Tarea 3

Para los siguientes programas realice los siguientes pasos

1. Determinar cual es el peor de los casos respecto al tiempo de ejecución
2. Hacer una tabla que enumere las operaciones que se puede suponer que se ejecutan en tiempo constante y determinar cuántas veces se ejecuta cada una.
3. Derivar una fórmula para el tiempo de ejecución del algoritmo
4. En caso de que la fórmula sea una ecuación de recurrencia, resolver la ecuación
5. Determinar el orden de complejidad del algoritmo

Nota: Los algoritmos de solución de cada problema no son necesariamente las soluciones más eficientes

```
fun isPrime (n: nat) ret b: bool
var i : nat
{true}
b,i := true, 2
{i≤n ∧ b = (∀ k | 2 ≤ k < i : n mod k > 0) }
do i<n ∧ b → b,i := (n mod i > 0),i+1 od
ret b
{b = (∀ k | 2 ≤ k < n : n mod k > 0) }
```

```
fun division (a: nat, b: nat) ret <q,r>: pair of nat
{b>0}
q,r := 0,a;

{b>0 ∧ a = q*b + r}
do r ≥ b → q,r := q+1,r-b
od
ret <q,r>
{a = q*b + r ∧ r < b}
```

```
fun search (a : array [0,n) of real, e: real, f: nat) ret i: nat
{0 ≤ f < n}
if a[f] = e → i:= f
[] a[f] ≠ e ∧ f=0 → i: = -1
[] a[f] ≠ e ∧ f>0 → i:= search(a,e,f-1)
fi
ret i
{(i=-1 ∧ (∀ k | 0 ≤ k ≤ f : a[k] ≠ e)) ∨ a[i] = e}
```

```
fun search (a : array [0,n) of real, e: real, s: nat, f: nat) ret i: nat
var m : nat
{0 ≤ s < n ∧ s ≤ f < n}
m:= s + f
if m mod 2 = 0 → m:= m / 2
[] m mod 2 = 1 → m:= (m-1) / 2
fi
{0 ≤ s ≤ m ≤ f < n ∧ (s=f ∨ m<f)}
if a[m] = e → i:= m
[] a[m] ≠ e ∧ s=f → i: = -1
[] a[m] ≠ e ∧ s<f → i:= search(a,e,m+1,f);
                     if i=-1 → i:= search(a,e,s,m);
                     [] i≠-1 → skip;
                     fi
fi
ret i
{(i=-1 ∧ (∀ k | s ≤ k < f : a[k] ≠ e)) ∨ a[i] = e}
```

```
fun sumMainDiagonal (a : matrix [0,n)[0,n) of real) ret s: real
var i: nat
var j: nat
{true}
i:=0
{s=(∑ k | 0 ≤ k < i : a[k,k])}
do i < n →
     j:=0
     {(i≥j ∧ s=(∑ k: 0≤k<i | a[k,k]))∨(i<j ∧ s=(∑ k: 0≤k≤i | a[k,k]))}
     do j < n →
          if i=j → s:= s+a[i,j]
          [] i≠j → skip
          fi
          j:= j+1
     od
     i:=i+1
od
ret s
{s=(∑ k | 0 ≤ k < n : a[k,k])}
```

```
fun sumMainDiagonal (a : matrix [0,n)[0,n) of real) ret s: real
var i: nat
{true}
i:=0
{s=(∑ k | 0 ≤ k < i : a[k,k])}
do i < n → s,i:=s+a[i,i],i+1 od
ret s
{s=(∑ k | 0 ≤ k < n : a[k,k])}


fun search (a:matrix [0,m)[0,n) of real, e: real) ret <i,j>: pair of nat
var i,j: nat
i,j := 0,0;
{(i<m ∧ j<n ∧ (∀ k,l | 0≤k<i ∧ 0≤l<n : a[k,l]≠e))
           ∧ (∀ l | 0≤l<j : a[i,l]≠e))}
do i≠m ∧ a[i,j] ≠ e →
     if j<n-1 → j:= j+1
     [] j=n-1 → i,j:= i+1,0
     fi
od
ret <i,j>
{(i=m ∧ (∀ k,l | 0≤k<m ∧ 0≤l<n : a[k,l] ≠ e)) ∨ a[i,j]=e}
```