Practica 2 (II / 2008)
Taller de Programación INF-143
Paralelos: A, B, C y D

HALLAR LA PRECONDICIÓN, POSTCONDICIÓN E INVARIANTE DE LOS SIGUIENTES CÓDIGOS. TAMBIÉN IMPLEMENTAR JML Y ASSERTS

1.-
```
static long suma(int n) {
  long s = 0;
  for( int i = 1; i <= n; i++ )
    s = s + i;
  return s;
}
```
2.-
```
static int mult_sum(int x, int y) {
  int m = 0;
  for( int i = 1; i <= x; i++ )
    for( int j = 1; j <= y; j++ )
      m = m + 1;
  return m;
}
```
3.-
```
static long expo(long x, long y) {
  long e = 1;
  while( y>0 ) {
    if ( y%2==1 ) // if( impar(y) )
      e = e * x;
    y = y / 2;
    x = x * x;
  }
  return e;
}
```
4.-
```
int busbin(int a[], int c, int n) {
   int inf, sup, i;
   inf = 1;
   sup = n;
   while (sup >= inf)
   {
      i = (inf + sup) / 2;
      if (a [i] == c)
         return (i);
      else
         if (c < a [i])
            sup = i - 1;
         else
            inf = i + 1;
   }
   return (0);
}
```

5.-
```
int Sumadigitos (int num){
      int s;
      s =num % 10;
       while (num >= 10)
       {
        num = num / 10;
        s=s + (num % 10) ;
       }
       return  s  ;
}
```

6.-
```
int Buscar (int a [], int c){
      int j;
      j=1;
      while ((a[j]<c) and (j<n))
       {
        j =j+1;
       }
      if (a[j]==c)
      return ( j );
      else
      return (0);
}
```

7.-
```
int Euclides (int m, int n){
      int temp;
      while (m > 0) {
        temp =m;
        m=n % m;
        n=temp ;
      }
       Return ( n) ;
}
```

HALLAR LA NOTACIÓN JML PARA LAS SIGUIENTES DESCRIPCIONES DE MÉTODOS (FUNCIONES) COMO SE MUESTRA EN EL EJEMPLO (EJERCICIO 1)

1.- $A[n]$ es el vector ordenado ascendentemente del vector $V[n]$, donde $V[n]$ no será modificado.

```
//@ requires V!=null && V.length>1;
//@ ensures \result.length == V.length;
//@ ensures (\forall int k; 1<k && k<\result.length;
//@                     \result[(int)(k-1)]<=\result[k]);
//@ ensures (\forall int m; 0<=m && m<V.length; V[m]==\old(V[m]) );
static int[] orden(int V[]) {
   int[] A = (int[]) V.clone();
   Arrays.sort( A );
   return A;
}
```

2.- Método que verifica que x aparece en $A[n]$.

```
static boolean aparece(int A[], int x) {...
```

3.- El array $A[n]$ está formado por potencias de 2 (no necesariamente seguidas).

```
static boolean formadoPorPotencias(int A[]) { ...
```

4.- Retorna el mínimo elemento de $A[n]$.

```
static int minimo(int A[]) { ...
```

5.- Retorna el máximo elemento de la sección $A(i..j)$ de $A[n]$.

```
static int maximo_entre(int A[], int i, int j) {...
```

6.- Retorna la suma de enteros de la sección $A(i..j)$ del array de enteros $A[n]$.

```
static int suma_entre(int A[], int i, int j) {...
```

7.- Verifica que $A[n]$ no tiene elementos repetidos.

```
static boolean no_hay_repetidos(int A[]) {...
```

8.- Verifica que x es menor que todos los elementos de $A[n]$.

```
static boolean menor_que_todos(int x, int A[]) {...
```

9.- Dado un $V[n]$. Retornar un $A[n]$ que será igual a $(v_{n-1}, v_0, v_1, \ldots, v_{n-2})$.

```
static int[] mover_der(int V[]) {...
```

10.- Verifica que $A[n]$ tienen al menos una par de elementos consecutivos diferentes.

```
static boolean existen_diferentes(int A[]) {...
```

11.- Retornar v, que es el número de veces que aparece x en $A[n]$.

```
static int contar(int A[], int x) {...
```

Nota.- $A[n] = A(0..n-1)$ y $A(i..j) \in A(0..n-1) \Rightarrow 0 \leq i \leq j < n$

Nota.- Especificar cuando el vector original no es modificado.

Pagina WEB: http://training.team-sim.info/