Aula		Fila		Col.	
Nombre		1er Ap.		2º Ap.	
					Firma:

NOTAS DE OBLIGATORIA LECTURA

1. Las soluciones de examen entregadas escritas a lápiz, pueden ser anuladas
2. Sé ecológico: escribe por ambas caras de cada hoja de examen
3. Para la realización de este examen es obligatoria la presentación de identificación oficial actualizada con fotografía: carnet de estudiante, dni, pasaporte o permiso de conducir formato tarjeta
4. El examen de tipo test se entrega junto con el resto del examen. Todas las preguntas de desarrollo se contestan en hoja aparte y en cualquier orden. Pueden contestarse varias preguntas en la misma hoja
5. La parte obligatoria durará 2h 30'
6. No se admite el uso de variables globales en todo el examen

Parte obligatoria. A resolver por TODOS los alumnos

TEST (1.5 puntos). Los aciertos cuentan como 0.25 puntos. 1 fallo descuenta 1/3 de acierto.

1. ¿Cúal de las siguientes opciones enumera tres dispositivos de entrada de un ordenador?
   a. Teclado, ratón y pantalla
   b. Teclado, pantalla y CPU
   c. Teclado, ratón y joystick
   d. Micrófono, pantalla y ratón

2. Si se admite que, en informática, el prefijo K (kilo) equivale a 1024, tres de las siguientes afirmaciones son correctas. Indica cuál es falsa.
   a. Un Megabyte son 1024*1024 bits
   b. Un Gigabyte son 1024 Megabytes
   c. Un Kilobyte son 1024 bytes
   d. Un Megabyte son 1024 Kilobytes

3. Indicar cuál de los siguientes canales de comunicación es el más rápido
   a. Entrada de datos por teclado
   b. Escritura por parte de la CPU en memoria
   c. Lectura de un dato desde el disco a la memoria central
   d. Tarjeta gráfica a monitor

4. Señalar aquella de las funciones que no sea propia de un Sistema Operativo
   a. Tratamiento de errores y particularidades de cada dispositivo
   b. Control de quien accede a los ficheros
   c. Gestión de memoria
   d. Almacenamiento de datos

5. Dado el siguiente fragmento de código:

```
int p=-10; while ( p <= 10 ) printf("p=%d\n", p);
```

indica cuál de las siguientes respuestas es la correcta:
   a. El bucle realiza 20 iteraciones en total
   b. Ninguna de las otras respuestas es correcta
   c. El bucle realiza un total de 21 iteraciones
   d. El bucle realiza un número infinito de iteraciones

6. De los siguientes fragmentos de un programa, hay uno cuya salida por pantalla es diferente a la de los otros 3. ¿Cuál es?

a) `int i, n=1; i=1; if ( i < n ) {printf("%d\n", i); i++;}`
b) `int i=2, n=1; while ( i <= n ) {printf("%d\n", i-1);i++;}`
c) `int i, n=1; for ( i=1 ; i < n ; i++ ) printf("%d\n", i);`
d) `int i=1, n=1; do {printf("%d\n", i); i++;} while (i<n );`

ENUNCIADOS

1. (1 punto) Se dice que un año es trisiesto si es divisible por 5 y a la vez no lo es de 500. Escribir un programa en C que solicite al usuario un año y muestre por pantalla un mensaje que indique si es o no un año trisiesto.
   El año que se solicita al usuario debe cumplir las siguientes condiciones: debe ser un año positivo y debe ser menor o igual al año actual (2006). Si el usuario introduce un año incorrecto, el programa debe mostrar un mensaje de error y volver a solicitarlo. Este proceso se repetirá hasta que el año introducido sea correcto. El programa debe utilizar una función que compruebe si el año es trisiesto. A la función se le pasa como parámetro el año correcto y retorna a la función principal verdadero si el año es trisiesto y falso si no lo es. La función principal mostrará un mensaje según el valor devuelto por la función, indicándole al usuario si el año introducido es trisiesto o no.

Solución

```
//Cabecera de programa común a ambas versiones
#include <stdio.h>
#define TRUE 1
#define FALSE 0

int trisiesto (int a)
{
```

Solución corta

```
   return a%5==0 && a%500!=0;
}

void main()
{
    int anyo;

    do
    {
       printf("Introduce el año: ");
       scanf("%d", &anyo);
       if (anyo<0 || anyo>2006)
           printf("Año incorrecto\n");
    } while (anyo<=0 || anyo>2006);

    if (trisiesto (anyo))
```

Solución elegante

```
    if (a%5==0 && a%500!=0)
       return TRUE;
    else
       return FALSE;
}

void main()
{
     int anyo, es_trisiesto;

     do
     {
        printf("Introduce el año: ");
        scanf("%d", &anyo);
        if (anyo<0 || anyo>2006)
            printf("Año incorrecto\n");
     } while (anyo<=0 || anyo>2006);

     es_trisiesto = trisiesto (anyo);

     if (es_trisiesto==TRUE)
```

```
//Final de programa común a ambas versiones
      printf("El año %d es trisiesto\n", anyo);
    else
       printf("El año %d no es trisiesto\n", anyo);
    system("PAUSE");
}
```

2. (4 puntos) **PrograMan,** el superhéroe que tiene superpoderes de programación, estaba realizando un programa para gestionar los combates de todos los superhéroes contra todos los supervillanos cuando el Doctor Virus lo paralizó. AquaMan contacta con la ETSID para que sus alumnos acaben el programa que quedó a medias y que se encuentra al final del enunciado.
   Los villanos no pueden combatir entre sí, ni los superhéroes tampoco. Cada combate sólo puede realizarse entre un único supervillano y un único superhéroe.

El programa trabaja con la matriz de combates denominada **`Clasific`** en la que se guarda el resultado de todos los combates. Se tiene una fila por cada superhéroe y una columna por cada supervillano.
En la fila **`f`** columna **`c`** de la matriz **`Clasific`** se guardará el resultado del combate entre el superhéroe número **`f`** y el supervillano número **`c`**. El resultado será **`NO_COMBATE`** si no se ha realizado, **`GANA_SUPERHEROE`** si ha ganado el superhéroe, **`GANA_SUPERVILLANO`** si ha ganado el supervillano y **`EMPATE`** si ha habido empate. El valor de estas y otras constantes, estructura de la matriz, invocación de funciones, etc. está definido en el código fuente mostrado en la siguiente página.
Cada superhéroe está identificado por un número que va de **`0`** a **`SUPERHEROES-1`**. Cada supervillano, por un número del **`0`** al **`SUPERVILLANOS-1`**.
El programa debe mostrar por pantalla el número del superhéroe ganador (el que más combates haya ganado) así como indicar si han ganado más combates los buenos, los malos o si van empatados.
Para ello se deben implementar las siguientes funciones:

1. **`Inicia`** (0.5 puntos)
   Inicia todos los elementos de la matriz al valor **`NO_COMBATE`**.
2. **`Carga`** (1 punto)
   Carga los datos del fichero "comic.txt", actualizando la matriz con los resultados de los combates. Se dispone de un fichero en el que están almacenados todos los combates que ha habido junto a su resultado. En la primera línea del fichero se tiene el número de combates que hay en él. A continuación hay una línea por cada combate, donde se indica el número de superhéroe, el número de supervillano con el que se ha enfrentado y el resultado del combate, según los valores antes mencionados.
   Ejemplo de fichero:

```
3          (Hay 3 combates en el fichero)
0 2 1      (El superhéroe 0 ha luchado con el supervillano 2 y ha ganado el supervillano)
3 1 2      (El superhéroe 3 ha luchado con el supervillano 1 y han empatado)
3 7 3      (El superhéroe 3 ha luchado con el supervillano 7 y ha ganado el superhéroe)
```

   Si un combate no aparece en el fichero, se asume que no se ha realizado.
3. **`Ganador`** (1.5 puntos)
   Devuelve a la función principal el número del superhéroe ganador (el que haya ganado más combates). La función no tiene que escribir nada por pantalla.
4. **`Ganadores`** (1 punto)
   Retorna a la función principal si ha habido más combates ganados por los supervillanos (devuelve el valor de la constante **`MALOS`**) o por los superhéroes (devuelve el valor **`BUENOS`**). Si tienen igual número de combates ganados, la función debe devolver **`EMPATE`**.

Solución

```
/* EJERCICIO 2 PARTE OBLIGATORIA */
#include <stdio.h>

/*Cuantos superhombres hay*/
#define SUPERVILLANOS       342
#define SUPERHEROES         555

/*Estado de cada batalla*/
#define NO_COMBATE                  0
#define GANA_SUPERVILLANO           1
#define EMPATE                      2
#define GANA_SUPERHEROE             3

/*Tipificación de buenos y malos*/
#define MALOS       0
#define BUENOS      1
```

```
void Inicia(int Clasific[SUPERHEROES][SUPERVILLANOS])
{
   int i,j;
   for (i=0;i<SUPERHEROES;i++) {
      for (j=0;j<SUPERVILLANOS;j++) {
         Clasific[i][j]=NO_COMBATE;
      }
   }
}

void Carga(int Clasific[SUPERHEROES][SUPERVILLANOS])
{
   int num_combates,i;
   int superheroe,supervillano,resultado;
   FILE *fp;
   fp=fopen("comic.txt","r");
   if (fp==NULL) {
      printf("El fichero comic.txt no existe.\n");
   }
   else {
      fscanf(fp,"%d",&num_combates);
      for (i=1;i<=num_combates;i++) {
         fscanf(fp,"%d",&superheroe);
         fscanf(fp,"%d",&supervillano);
         fscanf(fp,"%d",&resultado);
         Clasific[superheroe][supervillano]=resultado;
      }
      fclose(fp);
   }
}

int Ganador(int Clasific[SUPERHEROES][SUPERVILLANOS])
{
            int combates_ganados[SUPERHEROES];
            int i,j,ganador,combates_ganador;

            for (i=0;i<SUPERHEROES;i++) {
               combates_ganados[i]=0;
               for (j=0;j<SUPERVILLANOS;j++) {
                  if (Clasific[i][j]==GANA_SUPERHEROE) {
                     combates_ganados[i]++;
                  }
      }
   }
   ganador=0;
   combates_ganador=combates_ganados[0];
   for (i=1;i<SUPERHEROES;i++) {
      if (combates_ganados[i]>combates_ganador) {
         combates_ganador=combates_ganados[i];
         ganador=i;
      }
   }
   return ganador;
}
```

```
int Ganadores(int Clasific[SUPERHEROES][SUPERVILLANOS])
{
        int combates_superheroes=0,combates_villanos=0;
        int i,j;
        for (i=0;i<SUPERHEROES;i++) {
           for (j=0;j<SUPERVILLANOS;j++) {
              if (Clasific[i][j]==GANA_SUPERHEROE) {
                 combates_superheroes++;
              }
              if (Clasific[i][j]==GANA_SUPERVILLANO) {
                 combates_villanos++;
              }
     }
   }
   if (combates_superheroes>combates_villanos)
        return BUENOS;
        else if (combates_superheroes<combates_villanos)
                return MALOS;
        else
                return EMPATE;
}


/*Programa principal*/
int main()
{
   int Clasific[SUPERHEROES][SUPERVILLANOS]; /*Tabla resultados*/
   Inicia (Clasific); /*Coloca todas las posiciones a NO_COMBATE*/
   Carga (Clasific); /*Carga la clasificación desde disco*/

   printf("Bienvenido al programa de gestion de superheroes.\n");
   printf("El superheroe ganador es el numero %d\n", Ganador(Clasific));

   switch (Ganadores(Clasific)){
      case BUENOS:  printf("Los ganadores son los buenos.\n");
                        break;
      case MALOS:   printf("Los ganadores son los malos.\n");
                break;
      case EMPATE:  printf("De momento, van empatados los buenos y los malos.\n");
                 break;
      default: printf ("Error en el cómputo de batallas ganadas");
   }
   system("pause");
   return 0;
}
```

Aula		Fila		Col.	
Nombre		1er Ap.		2º Ap.	
					Firma:

NOTAS DE OBLIGATORIA LECTURA

1. Las soluciones de examen entregadas escritas a lápiz, pueden ser anuladas
2. Sé ecológico: escribe por ambas caras de cada hoja de examen
3. Para la realización de este examen es obligatoria la presentación de identificación oficial actualizada con fotografía: carnet de estudiante, dni, pasaporte o permiso de conducir formato tarjeta
4. Todas las preguntas de desarrollo se contestan en hoja aparte y en cualquier orden. Pueden contestarse varias preguntas en la misma hoja
5. La parte optativa durará 1h 30'
6. No se admite el uso de variables globales en todo el examen

Parte opcional. Sólo para los alumnos que renuncian a la nota de los exámenes parciales o no los han realizado

1. (2 puntos) Desarrollar una función en lenguaje C que reciba dos argumentos de tipo cadena. El segundo argumento contiene una frase cualquiera TODA ella en minúsculas. La primera cadena está en blanco y, tras acabar la ejecución de la función, contendrá todas las vocales que NO aparecen en la segunda cadena.
   A continuación se muestra un programa en el que se utiliza la función (de nombre **`vocales_que_no_hay`**) y un ejemplo de ejecución:

```
#include <stdio.h>
#include <string.h>
int main()
{char s[100], vocales[6];
 printf("Introduzca la frase: ");
 gets(s);
 vocales_que_no_hay(vocales,s);

 printf("En esa frase NO salen estas vocales: %s\n",vocales);
 return 0;}
```

Salida por pantalla de una posible ejecución:

```
Introduzca la frase: el perro no se ve
En esa frase NO salen estas vocales: aiu
```

Solución

```
/* EJERCICIO 1 PARTE OPCIONAL */
#include <stdio.h>
#include <string.h>

void vocales_que_no_hay(char vocales[],char s[])
{
   int i,k,lg;
   int hay_a=0,hay_e=0,hay_i=0,hay_o=0,hay_u=0;
   lg=strlen(s);
   for (i=0;i<lg;i++) {
      switch (s[i]) {
         case 'a': hay_a=1;
                   break;
         case 'e': hay_e=1;
                   break;
         case 'i': hay_i=1;
                   break;
         case 'o': hay_o=1;
                   break;
         case 'u': hay_u=1;
                   break;
      }
   }
   k=0;
   if (hay_a==0) {
            vocales[k]='a';
            k++;
            }
   if (hay_e==0) {
            vocales[k]='e';
            k++;
            }
   if (hay_i==0) {
            vocales[k]='i';
            k++;
            }
   if (hay_o==0) {
            vocales[k]='o';
            k++;
            }
   if (hay_u==0) {
            vocales[k]='u';
            k++;
            }
   vocales[k]='\0';
}


int main()
{
   char s[100],vocales[6];
   printf("Introduzca la frase: ");
   gets(s);
   vocales_que_no_hay(vocales,s);
   printf("En esa frase NO salen estas vocales: %s\n",vocales);
   system("pause");
   return 0;
}
```

2. (1.5 puntos) Realizar un programa que pida al usuario un número entero positivo. Si el número introducido es múltiplo de 2, o de 3, o de 5, el programa continuará pidiendo un nuevo número. En cambio, si el número introducido no es múltiplo ni de 2, ni de 3, ni de 5, el programa acabará.
   Ejemplo de ejecución.

   Introduzca por favor un número:
   2
   Introduzca por favor un número:
   25
   Introduzca por favor un número:
   30
   Introduzca por favor un número:
   45
   Introduzca por favor un número:
   49
   No es multiplo ni de 2, ni de 3, ni de 5.

   Fin de programa

Solución

```
int main()
{
   int numero;
   do {
      do {
         printf("Introduzca por favor un numero: ");
         scanf("%d",&numero);
         if (numero<=0) {
            printf("El numero debe ser positivo.\n");
         }
      } while (numero<=0);
   } while (((numero%2)==0) || ((numero%3)==0) || ((numero%5)==0));
   printf("No es multiplo ni de 2, ni de 3, ni de 5.\n");
   system("pause");
   return 0;
}
```