1.- Usando la ***notación O,*** *determinar la eficiencia de las siguientes* funciones:

(a) → $O(n \log_2 n)$

```
void eficiencia1(int n)
{
  int x=0; int i,j,k;
           for(i=1; i<=n; i+=4)
                for(j=1; j<=n; j+=[n/4])
                     for(k=1; k<=n; k*=2)
                          x++;
}
```

$O(1)$ · $O(\log_2 n)$ · $O(\log_2 n)$ · $O(1)$ → $O(n \cdot \log_2 n)$

(b)

int eficiencia2 (bool existe) ↔ $O(n \log_2 n)$

```
{
   int sum2=0; int k,j,n;

   if (existe)
         for(k=1; k<=n; k*=2)
              for(j=1; j<=k; j++)
                   sum2++;
   else
          for(k=1; k<=n; k*=2)
            for(j=1; j<=n; j++)
                   sum2++;
 return sum2;
}
```

$O(1)$ · $O(K)$ · $O(K \log_2 n)$

$O(1)$ · $O(n)$ · $O(n \cdot \log_2 n)$

→ $O(n \log_2 n)$

(c)

void eficiencia3 (int n) → $O(n \log_2 n)$

```
{
  int j; int i=1; int x=0;
   do{
      j=1;
      while (j <= n){
         j=j*2;
         x++;
      }
      i++;
   }while (i<=n);
}
```

$O(1)$ · $O(\log_2 n)$ · $O(n \log_2 n)$

void eficiencia4 (int n)

```
{
int j; int i=2; int x=0;
 do{
     j=1;
     while (j <= i){
         j=j*2;
         x++;
     }
     i++;
  }while (i<=n);
}
```

$O(1)$ · $O(\log_2 i)$ · $O(\log_2 n!)$

2.- Considerar el siguiente segmento de código con el que se pretende buscar un entero ***x*** *en una lista de enteros L de tamaño n (el bucle* ***for*** *se ejecuta* ***n veces****):*

```
void eliminar (Lista L, int x)
   {
      int aux, p;
      for (p=primero(L); p!=fin(L);)
          {
             aux=elemento (p,L);
             if (aux==x)
                borrar (p,L);
             else p++;
          }
   }
```

a) La eficiencia es $O(n^3)$.

Creamos var. auxiliar y la igualamos fuera del for a fin(L) y luego la comparamos en la condición del for.

b) La eficiencia es $O(n^2)$

Lo mejoraría igual que en el apartado anterior.

c) La eficiencia es $O(n)$

Si fuera un array ordenado.