EXAMEN DE SISTEMAS INFORMÁTICOS INDUSTRIALES
(SOLUCIÓN) (TEORÍA)

JUNIO 2013

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

(a) Un vector en C o C++ puede definirse con diferentes tipos de datos para cada elemento. **F**
(b) En C++ se puede utilizar para mostrar el valor de una variable entera i tanto "cout<<i;" como "printf("%d",i);" **V**
(c) A un puntero se le puede asignar la dirección de memoria de una variable usando el operador de indirección (*). **F**
(d) El paso de parámetros a una función por referencia es el método recomendado para pasar parámetros en C. **F**
(e) La función scanf no puede utilizarse para leer una cadena de caracteres por teclado si ésta contiene espacios en blanco. **V**
(f) Una función global en C++ puede acceder a los datos privados de un objeto si la función se define en el mismo fichero en que se definen los métodos de la clase. **F**
(g) En una función miembro de una clase en C++ no debemos pasar explícitamente como parámetro el objeto sobre el que se aplica la función. **V**
(h) Para que un cliente se comunique con un servidor mediante sockets, el cliente no necesita conocer el puerto del servidor. **F**

(2 puntos)

2. Reservar memoria en C de forma dinámica para un array de 50 elementos, donde cada elemento es una estructura formada por: un entero (i), un flotante (f) y una cadena de 50 caracteres (cadena). *(0.5 puntos)*

Solución:

Definimos la estructura:

```
typedef struct {
        int i;
        float f;
        char cadena[50];
} estructura ;
```

Declaramos un puntero a esta estructura:

```
estructura *pe;
```

Reservamos la memoria

```
pe = (estructura *)malloc(50*sizeof(estructura));
if (pe == NULL)
{
        printf("Error al reservar memoria\n");
        exit(0);
}
```

3. Supongamos que se quiere definir una clase en C++ para representar una pila. Explicar las diferencias que existen en el destructor de la clase en los siguientes casos: *(1 punto)*

(a) Si la pila se representa con un vector dinámico.
(b) Si la pila se representa con un vector estático.

Solución:

(a) En este caso en el destructor se libera la memoria utilizando el operador ***delete*** directamente sobre el puntero que apunta al vector dinámico: delete [] puntero;

(b) En este caso el destructor no debe liberar memoria, ya que se libera automáticamente cuando finaliza la ejecución del programa.

4. Escribir una función denominada **void CalcularNotas (float alumnos[][5], int n)** con las siguientes características:

- La matriz **alumnos** que recibe como parámetro contiene **n** filas. En cada fila se almacena: número del expediente del alumno, nota 1, nota 2, nota 3. Cada dato se almacena en una columna diferente empezando desde la primera columna.
- La función debe almacenar en la última columna la nota media de cada alumno si todas las notas son mayores o iguales a 4. Si no, se almacenará un -1.
- La función debe mostrar por pantalla el número de expediente de cada alumno junto a su nota final siguiendo este formato: **Alumno: *nº exp*, Nota final: *nota final***. La nota de cada alumno se mostrará en una línea diferente.

(2 puntos)

Solución:

```
void CalcularNotas (float alumnos[][5], int n)
{
      int i;
      for(i=0;i<n;i++){
            if(alumnos[i][1]<4 || alumnos[i][2]<4 ||
               alumnos[i][3]<4){
              alumnos[i][4]=-1
            }
```

```
            else{
              alumnos[i][4]=
             (alumnos[i][1]+ alumnos[i][2]+ alumnos[i][3])/3;
            }
      }
      for(i=0;i<n;i++){
            printf("Alumno: %f, Nota final: %f\n", alumnos[i][0],
alumnos[i][4]);
      }

}
```

5. En el siguiente programa se define una clase base y una clase derivada, *C1* y *C2*. En la función *main* se declaran dos objetos *v*, *w*, de tal forma que a continuación se asigna *w* a *v*: *v = w*. Se pide:

(a) ¿Qué valores tienen *v.a*, *v.b*, *w.a, w.b* después de la asignación *v = w*? *(1 punto)*

(b) ¿Se han asignado todos los valores de los datos de w a los datos de v? En caso afirmativo, explicar brevemente la respuesta. En caso negativo, modificar el programa para que la asignación se realice correctamente, utilizando herencia. *(1.5 puntos)*

```
class C1
{
protected:
  int a;
public:
  C1(int i = 0);
  C1& operator=(C1 &aux);
};

class C2 : public C1
{
private:
  int b;
public:
  C2(int i = 0);
  C2& operator=(C2 &aux);
};

C1::C1(int i)
{
  a = i;
}
```

```
C1& C1::operator=(C1 &aux)
{
  a = aux.a;
  return (*this);
}

C2::C2(int i) : C1(i)
{
  b = i;
}

C2& C2::operator=(C2 &aux)
{
  b = aux.b;
  return (*this);
}

void main(void)
{
  C2 v, w(5);

  v = w;
}
```

Solución:

(a)

- w.a=5, w.b=5
- v.a=0, v.b=5;

(c) No. Habría que modificar el operador asignación en la clase derivada:

```
C2& C2::operator=(C2 &aux)
{
  b = aux.b;
  C1::operator=(aux);
  return (*this);
}
```

6. Escribir un programa que pida al usuario que introduzca una cadena de caracteres por teclado, almacene la cadena de caracteres en un array y muestre pon pantalla el número de caracteres que hay en mayúscula y el número de caracteres que hay en minúscula. Para almacenar la cadena de caracteres utilizar un array de caracteres de tamaño 100. Ejemplo de ejecución (en negrita aparece la cadena introducida por el usuario):

(2 puntos)

```
Introduce cadena: Esto Es Una Cadena De Prueba
Numero de caracteres en mayuscula: 6
Numero de caracteres en minuscula: 17
```

Solución:

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

void main (void)
{
      char cadena[100];
      int i, may=0,min=0;
      printf("Introduce cadena:");
      gets(cadena);
      for(i=0;cadena[i]!='\0';i++){
            if(cadena[i]>='A' && cadena[i]<='Z') may++;
            else if(cadena[i]>='a' && cadena[i]<='z') min++;
      }
      printf("Numero de caracteres en mayúscula: %d\n", may);
      printf("Numero de caracteres en minuscula: %d\n", min);
}
```