ESTRUCTURAS DE DATOS

PROBLEMAS EVALUACIÓN CONTINUA

Jose Manuel Martínez de la Insua.

PROBLEMAS DE PUNTEROS

Problema 1.1
Escriba un trozo de código donde se declara un vector de 100 enteros y, mediante un
bucle, se asigne el valor 1 a todas las posiciones, sin usar el \operador []".

```
int v[100];
int * ptrv = v;
for(int i = 0; i < 100; i++){
        *ptrv = 1;
        ptrv++;
}
```

Problema 1.2
Declare una variable numeros como una vector de 1000 enteros. Escriba un trozo de código que recorra el vector y modifique todos los enteros negativos cambiándolos de signo. No se debe usar el operador '[]', es decir, se deberá usar aritmética de punteros. El bucle se controlara mediante un contador entero.

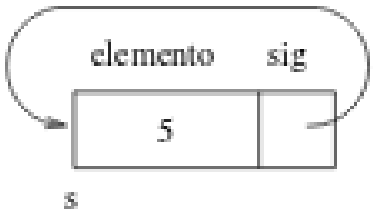
```
int v[1000];
int * ptrv = v;
for(int i = 0; i < 1000; i++){
        if(i%2==0) *ptrv = i;
        else *ptrv = (-1)*i;
        ptrv++;
}
ptrv = v;
for(int i = 0; i < 1000; i++){
        if(*ptrv < 0) *ptrv *= -1;
        ptrv++;
}
```

Problema 1.3
Modifique el código del problema anterior para controlar el final del bucle con un puntero a la posición siguiente a la ultima.

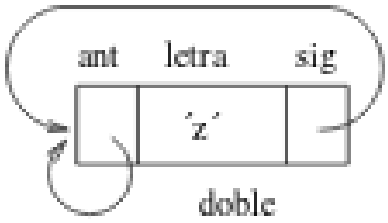
```
int numeros[1000];
int * ptrv = numeros;
int *fin = ptrv+1000;
int i = 0;
for(; ptrv < fin; ptrv++){
        if(i%2 == 0)*ptrv = i;
        else *ptrv = i*(-1);
        i++;
}

ptrv = numeros;
for(; ptrv < fin; ptrv++)
        if(*ptrv < 0) *ptrv *= -1;
```

Problema 1.4
Considere la gura 1. Se presentan gráficamente un conjunto de estructuras de datos.
Se puede observar que las matrices se representan indicando los índices y las estructuras indicando los
nombres de los campos. Escriba los trozos de código que corresponden a su creación. Nota: No se debe
usar memoria dinámica (para cada caso se incluye el nombre de las variables necesarias).

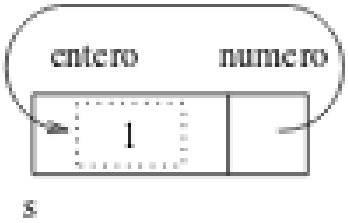
```
typedef struct N {
        int elemento;
        struct N* sig;
} Nodo;

typedef Nodo* ptrNodo;

ptrNodo s;
s=(ptrNodo)malloc(sizeof(Nodo));
s->elemento = 1;
s->sig = s;
```

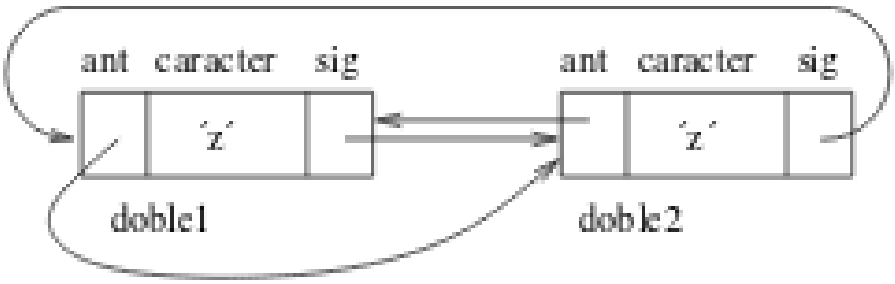


```
typedef struct N {
        struct N* ant;
        char letra;
        struct N* sig;
} Nodo;

typedef Nodo* ptrNodo;

ptrNodo doble;
doble=(ptrNodo)malloc(sizeof(Nodo));
doble->letra = 'z';
doble->sig = doble;
doble->ant = doble;
```

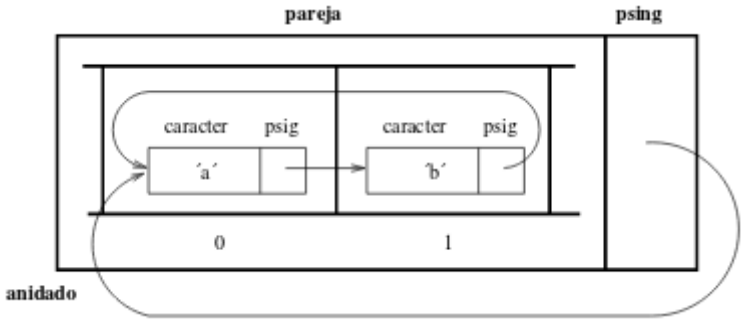


```
typedef struct N {
        int entero;
        int* sig;
} Nodo;

typedef Nodo* ptrNodo;

ptrNodo s;
s=(ptrNodo)malloc(sizeof(Nodo));
s->entero = 1;
s->sig = &(s->entero);
```





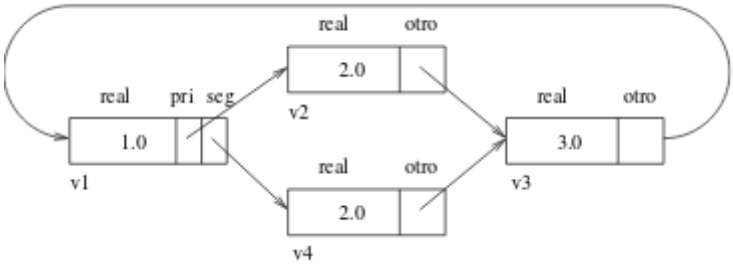

```
typedef struct N {
        struct N* ant;
        char caracter;
        struct N* sig;
} Nodo;

typedef Nodo* ptrNodo;


ptrNodo doble1,doble2;
doble1 = (ptrNodo)malloc(sizeof(Nodo));
doble2 = (ptrNodo)malloc(sizeof(Nodo));

doble1->caracter = 'z';
doble2->caracter = 'z';

doble1->sig = doble2;
doble2->sig = doble1;

doble1->ant = doble2;
doble2->ant = doble1;
```




```
typedef struct N {
        char caracter;
        struct N* psig;
} Nodo;
typedef Nodo* ptrNodo;

typedef struct A {
        vector <ptrNodo> pareja;
        ptrNodo psing;
} Anidado;
typedef Anidado* ptrAnidado;
```

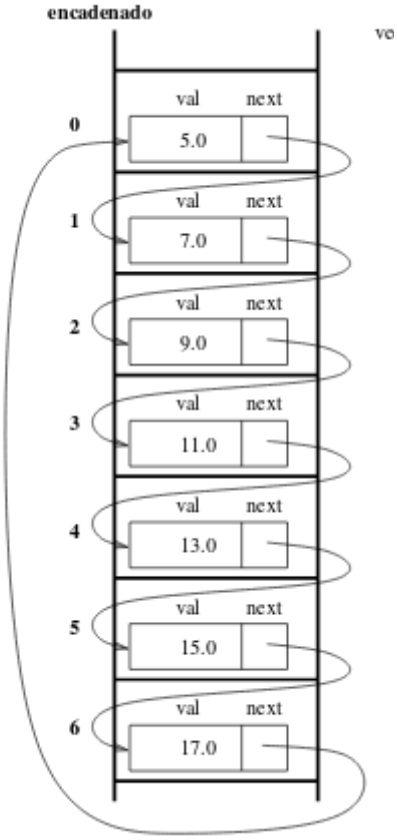
```
// Reserva de memoria
ptrNodo nodo0,nodo1;
nodo0 = (ptrNodo)malloc(sizeof(Nodo));
nodo1 = (ptrNodo)malloc(sizeof(Nodo));

ptrAnidado anidado;
anidado = (ptrAnidado)malloc(sizeof(Nodo));

// Asignación de variables
nodo0->caracter = 'a';
nodo1->caracter = 'b';

nodo0->psig = nodo1;
nodo1->psig = nodo0;

vector <ptrNodo> par;
par.push_back(nodo0);
par.push_back(nodo1);

anidado->pareja = par;
anidado->psing = nodo0;
```

```
typedef struct N {
        float real;
        vector <struct N*>
enlaces;
} Nodo;
typedef Nodo* ptrNodo;
```

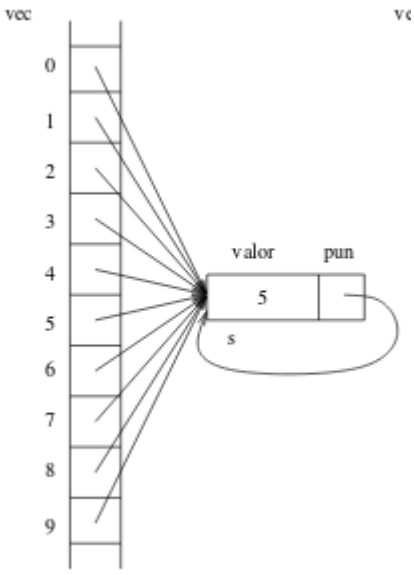



```
// Reserva de memoria
ptrNodo v1,v2,v3,v4,pri,seg,otro1,otro2;

v1 = (ptrNodo)malloc(sizeof(Nodo));
v2 = (ptrNodo)malloc(sizeof(Nodo));
v3 = (ptrNodo)malloc(sizeof(Nodo));
v4 = (ptrNodo)malloc(sizeof(Nodo));
pri = (ptrNodo)malloc(sizeof(Nodo));
seg = (ptrNodo)malloc(sizeof(Nodo));
otro1 = (ptrNodo)malloc(sizeof(Nodo));
otro2 = (ptrNodo)malloc(sizeof(Nodo));

// Asignación de variables
v1->real = 1.0f;
(v1->enlaces).push_back(pri);
(v1->enlaces).push_back(seg);
v2->real = 2.0f;
(v2->enlaces).push_back(otro1);
v3->real = 3.0f;
(v3->enlaces).push_back(otro2);
v4->real = 2.0f;
(v4->enlaces).push_back(otro1);

(v1->enlaces)[0] = v2;
(v1->enlaces)[1] = v4;

(v2->enlaces)[0] = v3;
(v3->enlaces)[0] = v1;
(v4->enlaces)[0] = v3;
```

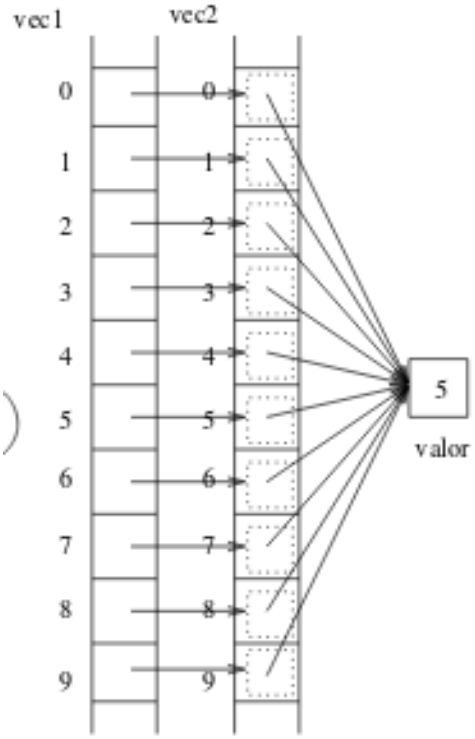
```
typedef struct N {
        float val;
        struct N* next;
} Nodo;
typedef Nodo* ptrNodo;


vector <ptrNodo> encadenado;
int count = 0;
for (float k = 5.0f; k < 18.0f; k+=2.0f){
        ptrNodo temp =
(ptrNodo)malloc(sizeof(Nodo));
        temp->val = k;
        encadenado.push_back(temp);
        encadenado[count]->next =
encadenado[(count+1)%7];
        count++;
}
```



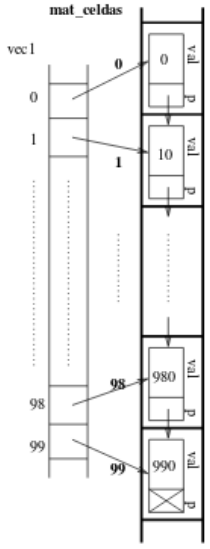
```
typedef struct N {
        int valor;
        struct N* pun;
} Nodo;
typedef Nodo* ptrNodo;

ptrNodo s = (ptrNodo)malloc(sizeof(Nodo));
s->valor = 5;
s->pun = s;

vector <ptrNodo> vec;
for (int k = 0; k < 10 ; k++ ){
        ptrNodo temp = (ptrNodo)malloc(sizeof(Nodo));
        temp = s;
        vec.push_back(temp);
}
```



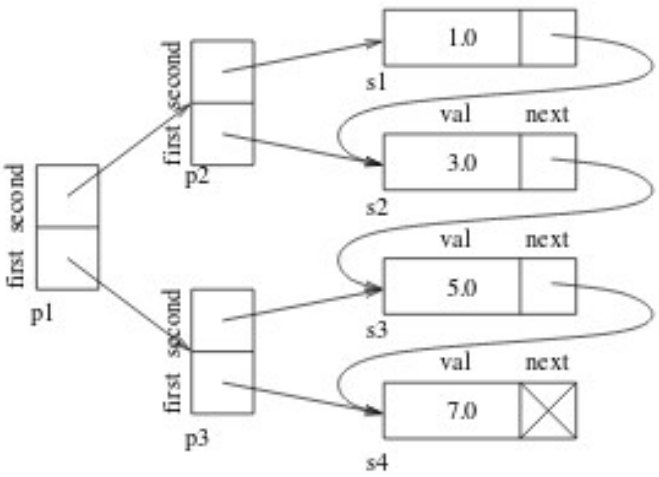
```
typedef int* ptr_int;
typedef ptr_int* ptr2_int;


int valor = 5;

vector <ptr2_int> vec1;
vector <ptr_int> vec2;

for (int k = 0; k < 10 ; k++ ){
        ptr_int pi = &valor;
        ptr2_int pi2 = &pi;
        vec1.push_back(pi2);
        vec2.push_back(pi);
}
```



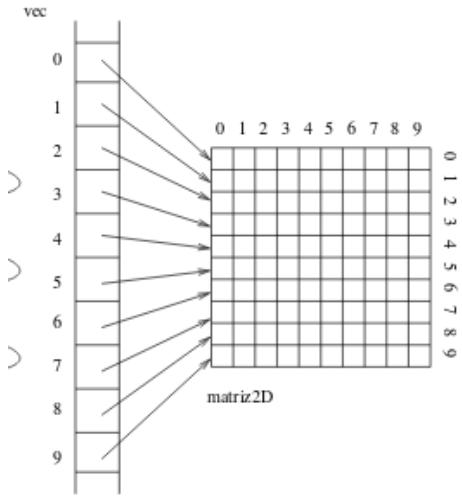
```
typedef struct N {
        int valor;
        struct N* p;
} Nodo;
typedef Nodo* ptrNodo;

vector <ptrNodo> vec1;
vector <Nodo> mat_celdas;

for (int k = 0; k < 100; k++){
        Nodo n;
        n.valor = k*10;
        mat_celdas.push_back(n);

        ptrNodo temp = (ptrNodo) malloc( sizeof(Nodo) );
        temp = &mat_celdas.at(k);
        vec1.push_back(temp);
}

for (int k = 0; k < 99 ; k++)
        (mat_celdas.at(k)).p = &mat_celdas.at(k+1);
```



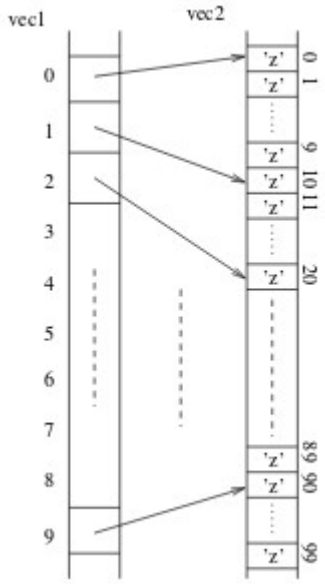
```
typedef struct H {
        float val;
        struct H* next;
} Hoja;
typedef Hoja* ptrHoja;

typedef struct R {
        ptrHoja first;
        ptrHoja second;
} Rama;
typedef Rama* ptrRama;

typedef struct P {
        ptrRama first;
        ptrRama second;
} Padre;
typedef Padre* ptrPadre;

Hoja s1,s2,s3,s4;
s1.val = 1.0f;
s2.val = 3.0f;
s3.val = 5.0f;
s4.val = 7.0f;

s1.next = &s2;
s2.next = &s3;
s3.next = &s4;
s4.next = NULL;

Rama p2,p3;
p2.first = &s2;  p2.second = &s1;
p3.first = &s4;  p3.second = &s3;

ptrPadre p1 = (ptrPadre) malloc (sizeof(Padre));
p1->first = &p3; p1->second = &p2;
```



```
typedef int* ptrInt;


int k = 0;
const int TAM = 10;
int matriz2D[TAM][TAM];

for (int i = 0; i < TAM; i++){
        for (int j = 0; j < TAM ; j++){
                matriz2D [i][j] = k;
                k++;
        }
}

vector <ptrInt> vec;

for (int w = 0; w < TAM ; w++){
        ptrInt temp = matriz2D[w];
        vec.push_back(temp);
}
```



```
typedef char* ptrChar;

const int INDEX = 10;
const int TAM = 100;

ptrChar vec1[INDEX];
char vec2[TAM];

for (int k = 0; k < TAM; k++)
        vec2[k] = 'z';

for (int k = 0; k < INDEX ; k++){
        ptrChar temp = &(vec2[k*INDEX]);
        vec1[k] = temp;
}
```



Problema 1.5
Supongamos tres vectores v1,v2,res de valores reales. En v1,v2 se almacenan, respectiva-mente,n,m valores ordenados de menor a mayor. Escribir un trozo de código para mezclar, de manera ordenada, los valores en el vector res que tiene capacidad para almacenas al menos n+m valores. No se debe usar el operador '[]', es decir, se debe usar aritmética de punteros.

```
typedef int * P;

int main(){

	const int TAM = 10;
	int v1[TAM];
	int v2[TAM];
	vector <int> res;

	for (int k = 0 ; k < TAM ; k++) {
		v1[k] = 5*k+1;
		v2[k] = 2*k;
	}

	P p1 = v1; P p1_fin = p1 + TAM;
	P p2 = v2; P p2_fin = p2 + TAM;

	while (!(p1 == p1_fin && p2 == p2_fin)){
		if (*p1 < *p2){
			res.push_back(*p1);
			p1++;
		}else{
			res.push_back(*p2);
			p2++;
		}

	}

	for(vector<int>::iterator it = res.begin(); it != res.end(); ++it)
		cout << *it << endl;
}
```