Apellidos: **Nombre:** **Matrícula:**

Examen Teórico

Programación para Sistemas

Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Curso 2021/2022 - Julio 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 **21 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

(1½ puntos) 1. De acuerdo con el manual de `fgets`:

```
SYNOPSIS
    #include <stdio.h>

    char *fgets(char *s, int size, FILE *stream);
[...]
    fgets()  reads  in  at  most one less than size characters from
    stream and stores them into the buffer pointed to by s.  Read-
    ing  stops after an EOF or a newline.  If a newline is read, it
    is stored into the buffer.  A terminating null byte  ('\0')  is
    stored after the last character in the buffer.
```

Supóngase que en la entrada estándar (`stdin`) hay al menos dos líneas. Supóngase que se quiere leer la primera línea de `stdin` con el siguiente código:

```
char line[1001];
fgets(line, 1000, stdin);
```

¿Qué expresión de C me dice el número de caracteres leídos por `fgets`?

Se pide señalar la respuesta correcta.

A. `sizeof(line)/sizeof(line[0])`

B. `strlen(line)`

C. `1000`

(1 punto) 2. Supongamos que ya se ha calculado el número de caracteres leídos por `fgets` en el ejercicio 1. Supongamos que una variable `N` almacena ese cálculo.

Se pide escribir una única llamada a `fgets` para leer la segunda línea de `stdin` y concatenarla en `line` immediatamente después del caracter `'\n'` de la primera línea ya almacenada. (**Nota:** téngase en cuenta que no debe superarse el límite de tamaño del vector `line`).

Solución:

```
fgets(line + N, 1000 - N, stdin);
```

(1½ puntos) 3. El siguiente código C pretende declarar un vector `a` de 10 elementos e inicializar todos sus elementos a `1`:

```
int j, a[10];
for (j = 0; j < 10; j++) *(a+j) = 1;
```

Se pide señalar la respuesta correcta:

A. El código es correcto.

B. El código compila pero es incorrecto.

C. El código no compila.

(3 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:

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

```
void delete(list_t l) {
  if (l != NULL) {
    list_t d = l;
    l = l->next;
    free(d);
  }
}
```

Supóngase una variable `c` de tipo `lista_t`. Supóngase que dicha variable es el inicio de una cadena enlazada correctamente construida (por lo tanto es distinto de `NULL`). Supóngase que se ejecuta la sentencia `delete(c)`.

Se pide señalar la afirmación correcta:

A. La función `delete` tiene un error de compilación.

B. La variable `c` queda apuntando a un nodo liberado (es un *dangling pointer*)

C. La variable `c` queda apuntado al resto de la cadena inicial menos el primer elemento.

(3 puntos) 5. Dado el siguiente programa C:

```
#include <stdio.h>
void swap(int *x, int *y) {
  int cx = *x;
  *x = *y;
  *y = cx;
}
```

```
int main( void ) {
  int x = 1, y = 2;
  swap(&x, &y);
  printf("%d-%d\n", x, y);
  return 0;
}
```

Se pide señalar la respuesta correcta:

A. La salida del programa es `1-2`.

B. La salida del programa es `2-1`.

C. No hay salida porque el programa no compila.

(1½ puntos) 6. El manual de la función `strcpy` de la biblioteca `<string.h>` dice:

```
SYNOPSIS
    #include <string.h>

    char *strcpy(char *dest, const char *src);
[...]
    The  strcpy() function copies the (null-terminated) string
    pointed to by src, including the terminating null byte ('\0'),
    to the buffer pointed to by dest. The strings may not overlap,
    and the destination string dest must be large enough to
    receive the copy. The strcpy() function returns a pointer to
    the destination string dest.
```

Supóngase que la función se ha programado de esta forma:

```
char *strcpy(char *dest, const char *src) {
  size_t i;
  for (i = 0; src[i] != '\0'; i++)
    dest[i] = src[i];
  return dest;
}
```

Se pide señalar la afirmación correcta:

A. Es una implementación correcta de la función.

B. Es una implementación incorrecta de la función.

C. El código no compila porque no se puede usar un puntero como si fuera un vector.

(1½ puntos) 7. El manual de la función `malloc` dice:

```
SYNOPSIS
    #include <stdlib.h>

    void *malloc(size_t size);
[...]
    The  malloc()  function  allocates  size  bytes  and  returns a
    pointer to the allocated memory.  The memory  is  not  initial-
    ized.   If  size  is 0, then malloc() returns either NULL, or a
    unique pointer value that can later be successfully  passed  to
    free().
```

Dado el siguiente programa:

```
int main( void ) {
  char *s = (char*)malloc(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`.

(1 punto) 8. El siguiente código C para la función `inv` invierte un número entero dado:

```
void inv(int *x) {
  *x = -(*x);
}
```

Se pide, suponiendo la existencia de una variable declarada como **`int n;`** y apropiadamente inicializada, escribir una línea de código con la llamada a `inv` para invertir el valor de la variable `n`.

Solución:

```
inv(&n);
```

(1 punto) 9. Suponiendo que las variables de Bash `A` y `B` contienen números enteros válidos. ¿Cuál de los siguientes mandatos comprueba si `A` es menor que `B`?

A. `[ $A < $B ]`

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

(1 punto) 10. El programa **`cat`** *concatena ficheros* que recibe como argumentos y los imprime por la salida estándar. El programa `grep` *filtra* las líneas de la entrada estándar que contienen un determinado texto que recibe como argumento. 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 en las que aparece la palabra `Sancho` en los ficheros `quijote1.txt` y `quijote2.txt`.

Solución: **`cat`** `quijote1.txt quijote2.txt | grep Sancho | wc`

(3 puntos) 11. Dado el siguiente script de Bash `parametros.sh`:

```
#!/bin/bash
echo $0-$1
```

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

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

Se pide señalar la respuesta correcta:

A. `esto-es`

B. `./parametros.sh-esto es una prueba`

C. `./parametros.sh-esto`

(2 puntos) 12. ¿Cuál es la opción que denota en Bash el número de parámetros en un *script*?

Se pide señalar la respuesta correcta:

A. `$*`

B. `$#`

C. `$@`

D. `$?`