EXAMEN DE SISTEMAS INFORMÁTICOS INDUSTRIALES
(SOLUCIÓN) (TEORÍA)

SEPTIEMBRE 2015

1. Indicar si las siguientes afirmaciones son verdaderas o falsas:

(a) En C++, una clase derivada puede acceder a los miembros privados de la clase base. **F**
(b) En C++ se pueden implementar diferentes destructores para la misma clase. **F**
(c) En C++, una variable miembro estática es común para todos los objetos de la clase (no hay una variable estática diferente para cada objeto de la clase, sino que todos los objetos comparten la misma variable estática). **V**
(d) En C es posible actualizar el indicador de posición de fichero al inicio del fichero. **V**
(e) Para aplicar el algoritmo de búsqueda binaria sobre un vector, no es necesario que esté ordenado. **F**
(f) Para declarar e inicializar un entero `j` con el valor 10 en C++ se puede hacer `int j=10;` o `int j(10);` **V**
(g) Dada la declaración `int a[10];` para asignar al elemento situado en la posición 1 del vector el valor 5, se puede utilizar `a[1]=5;` o `*(a+1)=5;` **V**
(h) Para implementar una arquitectura cliente/servidor en C ó C++ sólo puede utilizarse la librería de programación de sockets. **F**

(2 puntos)

2. Considérese que se ha creado una clase denominada *CPila* para trabajar con pilas de caracteres. La pila se almacena en un vector de caracteres que se reserva de forma dinámica según el tamaño de la variable *tam*, cuyo valor se asigna en el constructor. La declaración de esta clase es la siguiente:

```
const int MAX=100;
class CPila
{
private:
      int tam;  // tamaño de la pila
      char *ppila;  // puntero a la pila
      int top;  // ultima posicion ocupada +1
public:
      CPila(const int  t=MAX); // constructor
      CPila(const  CPila &pila_orig); // constructor copia
      ~CPila(void); // destructor
      // operador asignación
      CPila& operator  = (const  CPila & pila_orig);

      void  Meter(const char  c); // introduce un caracter
                                  // en la pila
      char  Sacar(void); // extrae un caracter de la pila
      int  Vacia(void); // Comprueba si la pila esta vacia
      int  Llena(void); // Comprueba si la pila esta llena
};
```

donde:

- La variable miembro *tam* indica el tamaño de la pila, *ppila* es un vector para almacenar los caracteres de la pila y *top* indica la posición donde hay que introducir cada carácter (restándole uno indicará de donde hay que sacar el carácter de la pila).
- El constructor a partir del parámetro *t* reserva la memoria del vector *ppila* de forma dinámica a partir del tamaño indicado en *t*, inicializa *tam* al valor de *t* y *top* al valor 0.
- El operador asignación permite copiar pilas de igual tamaño.
- El método *Meter* introduce un carácter en el vector *ppila* incrementando en uno el valor de *top*. No comprueba el tamaño de la pila.
- El método *Sacar* decrementa en uno el valor de *top* y devuelve el carácter que se encuentra en esa posición en el vector *ppila.* No comprueba el tamaño de la pila.
- El método *Vacia* devuelve 1 si la pila está vacía y 0 si no. El método *Llena* devuelve 1 si la pila está llena y 0 si no.

Considerando que puede hacerse uso de cualquier método de la clase, se pide:

(a) Definir el operador + como función global amiga de la clase para crear una nueva pila a partir de dos pilas. En este caso dentro de la declaración de la clase habría que incluir:

```
friend CPila operator+(const CPila &p1, const CPila &p2);
```

Por ejemplo, si:

- *pila1* tiene *tam*=3, *ppila*={'a', , } y *top*=1, y
- *pila2* tiene *tam*=4, *ppila*={'b','c', , } y *top*=2,

al hacer *pila1*+*pila2* se obtendrá:

- una pila con *tam*=7, *ppila*={'a','b','c', , , , } y *top*=3.

(2 puntos)

(a) Definir el operador << como función amiga de la clase de forma que permita mostrar por pantalla el contenido de una pila. En este caso dentro de la declaración de la clase habría que incluir:

```
friend ostream& operator << (ostream &out, CPila &pila);
```

Por ejemplo, si *pila* tiene *tam*=3, *ppila*={'a','b' , } y *top*=2, al hacer *cout*<<*pila* se mostrará por pantalla: *a b*

(1 punto)

Solución:

(a)

```
CPila operator+(const CPila &p1, const CPila &p2)
{
      int i,t=p1.tam+p2.tam;
      CPila suma(t);
```

```
      for(i=0;i<p1.top;i++){
            suma.Meter(p1.ppila[i]);
      }

      for(i=0;i<p2.top;i++){
            suma.Meter(p2.ppila[i]);
      }

      return(suma);
}
```

(b)

```
ostream& operator << (ostream &out, CPila &pila)
{
      for(int i=0;i<pila.top;i++){
            out<<pila.ppila[i]<<" ";
      }

      return out;
}
```

3. El fichero basedatos.txt contiene información estructurada sobre notas de diferentes personas. Un ejemplo de este fichero es el siguiente:

```
Pepe 3 5.4 7.9 6.5
Jose 5 2.4 9.7 6.4 5.8 7.8
Manuel 1 1.2
Javier 2 5.8 9.5
Antonio 3 3.0 8.0 4.0
```

En cada línea tenemos, separados por espacios, primero el nombre de la persona (máximo 50 caracteres sin espacios), después el número de notas (entero) disponible para esta persona, y finalmente cada una de las notas de esta persona (float). El máximo número de notas por persona será 10 y el número máximo de personas que podrá contener el fichero será también de 10. No será necesario realizar reserva dinámica de memoria.

Deberá escribirse un código que realice las siguientes operaciones:

(a) Definir una estructura para almacenar la información de una persona: el nombre, el número de notas, la lista de notas y la nota media. Finalmente definir un vector de estructuras para almacenar en cada posición la información de cada una de las personas de la base de datos.

(0.75 puntos)

(b) Leer el fichero y almacenar la información de cada línea en el vector estructuras.

(1.50 puntos)

(c) Calcular la nota media de cada persona y almacenarla en la estructura.

(0.50 puntos)

(d) Mostrar por pantalla el nombre y la nota media de cada persona.

(0.25 puntos)

```
Pepe: 6.6
Jose: 6.4
Manuel: 1.2
```

```
Javier: 7.7
Antonio: 5.0
```

Solución:

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

// Definición de la estructura
struct calificaciones{
      char nombre[50];
      float notas[10];
      int n_notas;
     float nota_media;
};

int main(int argc, char *argv[])
{
// Declaración de variables, vector de estructuras, fichero,
auxiliares y contadores
FILE *fichero;
struct calificaciones lista[10]; // Vector de estructuras
int total_entradas,i,j,n;
float temp;
char cadena[50];
    // Leer información del fichero y guardarlo en la estructura
    fichero = fopen("basedatos.txt", "r");
    total_entradas=0;
   while(!feof(fichero))
   {
        fscanf(fichero,"%s",lista[total_entradas].nombre);
        fscanf(fichero,"%d",&lista[total_entradas].n_notas);
        temp=0;
        for(n=0;n<lista[total_entradas].n_notas;n++)   //   Bucle
para leer las notas de cada persona
        {

fscanf(fichero,"%f",&lista[total_entradas].notas[n]);
            // Calcular nota media y guardarla en la estructura
            temp+=lista[total_entradas].notas[n];
        }
        // Calcular nota media y guardarla en la estructura

lista[total_entradas].nota_media=temp/lista[total_entradas].n_no
tas;
        // Mostrar nombre y nota media de cada entrada
        printf("%s:
%.1f\n",lista[total_entradas].nombre,lista[total_entradas].nota_
media);
    total_entradas++;
    }
   fclose(fichero);
  system("PAUSE");
}
```

4. Escribir un programa en C que lea por teclado una frase (para almacenar la cadena de caracteres utilizar un array de caracteres de tamaño 100) y un entero por teclado. A partir de esta cadena se reservará memoria de forma dinámica para un vector de enteros según el número de caracteres introducidos y se almacenará en cada posición del vector la suma del valor ASCII del carácter correspondiente más el entero introducido por teclado. A continuación se mostrará por pantalla el vector de enteros. Ejemplo de ejecución (en negrita aparecen los datos introducidos por el usuario):

```
Introduce frase: Hola Mundo
Introduce entero: 100

Vector codificado: 172 211 208 197 132 177 217 210 200 211
```

(2 puntos)

ASCII	Símbolo	ASCII	Símbolo	ASCII	Símbolo	ASCII	Símbolo	ASCII	Símbolo	ASCII	Símbolo
32	(espacio)	48	0	64	@	80	P	96	`	112	p
33	!	49	1	65	A	81	Q	97	a	113	q
34	"	50	2	66	B	82	R	98	b	114	r
35	#	51	3	67	C	83	S	99	c	115	s
36	$	52	4	68	D	84	T	100	d	116	t
37	%	53	5	69	E	85	U	101	e	117	u
38	&	54	6	70	F	86	V	102	f	118	v
39	'	55	7	71	G	87	W	103	g	119	w
40	(	56	8	72	H	88	X	104	h	120	x
41	)	57	9	73	I	89	Y	105	i	121	y
42	*	58	:	74	J	90	Z	106	j	122	z
43	+	59	;	75	K	91	[	107	k	123	{
44	,	60	<	76	L	92	\	108	l	124	|
45	-	61	=	77	M	93	]	109	m	125	}
46	.	62	>	78	N	94	^	110	n	126	~
47	/	63	?	79	O	95	_	111	o	127	□

Solución:

```
#include <stdio.h>
#include <stlib.h>

void main (void)
{
      char cadena[100];
      int *vector;
      int num,i,tam;

      printf("Introduce cadena:");
      gets(cadena);

      printf("Introduce entero:");
      scanf("%d",&num);
```

```
        // Se calcula el tamaño de la cadena
        for(i=0;cadena[i]!='\0';i++);
        tam=i;

        // Se reserve memoria para el vector
        vector=(int *)malloc(tam*sizeof(int));

        if(vector!=NULL){
              for(i=0;cadena[i]!='\0';i++){
                    vector[i]=cadena[i]+num;
              }
              printf("Vector codificado:");
              for(i=0;i<tam;i++) printf("%d ",vector[i]);
              printf("\n");
              free(vector);
        }
}
```