Problemas para la sesión de tutoría

Problema 1 Índice igual a valor

Sea v un vector de naturales con capacidad N e inicializado entre ini y sup (el vector puede ser vacío). Diseñar una función *igual_indice_valor* recursiva que devuelva un natural que indique el índice de $v[ini..fi]$ que almacena como valor el propio índice. Si tal índice no existiera, devolvería 0.
Ejemplos:

- **Ejemplo 1**

  Sea N=12, ini=1, fi=12

  Vector v:

1	2	3	4	5	6	7	8	9	10	11	12
10	10	3	5	1	10	10	0	0	0	0	0

  igual_indice_valor devolvería 5 puesto que existe un índice entre 1 y 12 (concretamente, 3) tal que $v[3] = 3$.

- **Ejemplo 2**

  Sea N=12, ini=5, fi=9

  Vector v:

1	2	3	4	5	6	7	8	9	10	11	12
1	1	3	5	1	3	1	1	2	10	11	12

  igual_indice_valor devolvería 0 puesto que **NO existe** ningún índice i entre 5 y 9 tal que $v[i] = i$.

- **Ejemplo 3**

  Sea N=12, ini=5, fi=4

  Vector v:

1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	6	7	8	9	10	11	12

  igual_indice_valor devolvería 0 puesto que **NO existe** ningún índice i entre 5 y 4 tal que $v[i] = i$ (caso de vector vacío).

Se pide:

1. Especificar la función *igual_indice_valor*. En particular, proponer una cabecera para la misma con los parámetros adecuados, una precondición y una postcondición.

2. Implementar en pseudocódigo la acción *igual_indice_valor*.

Problema 2 El màximo de un vector

Sea v un vector de naturales con capacidad N e inicializado entre *ini* y *sup* (el vector puede ser vacío). Diseñar una acción *maximo* recursiva con un parámetro de salida que devuelva el valor màximo entre esos dos índices.
Se pide:

1. Especificar la función *maximo*. En particular, proponer una cabecera para la misma con los parámetros adecuados, una precondición y una postcondición.

2. Implementar en pseudocódigo la acción *igual_indice_valor*.

Problema 3 La partición de igual suma

Sea v un vector de naturales con capacidad N e inicializado entre *ini* y *sup* (el vector puede ser vacío). Diseñar una acción *igual_suma* recursiva con un parámetro de salida booleano que indique si es posible dividir el vector en dos partes de elementos consecutivos tales que entre las dos cubran la totalidad del vector y sumen lo mismo (una o incluso, las dos partes pueden ser vacías).
Ejemplos:

- **Ejemplo 1**

  Sea N=12, ini=1, fi=12

  Vector v:

1	2	3	4	5	6	7	8	9	10	11	12
1	10	3	5	1	10	10	0	0	0	0	0

  v[1..12] puede dividirse en dos partes v[1..5] y v[6..12] tales que ambas suman 20. Por tanto el parámetro de salida valdría $b = cierto$.

- **Ejemplo 2**

  Sea N=12, ini=5, fi=9

  Vector v:

1	2	3	4	5	6	7	8	9	10	11	12
1	1	3	5	1	3	1	1	2	1	1	1

  v[5..9] puede dividirse en dos partes v[5..6] y v[7..9] tales que ambas suman 4. Por tanto el parámetro de salida valdría $b = cierto$.

- **Ejemplo 3**

  Sea N=12, ini=1, fi=1

  Vector v:

1	2	3	4	5	6	7	8	9	10	11	12
0	1	3	5	1	3	1	1	2	1	1	1

$v[1..1]$ puede dividirse en dos partes $\emptyset$ y v[1..1] tales que ambas suman 0 (entendemos, por tanto, que una parte vacía suma 0). Por tanto el parámetro de salida valdría $b = cierto$.

- **Ejemplo 4**

  Sea N=12, ini=2, fi=1

  Vector v:

1	2	3	4	5	6	7	8	9	10	11	12
6	1	3	5	1	3	1	1	2	1	1	1

  $v[1..0]$ puede dividirse en dos partes $\emptyset$ y $\emptyset$ tales que ambas suman 0. Por tanto el parámetro de salida valdría $b = cierto$.

- **Ejemplo 5**

  Sea N=12, ini=1, fi=5

  Vector v:

1	2	3	4	5	6	7	8	9	10	11	12
1	1	3	5	1	3	1	1	2	1	1	1

  $v[1..5]$ **NO** puede dividirse en dos partes tales que ambas sumen lo mismo. Por tanto el parámetro de salida valdría $b = falso$.

  Notad que $v[1..3]$ suma lo mismo que $v[4]$. Pero esta partición no es adecuada ya que no contiene todos los elementos de $v[1..5]$ (no contiene $v[5]$).

Se pide:

1. Especificar la acción *igual_suma*. En particular, proponed una cabecera para la misma con los parámetros adecuados, una precondición y una postcondición. Podéis proponerlas en lenguaje natural.
2. Implementar en pseudocódigo la acción *igual_suma*.

Proposta de solució

```
#include <iostream.h>
#include <vector.h>

/*********************************************
igual_suma(v,ini,fi,dif,sum,b)

Pre:0<=ini<=fi+1<=N i dif=D

Post: b=cert si v[ini..fi] es pot descomposar en dues parts v[ini..k] i
      v[k+1..sup] tal que suma(v[k+1..sup]) - suma(v[ini..k]) = dif.
      sum=suma(v[ini..sup]) per algun k t.q. ini-1 <= k <= fi

      Nota: k no es retornat per igual_suma (no es demana).
*********************************************/

void igual_suma(vector<int>& v, int ini, int fi, int dif, int& sum, bool& b)
{

  if (ini==fi+1) {sum=0; b=(dif==0);}
  else {
   igual_suma(v,ini+1,fi,dif+v[ini],sum,b);
sum=sum+v[ini];
b=(b || (dif==sum));
       }
}

/*********************************************
i=igual_index_valor (v,ini,fi)

Pre:0<=ini<=fi+1<=N

Post: i=-1 si no existeix cap index k entre ini..fi t.q. v[k]=k.
      ini<=i<=fi  sii  v[i]=i

*********************************************/


int igual_index_valor (vector<int>& v, int ini, int fi)
{
 if (ini==fi+1) return -1;
 else if (v[ini]==ini) return ini;
 else return igual_index_valor(v,ini+1,fi);
}
```

```
/*********************************************
maxim(v,ini,fi,max)

Pre:0<=ini<=fi+1<=N i  max=M

Post: max es el maxim entre M i el mes gran de v[ini..fi].
*********************************************/

void maxim(vector<int>& v, int ini, int fi, int& max)
{
 if (ini==fi+1) ;
 else {if (v[ini]>max) {max=v[ini];}
       maxim(v,ini+1,fi,max);
      }
}




void main()
{
 int max, ini, fi, i, sum;
 bool b;
 vector<int> v(20);

 cout<<"max=\n";
 cin>>max;

 cout<<"ini=\n";
 cin>>ini;

 cout<<"fi=\n";
 cin>>fi;

 for(i=0;i<max;i++){

  cin>>v[i];

 }

/**********
 igual_suma(v,ini,fi,0,sum,b);

 if (b) {cout<<"particio possible\n"<<"suma="<<sum<<endl;}
```

```
 else {cout<<"particio NO possible\n"<<"suma="<<sum<<endl;}


 i=igual_index_valor(v,ini,fi);
 cout<<"index sortida="<<i<<endl;
**********/

 i=0;
 maxim(v,ini,fi, i);
 cout<<"maxim="<<i<<endl;


}
```