EXAMEN DE SISTEMAS INFORMÁTICOS INDUSTRIALES
(SOLUCIÓN) (TEORÍA)

SEPTIEMBRE 2016

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

(a) A un puntero se le puede asignar la dirección de memoria de una variable usando el operador indirección (*). **F**
(b) Una función en C puede llamarse a sí misma. **V**
(c) Una función global en C++ puede acceder a los datos privados de un objeto si la función se declara como amiga (friend) dentro de la clase. **V**
(d) Una clase abstracta es una clase que tiene al menos una función virtual pura. **V**
(e) En C no es posible actualizar el indicador de posición de fichero al inicio del fichero. **F**
(f) En C++, para leer desde teclado un entero y almacenarlo en la variable **dato** debería utilizarse: **cin >> "%d" >> dato; F**
(g) Dado el vector **int x[3] = {10, 11, 12};** el valor de ***x** es **10**. **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. Se desea crear en C++ dos clases:

(a) Una clase base denominada **CMultiplos2** que contendrá como variables miembros protegidas un vector de números enteros, denominado **vector2**, y un entero, denominado **tam**, que indicará el tamaño del vector. Cuando se desee crear un objeto de esta clase, se deberá especificar el valor **tam**, de forma que se reservará memoria de forma dinámica para el **vector2** y se almacenarán en este vector números consecutivos múltiplos de 2 empezando desde el valor 2. Por ejemplo, si **tam=4**, el **vector2** se rellenará con los siguientes valores: **2 4 6 8**.
(b) Una clase derivada de la clase **CMultiplos2** denominada **CMultiplos2y3** que añadirá una variable miembro privada denominada **vector3**. Cuando se desee crear un objeto de esta clase, se deberá especificar el valor **tam**, de forma que se reservará memoria de forma dinámica para el **vector2** y para el **vector3**. El **vector2** se creará tal como se ha indicado anteriormente. En el **vector3** se almacenarán números consecutivos múltiplos de 3 empezando desde el valor 3. Por ejemplo, si **tam=4**, el **vector3** se rellenará con los siguientes valores: **3 6 9 12**.

Se pide:

(a) Definir la clase **CMultiplos2** incluyendo el constructor con parámetro y el destructor. *(0.5 puntos)*
(b) Implementar el constructor de la clase **CMultiplos2** que recibe como parámetro el valor de **tam**. Se considera que el valor de **tam** siempre es mayor o igual a 1 (no es necesario realizar ninguna comprobación). *(0.75 punto)*
(c) Implementar el destructor de la clase **CMultiplos2**. *(0.25 puntos)*
(d) Definir la clase **CMultiplos2y3**, que deriva de la clase **CMultiplos2**, incluyendo el constructor con parámetro y el destructor. *(0.5 puntos)*
(e) Implementar el constructor de la clase **CMultiplos2y3** que recibe como parámetro el valor de **tam**. Se considera que el valor de **tam** siempre es mayor o igual a 1 (no es necesario realizar ninguna comprobación). Como se ha indicado anteriormente, se deberá crear tanto el **vector2** como el **vector3**. Se deberá hacer uso del mecanismo de herencia para crear el **vector2**. *(0.75 punto)*
(f) Implementar el destructor de la clase **CMultiplos2y3**. *(0.25 puntos)*

Solución:

```
class CMultiplos2{
 protected:
      int tam;
      int *vector2;
 public:
      CMultiplos2(int t);
      ~CMultiplos2(void);
};


CMultiplos2::CMultiplos2 (int t)
{
      tam=t;
      vector2=new int [tam];
      for(int i=1;i<=tam;i++){
            vector2 [i]=2*i;
      }
}

CMultiplos2::~CMultiplos2(void)
{
      delete []vector2;
}


class CMultiplos2y3: public CMultiplos2{
 private:
      int *vector3;
 public:
      CMultiplos2y3(int t);
      ~CMultiplos2y3(void);
};


CMultiplos2y3::CMultiplos2y3 (int t): CMultiplos2(t)
{
      vector3=new int [tam];
      for(int i=1;i<=tam;i++){
            vector3 [i]=3*i;
      }
}

CMultiplos2y3::~CMultiplos2y3(void)
{
      delete []vector3;
}
```

3. Crear un programa en C que solicite 10 datos. Para cada uno se comprobará si es un número, y en su caso si se trata de un número par o impar para almacenarlos en vectores independientes. No es necesario reservar la memoria dinámicamente.

(2.5 puntos)

a. Se solicitarán los 10 datos. Cada uno de los datos solicitados como cadena de caracteres se convertirá a un número entero.
b. A continuación se comprobará si la cadena se trata de un número:
   i. Si no es un número se indicará por pantalla.
   ii. Si es un número par se indicará por pantalla y se almacenará en un vector para los números pares.
   iii. Si es un número impar se indicará por pantalla y se almacenará en un vector para los números impares.
   iv. Finalmente se indicará por pantalla cuantos números pares e impares se han leído.

```
ejercicio.exe
Introduzca cadena: 3
Es un numero impar.
Introduzca cadena: 1
Es un numero impar.
Introduzca cadena: r
No es un numero.
Introduzca cadena: t
No es un numero.
Introduzca cadena: 2
Es un numero par.
Introduzca cadena: hola
No es un numero.
Introduzca cadena: 7
Es un numero impar.
Introduzca cadena: 9
Es un numero impar.
Introduzca cadena: e
No es un numero.
Cantidad de numeros pares: 1
Cantidad de numeros impares: 4
Presione una tecla para continuar . . .
```

Solución:

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

int main(int argc, char *argv[])
{
    int i;
    char caracter[1];
    int dato;
    int pares[10], impares[10];
    int n_pares=0, n_impares=0;

    for(i=0;i<9;i++)
    {   printf("Introduzca cadena: ");
        scanf("%s",caracter);
        dato=atoi(caracter);
        if(dato==0)
```

```
            {
                printf("No es un numero.\n");
                continue;
            }
        else
        {
             if(dato%2==0)
                {
                printf("Es un numero par.\n");
                pares[n_pares]=dato;
                n_pares++;
                }
             else
                {
                printf("Es un numero impar.\n");
                impares[n_impares]=dato;
                n_impares++;
                }
        }
    }
    printf("Cantidad de numeros pares: %d\n",n_pares);
    printf("Cantidad de numeros impares: %d\n",n_impares);

    system("PAUSE");
}
```

4. Crear un programa que lea una cadena de caracteres. A continuación se reservará memoria dinámicamente para un vector en función del número de palabras de la cadena. En el vector se almacenará la longitud de cada palabra. Finalmente se mostrará la media de los caracteres de las palabras de la cadena.

a. En primer lugar se solicitará al usuario que escriba una cadena de caracteres y se almacenará.
   - La cadena de caracteres estará formada por palabras (únicamente letras) que estarán separadas por un único espacio. La cadena terminará con una letra.

   (0.25 puntos)

b. A continuación se contará el número de palabras de la cadena.

   (0.35 puntos)

c. Después se reservará memoria dinámicamente para almacenar en un vector el número de caracteres de cada palabra.

   (0.40 puntos)

d. Ahora se contarán los caracteres de cada palabra almacenándolos en el vector.

   (1.00 puntos)

e. Después se calculará la media del vector, es decir, la media de caracteres de las palabras que conforman la cadena.

   (0.25 puntos)

f. Finalmente se mostrarán la cantidad de caracteres que tiene cada palabra así como la media.

   (0.25 puntos)

```
ejercicio.exe
Introduzca la cadena: Esto es un ejemplo para el examen

Cantidad de palabras: 7
Palabra(1): 4 caracteres
Palabra(2): 2 caracteres
```

```
Palabra(3): 2 caracteres
Palabra(4): 7 caracteres
Palabra(5): 4 caracteres
Palabra(6): 2 caracteres
Palabra(7): 6 caracteres
Media de caracteres por palabra: 3.86
Presione una tecla para continuar . . .
```

Solución:

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

int main(int argc, char *argv[])
{
    int i,lon,palabras=0,contador,posicion_vector, *vector;
    float media=0.0;
    char cadena[100];

    printf("Introduzca la cadena: ");
    gets(cadena);
    lon=strlen(cadena);

    // Contar palabras
    for(i=0;i<lon;i++)
    {
        if(cadena[i]==' ')
            palabras++;
    }
    palabras++;
    printf("\nCantidad de palabras: %d\n",palabras);

    // Reserva dinamica de memoria vector
    vector = (int *)malloc(palabras*sizeof(int));

    // Poner en cada posicion la longitud de cada palabra
    contador=0;
    posicion_vector=0;
    for(i=0;i<lon;i++)
    {
        if (cadena[i]==' ')
         {
            vector[posicion_vector]=contador;
            contador=0;
            posicion_vector++;
        }
        else
            contador++;
    }
    vector[posicion_vector]=contador; // Para la última palabra

    for(i=0;i<palabras;i++)
    {
        printf("Palabra(%d): %d caracteres\n",i+1,vector[i]);
        media+=vector[i];
    }
    printf("Media de caracteres por palabra: %.2f\n",media/palabras);

    system("PAUSE");
}
```