EXAMEN DE SISTEMAS INFORMÁTICOS INDUSTRIALES
(SOLUCIÓN) (TEORÍA)

La duración del examen es de 2 horas. **JUNIO 2017**

1. Indicar si las siguientes afirmaciones son Verdaderas o Falsas

(2.4 puntos)

F	(a) Si se define la variable cadena como **char cadena[10];**, la instrucción siguiente es correcta **scanf("%s",&cadena);**
F	(b) Si dos cadenas de caracteres tienen la misma dimensión puede asignarse el contenido de la **cadenaB** a la **cadenaA** de la siguiente forma: **cadenaA=cadenaB;**
F	(c) Sea el vector de enteros **int v[5]**. Para almacenar en la última posición del vector el valor 100, es correcto poner **v[5]=100;**
V	(d) La función scanf no puede utilizarse para leer una cadena de caracteres por teclado si ésta contiene espacios en blanco.
V	(e) Las variables miembro de una clase base en C++ declaradas como **protected** pueden ser accedidas desde una clase derivada de la clase base.
V	(f) C++ permite la herencia múltiple, es decir, una clase derivada puede heredar de más de una clase base.
V	(g) Una función global amiga de una clase (**friend**) puede acceder a los miembros de la clase como si fuera un método de la clase.
V	(h) Por defecto, la función **recv** para lectura de un socket quedará a la espera de recibir datos de forma indefinida.
V	(i) En una comunicación asíncrona el transmisor puede enviar datos en cualquier momento.
F	(j) En una transmisión síncrona se requiere de bits de inicio y fin para identificar cada byte de datos.
F	(k) Para que un cliente se comunique con un servidor mediante sockets, el cliente no necesita conocer el puerto del servidor.
F	(l) La ventaja de utilizar el protocolo UDP es que es un protocolo orientado a la conexión y con control de errores en la transmisión.

2. En este ejercicio se pide diseñar un programa que sea capaz de leer información de un fichero llamado *información.txt*. Se leerá línea a línea y mediante una función se almacenará la información en un vector de estructuras. Posteriormente se mostrarán las palabras de la frase leída con un número de caracteres igual o superior al valor leído al final de la cadena. Un ejemplo de fichero puede verse tras el enunciado. En el fichero habrá siempre 5 líneas. Se pide:

(a) En primer lugar se pide completar el main que se da de base. Primero se definirá la estructura a utilizar con un campo para almacenar la cadena de caracteres (máximo 100 caracteres) y otro campo tipo *int* para el número. Después se declararán las variables necesarias y se realizará reserva dinámica de memoria para el vector de estructuras (para 5). Se realizará el bucle que lee línea a línea el fichero y llame a la funcion1 proporcionándole la información necesaria. Y finalmente se cerrará adecuadamente. *(1.4 puntos)*

(b) En este apartado se deberá completar la *funcion1* de la cual se da el prototipo. Esta función recibe el vector de estructuras completo, la posición donde hay que escribir la nueva línea y la línea leída. Se deberá guardar en la estructura la frase por un lado y el número por otro en el campo que corresponda y se mostrará por pantalla el resultado tal y como se muestra en el ejemplo de ejecución. *(1.4 puntos)*

(c) En este apartado se deberá completar la *funcion2* de la cual se da el prototipo. Esta función recibe una posición del vector de estructuras y deberá mostrar de la frase almacenada únicamente las palabras que tengan un número de caracteres igual o mayor al número indicado en la estructura. Puede verse un ejemplo de ejecución de lo que mostrará esta función.*(1.6 puntos)*

información.txt

```
Frase numero uno 5
Otra frase mas a mostrar 4
Y aqui ponemos otra mas 3
Seis cinco cuatro tres dos uno cero 4
esta es una frase con la palabra mas larga 6
```

Ejemplo de ejecución donde puede verse lo que muestra *funcion1* y *funcion2*:

```
Salida de la Funcion 1:
lista[0].frase=>Frase numero uno
lista[0].numero=>5
lista[1].frase=>Otra frase mas a mostrar
lista[1].numero=>4
lista[2].frase=>Y aqui ponemos otra mas
lista[2].numero=>3
lista[3].frase=>Seis cinco cuatro tres dos uno cero
lista[3].numero=>4
lista[4].frase=>esta es una frase con la palabra mas larga
lista[4].numero=>6

Salida de la Funcion 2:
Palabras de al menos 5 caracteres -> Frase numero
Palabras de al menos 4 caracteres -> Otra frase mostrar
Palabras de al menos 3 caracteres -> aqui ponemos otra mas
Palabras de al menos 4 caracteres -> Seis cinco cuatro tres cero
Palabras de al menos 6 caracteres -> palabra
Presione una tecla para continuar . . .
```

Código dado de base para completar:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define total_lineas 5
// Definición de la estructura




void funcion1(info *lista,char linea_leida[],int posicion);
void funcion2(info lista);

int main(int argc, char *argv[])
{
    int i, contador=0;
    FILE *pFich;
    info *lista;
    char linea[100];

// Reserva dinámica y apertura del fichero




     printf("\nSalida de la Funcion 1:\n");
     // Bucle de lectura. En cada iteración del bucle se llamará
     // a la funcion1 con la línea leída



             funcion1(lista,linea,contador);



// Cierre

// Uso de la funcion2
    printf("\nSalida de la Funcion 2:\n");
    for (i=0;i<total_lineas;i++){
        // Con la estructura rellenada, se llama a la función 2
        // en el bucle. En cada iteración se le pasa una de las
        // posiciones del vector de estructuras. Y la función 2
        // muestra la información como se indica en el enunciado
        funcion2(lista[i]);
    }
    system("PAUSE");
    return 0;
}
```

SOLUCIÓN:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define total_lineas 5

// Definición de la estructura
typedef struct info{
       char frase[100];
       int numero;
}info;

void funcion1(info *lista,char linea_leida[],int posicion);
void funcion2(info lista);

int main(int argc, char *argv[])
{
    int i, contador=0;
    FILE *pFich;
    info *lista;
    char linea[100];

    lista=(info*)malloc(total_lineas*sizeof(info));

       pFich = fopen("informacion.txt", "r");

       printf("\nSalida de la Funcion 1:\n");

       while (!feof(pFich))
       {
         fgets(linea, 100, pFich);
         // Cada vez que se lea una línea del fichero se llama a la función 1
         // En esta función se recibirá la estructura completa, la línea leída
         // y la posición a rellenar de la estructura
         funcion1(lista,linea,contador);
         contador++;
       }
       fclose(pFich);

    printf("\nSalida de la Funcion 2:\n");
    for (i=0;i<total_lineas;i++){
        // Con la estructura rellenada, se llama a la función 2 en el bucle
        // En cada iteración se le pasa una de las posiciones
        // del vector de estructuras. Y la función 2 muestra la información
        // como se indica en el enunciado
        funcion2(lista[i]);
    }
    system("PAUSE");
    return 0;
}
```

```
// Función que recibe la esctrcutura completa, la posición en la que
// toca escribir y la línea leida. Deberá guardar en la estructura
// la frase por un lado y el número por otro
void funcion1(info *lista,char linea_leida[],int posicion)
{
    int i;
    // Buscar el último espacio para localizar el número.
    for (i=strlen(linea_leida)-1;linea_leida[i]!=' ';i--);
    // Copia la frase a la estructura
    strcpy(lista[posicion].frase,linea_leida);
    // Cierra la cadena en el espacio.
    lista[posicion].frase[i]='\0';
    // Convierte el número a entero y lo guyarda en la estructura
    lista[posicion].numero=atoi(&linea_leida[i+1]);
    // Muestra la información que acaba de guardar en la estructura
    printf("lista[%d].frase=>%s\n",posicion,lista[posicion].frase);
    printf("lista[%d].numero=>%d\n",posicion,lista[posicion].numero);
}
```

```
// Función que recibe una posición del vector de listas, es decir,
// una estrucutura. Mostrará sólo las palabras que tengan un número
// de caracteres mayor o igual al indicado en la estrucutra
void funcion2(info lista)
{
    int i,j,k,cont;

    printf("Palabras de al menos %d caracteres -> ",lista.numero);
    cont=-1;
    // Recorro la cadena
    for (i=0;i<=strlen(lista.frase);i++)
    {
        // Si encuentro un espacio o el final de la cadena
        if (lista.frase[i]==' ' | i==strlen(lista.frase))
        {
            // Compruebo la longitud y si cumple la condicón
            if (i-cont-1>=lista.numero)
            {
                // Muestro la palabra por pantalla
                for (k=cont+1;k<i;k++)
                    printf("%c",lista.frase[k]);
                printf(" ");
            }
            cont=i; // Y continuo por el final de dicha palabra
        }
    }
    printf("\n");
}
```

3. Dado un código en C++ se pedirá en primer lugar obtener la salida al ejecutar el main. A continuación se añadirá un constructor nuevo parametrizado y finalmente se sobrecargará el operador = según las indicaciones. Se pide:

(a) Mostrar la información que sale por pantalla tras ejecutar el main que se muestra tras el enunciado y que utiliza la clase base y la clase derivada vector. *(1.0 puntos)*

```
#include "iostream"
#include <stdlib.h>
#include <math.h>

using namespace std;

class base{
    protected:
        int dim;
        int h;
    public:
        base(int d = 3);
        ~base(void);
        void funcion1(int &y);
};

base::base(int d){
    cout << "Constructor base" << endl;
    dim=d;
    h=dim*3;
    cout << "Valor de h = " << h << endl;
}

base::~base(void){
    cout << "Destructor base" << endl;
}


void base::funcion1(int  &y){
    cout << "base::funcion1 " << y << endl;
    y = 0;
    cout << "dim: "<< dim;
    cout << ", h: "<<h<< ", y: "<<y<<endl;
}
```

```
class vector : public base{
    private:
        int *vec;
    public:
        vector(int d = 2);
        ~vector(void);
        void mostrar_vector(void);
        void funcion1(int &f);
};

vector::vector(int d) : base(d){
    cout << "Constructor vector " << endl;
    h=pow(h,2);
    cout << "Valor de h = " << h << endl;
    vec=new int[dim];
    for (int i=0;i<dim;i++)
        vec[i]=dim;
}

void vector::mostrar_vector(void){
    for (int i=0;i<dim;i++)
        cout << "vector["<<i<<"]=>"
        cout <<vec[i]<<endl;
}

vector::~vector(void){
    cout << "Destructor vector" << endl;
    delete []vec;
}

void vector::funcion1(int &b){
    cout << "vector::funcion1" << endl;
    b=dim;
    cout << "dim: "<< dim;
    cout <<", h: "<<h<< ", b: "<<b<<endl;
}
```

```
int main(void){
    int n=5;
    vector v;
    v.mostrar_vector();
    v.funcion1(n);
    cout << endl << "n: " << n << endl;
    system("PAUSE");
    return 0;
}
```

(b) Añadir a la clase derivada *vector* un constructor parametrizado que permita indicar tanto la dimensión del vector como recibir como parámetro un array con los valores enteros para inicializar. El constructor deberá realizar todos los pasos necesarios para definir completamente el objeto y hará uso del constructor de la clase base. *(1.0 puntos)*

(c) Para finalizar deberéis realizar la sobrecarga del *operador* = para la clase *vector*. El objetivo es que cuando dos objetos de la clase *vector* se igualen, al objeto a la izquierda del igual se le deberá añadir una posición extra que tenga como valor la suma de los valores del vector de la derecha. *(1.2 puntos)*

Ejemplo: Sí tenemos un objeto vec_a con dimensión 3 y valores del vector [3 4 1] y otro objeto vec_b con dimensión 2 y valores del vector [5 7]. Al realizar la operación vec_a=vec_b; el resultado de los vectores deberá ser el siguiente: vector de vec_a [3 4 1 12] y vector de vec_b se quedará igual [5 7];

SOLUCIÓN apartado a:

```
Constructor base
Valor de h = 6
Constructor vector
Valor de h = 36
vector[0]=>2
vector[1]=>2

vector::funcion1
dim: 2, h: 36, b: 2
n: 2

Destructor vector
Destructor base
```

SOLUCIÓN apartado b:

```
vector::vector(int d,int v[]) : base(d){
    cout << "Constructor vector " << endl;
    vec=new int[dim];
    for (int i=0;i<dim;i++)
        vec[i]=v[i];
}
```

SOLUCIÓN apartado c:

```
vector& vector::operator =(vector &der)
{
    int *temp=new int[dim+1];
    for (int i=0;i<dim;i++){
        temp[i]=vec[i];
    }
    temp[dim]=0;
     for (int i=0;i<der.dim;i++){
        temp[dim]+=der.vec[i];
    }
    dim++;
    delete vec;
    vec=temp;
    return(*this);
}
```