repeat
         ultimo := ultimo^.sig
      until ultimo^.sig =nil;

      { creo celda nueva al final }
      new(ultimo^.sig);
      ultimo:= ultimo^.sig;
      ultimo^.elem:= primero^.elem;
      ultimo^.sig:= nil;

      { borro la primera celda }
      l:= l^.sig;
      dispose(primero);

   end
end;
```