Apellidos: **Nombre:** **Matrícula:**

Examen Teórico

Programación para Sistemas

Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Curso 2021/2022 - Enero 2022

Normas

- Se deberá rellenar **apellidos, nombre y número de matrícula** en cada hoja.
- Se deberá tener el **DNI** o el carnet de la UPM en **lugar visible**.
- El examen consta de **12 preguntas** que suman un total de **15 puntos**.
- La duración total del mismo es de **30 minutos**.
- Las preguntas de **tipo test** sólo tiene **una opción válida**. Si se marca más de una opción considerará una respuesta incorrecta. Toda **respuesta incorrecta restará** la puntuación de la pregunta dividida por el número de opciones. Toda pregunta no contestada no restará. Las preguntas de respuesta libre no restarán.
- Las calificaciones se darán a conocer a través del Moodle de la asignatura y la solución al examen se proporcionará antes de la revisión.

Cuestionario

(1 punto) 1. A continuación se muestran dos códigos C que intentan abrir un fichero `f.txt` y comprobar si éste ha sido abierto exitosamente (según el manual, "la función `feof()` comprueba el indicador de fin-de-fichero del fichero apuntado por el argumento devolviendo un valo no cero si el indicador está activado.")

Código 1

```
FILE *f = fopen("f.txt", "r");
if (feof(f))
  fprintf(stderr,
          "No se pudo abrir\n");
```

Código 2

```
FILE *f = fopen("f.txt", "r");
if (f == NULL)
  fprintf(stderr,
          "No se pudo abrir\n");
```

Se pide señalar cuál de los dos códigos realiza la comprobación correctamente.

A. Código 1.

B. Código 2.

(2 puntos) 2. Sea `a` un vector de enteros declarado en C de esta forma

```
int a[10];
```

Se pide escribir la expresión en lenguaje C que devuelve la dirección de memoria del quinto elemento (índice 4) del vector.

> **Solución:** `&a[4]` o `a+4` (el enunciado pide una *expresión*, no una sentencia, y mucho menos un `print`, pero si se usa un `print` el formato deberá ser `"%p"`)

(1½ puntos) 3. Dado el siguiente programa C:

```c
#include <stdio.h>

void fun(int x)
{
  x = x+5;
}

int main(void)
{
  int x=5;
  fun(x);
  printf("%d\n",x);
  return 0;
}
```

Se pide señalar la respuesta correcta:

A. La salida del programa es `10`.

B. La salida del programa es `5`.

C. No hay salida porque el programa no compila.

(2 puntos) 4. A continuación se presenta una porción de código C que define el tipo `list_t` para implementar listas como cadenas enlazadas, y una función `delete` para borrar el primer elemento de una lista si no está vacía:

```c
typedef struct node_s {
  int data;
  struct node_s *next;
} *list_t;

void delete(list_t *pl) {
  if (*pl != NULL) *pl = (*pl)->next;
}
```

Se pide señalar la respuesta correcta:

A. La función `delete` provoca una fuga de memoria (*memory leak*)[1].

B. La función `delete` **no** provoca una fuga de memoria (*memory leak*).

(1 punto) 5. El manual de la función `strdup` de la biblioteca `<string.h>` dice:

```
SINOPSIS
       #include <string.h>

       char *strdup(const char *s);
[...]
       En caso de éxito, la función strdup() devuelve un puntero
       a un nuevo string duplicado de s.  Devuelve NULL si no hay
       memoria disponible suficiente. [...]
```

¿Es necesario liberar la memoria de un puntero devuelto por la función `strdup()`?

Se pide señalar la respuesta correcta:

A. Sí.

B. No.

[1] Una fuga de memoria es un error de software que ocurre cuando no se libera memoria reservada.

(1 punto) 6. Dado el siguiente programa en C:

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

int main(void)
{
  char *s = (char*)calloc(10, sizeof(char));
  printf("%lu\n",sizeof(s)/sizeof(s[0]));
  free(s);
  return 0;
}
```

y asumiendo que `sizeof(char*)` y `sizeof(char)` devuelven `4` y `1` respectivamente[2].

Se pide señalar la respuesta correcta:

A. La salida del programa es `10`.

B. La salida del programa es `4`.

(1½ puntos) 7. El manual de la función `calloc` en la biblioteca `<stdlib.h>` dice:

```
SINOPSIS
       #include <stdlib.h>

       void *calloc(size_t nmemb, size_t size);
[...]
       The  calloc()  function  allocates memory for an array of
       nmemb elements of size bytes each and returns a pointer to
       the  allocated  memory.  The memory is set to zero.  [...]
```

Dado el siguiente programa:

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

int main(void)
{
  char *s = (char*)calloc(10, sizeof(char));
  printf("%lu\n", strlen(s));
  free(s);
  return 0;
}
```

Se pide señalar la respuesta correcta:

A. La salida del programa es `0`.

B. La salida del programa es `10`.

C. No es posible conocer la salida al no haber inicializado `s`.

[2]El resultado de **`sizeof`** es de tipo `size_t`, es decir *long unsigned*

(1 punto) 8. Dado el siguiente código C para la función `g`:

```
void g(int *x)
{
  *x = *x + 1;
}
```

Se pide escribir dos líneas de código C para declarar una varible de tipo **`int`**, asignarle `42` y llamar a la función `g` correctamente (con la intención de incrementar dicha variable en `1`).

Solución:

```
int x = 42;
g(&x);
```

(1 punto) 9. ¿Cuál es la opción que denota en Bash el valor del estado de terminación del último mandato ejecutado?

Se pide señalar la respuesta correcta:

A. `$*`

B. `$@`

C. `$?`

D. `$!`

(1 punto) 10. Suponiendo que las variables Bash `A` y `B` contienen números enteros válidos. ¿Cuál de los siguientes mandatos comprueba si dichos enteros son iguales?

A. `[ $A == $B ]`

B. `[ $A -eq $B ]`

(1 punto) 11. El programa **`cat`** *concatena ficheros y los imprime por la salida estándar.* El programa `wc` *cuenta el número de líneas de la entrada estándar.*

Se pide escribir en una sóla línea un mandato Bash que saque por la salida estándar el número de líneas de un fichero `mifichero.txt`

Solución: **`cat`** `mifichero.txt | wc` (por supuesto se ha considerado correcto "**`cat`** `mifichero.txt | wc -l`", y también "`wc -l mifichero.txt`", pero no "`wc -l mifichero.txt |` **`cat`**" dado que es un tanto absurdo y que el texto del enunciado no menciona nada sobre los parámetros de `wc`)

(1 punto) 12. Dado el siguiente script de Bash `parametros.sh`:

```
#!/bin/bash
echo $#
```

¿Cuál es el resultado de la siguiente invocación del mismo desde la línea de mandatos?

```
$ ./parametros.sh esto es una prueba
```

Solución:

```
4
```