EJERCICIOS TIPO TEST DE "PROGRAMACIÓN" ORDENADOS POR TEMAS

Escuela Politécnica Superior
Universidad de Alcalá

GRADO EN INGENIERIA EN TECNOLOGIAS DE LA TELECOMUNICACION (GITT)

GRADO EN INGENIERIA EN SISTEMAS DE TELECOMUNICACION (GIST)

GRADO EN INGENIERIA ELECTRONICA DE COMUNICACIONES (GIEC)

GRADO EN INGENIERIA TELEMATICA (GIT)

Fecha Actualización: 30/05/2019

INFORMACION SOBRE LA FORMA DE RESPONDER LOS TESTS

Tras el enunciado de cada pregunta, existen cuatro respuestas numeradas: 01) 02) 04) 08).

La respuesta a cada pregunta se aporta mediante un número que corresponde con la SUMA de las respuestas correctas.

Ejemplos:

-- Si son correctas las opciones 01) y 04), la respuesta será: 05
-- Si son correctas las opciones 01), 02) y 08), la respuesta será: 11
-- Si sólo es correcta la opción 08), la respuesta será: 08
-- Si ninguna opción es correcta, la respuesta será: 00

Las respuestas correctas se encuentran anotadas en la última página de este documento.

PUNTEROS A ARRAYS Y A CADENAS DE CARACTERES

1. La función:

```
void fun(unsigned char *cad1, unsigned char *cad2)
{
   while (*cad2++=*cad1++);
}
```

01) Copia la cadena cad2 en la cadena cad1.
02) Copia la cadena cad1 en la cadena cad2.
04) Copia la cadena cad2 en la cadena cad1 excepto el caracter '\0'.
08) Copia la cadena cad1 en la cadena cad2 excepto el caracter '\0'.

---

2. ¿Qué se imprime al ejecutar el siguiente programa?

```
main()
{   int x[ ]={3,88,6,67,99};
   while (*x<=90)
      printf("%d  ",*x++);
}
```

01) 3 88 6 67 99
02) 99 67 6 88 3
04) 3 88 6 67
08) 99

---

3. Suponiendo que se ha declarado: **int x[10], *p=x;**
indique qué instrucciones pueden ser correctas

01) **scanf(“%d”,&x[1]);**
02) **scanf(“%d”,x+1);**
04) **printf(“%d”,*x++);**
08) **scanf(“%d”,p++);**

---

4. Dadas las siguientes sentencias:

```
char a[ ]="Examen de programacion";
char *p= "Examen de mayo";
```

Indicar qué sentencias son correctas.

01) *(a+1)=p[3];
02) *(a+2)='A';
04) a=p++;
08) p++=a;

---

5. ¿Qué se imprime al ejecutar la instrucción **fun(“EUP”);** ?

```
void fun(char *cad)
{ int n; char *p=cad;
  while (*cad++);
  n=cad-p;
  printf(“%d%s”,n,p);
}
```

01) 1E2U3P
02) EUP3
04) 3EUP
08) 4EUP

---

6. Dadas estas declaraciones: **int a[5]={10,20,30,40,50}, *p=a, i=2;**

01) **printf("%d",*p[i]); imprime 30**
02) **printf("%d",*(p+i)); imprime 30**
04) **printf("%d",*++p+i); imprime 22**
08) **printf("%d",*a++); imprime 10**

---

7. Con la siguiente declaración: **int x[ ]={3,2,8,7,5}, *p=x;**

01) La instrucción **printf("%d", *x+3);** imprime 6
02) La instrucción **printf("%d", (*x)++);** imprime 3
04) La instrucción **printf("%d", *(x+3));** imprime 7
08) La instrucción **printf("%d", *p++);** imprime 3

---

8. Con las siguientes declaraciones, indicar qué llamadas a la función **strcpy()** son correctas.
char cad[40]="escuela", otro[40], *p=otro, *q=cad;

01) strcpy(otro, cad);
02) strcpy(p, q);
04) strcpy(otro, "politecnica");
08) strcpy(p, &cad[0]) ;

---

9. Con la siguiente declaración: **int x[ ]={3,2,8,7,5};**

01) La instrucción **printf("%d", *x++);** imprime 3
02) La instrucción **printf("%d", (*x)++);** imprime 3
04) La instrucción **printf("%d", *(x+3));** imprime 7
08) La instrucción **printf("%d", *(x+3)=-1);** imprime -1

---

10. Dadas las siguientes sentencias:

char a[ ]="Hoy es lunes";
char *p= "Dia de examen";

Indicar qué sentencia es **INCORRECTA**

01) *(a+1)=p[3];
02) *(a+2)='A';
04) a[0]=p[0];
08) p++=a++;

---

11. Con la siguiente declaración, ¿cómo podemos saber el número de elementos del array?

int array[15]={0};

01) sizeof(array)/sizeof(int);
02) sizeof(array)*sizeof(int);
04) sizeof(array);
08) strlen(array);

---

12. De las siguientes funciones que intercambian dos valores del array **a**, indicar cual es **INCORRECTA**

```
void main()
{
      int array[2]={10,20};
      cambiar(array);
}
```

01)
```
void cambiar(int a[ ])
    {
        int aux = a[0]; a[0]=a[1]; a[1]=aux;
    }
```

02)
```
void cambiar(int *a)
    {
        int aux = *a; *a = *(a+1); *(a+1)=aux;
    }
```

04)
```
void cambiar(int *a)
    {
        int aux = a; a = a[1]; a[1]=aux;
    }
```

08)
```
void cambiar(int *a)
    {
        int aux = a[0]; a[0] = *(a+1); *(a+1)=aux;
    }
```

---

13. ¿Qué imprime el siguiente programa, si por teclado se introduce el texto **Salamanca** y Enter?

```
main()
{
  char cad[40], *p=cad;
  gets(cad);
  p[0]++;
  *(p+1) = 'e';
  printf("%s", p);
}
```

01) elamanca
02) Talamanca
04) Selamanca
08) Telamanca

---

14. Dado el siguiente código, indicar por qué hay que susitituir los comentarios para que el programa funcione adecuadamente:

```
char cadena1[ ]={"Jane Doe"};
char cadena2[ ]={"John Doe"};

void main()
{
```

```
    if ( /* Comentario 1 */ )
       printf("La cadena 1 es más larga\n");
    else
       printf("La cadena 1 no es más larga\n");

    if ( /* Comentario 2 */ )
       printf("La cadena 1 está alfabéticamente antes\n");
    else
       printf("La cadena 2 está alfabéticamente antes o son iguales\n");
}
```

01) Comentario 1 → strlen(cadena1) > strlen(cadena2)
Comentario 2 → strcmp(cadena1,cadena2) < 0

02) Comentario 1 → cadena1 > cadena2
Comentario 2 → cadena1 > cadena2

04) Comentario 1 → *cadena1 > *cadena2
Comentario 2 → strcmp(cadena1,cadena2) < 0

08) Comentario 1 → *cadena1 > *cadena2
Comentario 2 → cadena1[0] > cadena2[0]

PUNTEROS A ESTRUCTURAS

1. Con las siguientes declaraciones:

```
typedef struct
{  float real;
   float imag;
} complejo;
complejo x[4],*p=x;
```

indique qué instrucciones pueden ser correctas.

01) **scanf("%f%f",&x[0].real,&x[0].imag);**
02) **scanf("%f%f",p->real,p->imag);**
04) **scanf("%f%f",&p->real,&p->imag);**
08) **printf("%f%f",*p->real,*p->imag);**

---

2. Indicar qué resultado se produce al ejecutar el siguiente programa:

```
struct datos
{   int i1, i2;
    double d;
    char c;
};
main( )
{  struct datos dato1={2,10,5.55,'z'};
   struct datos *p=&dato1;
   p->c='a';
   printf("%d,%c,%c",p->i1,p->c,dato1.c);
}
```

01) 2,a,z
02) valor basura,a,valor basura
04) 2,a,a
08) valor basura,valor basura,z

---

3. Con las siguientes instrucciones, indique qué afirmaciones son correctas:

```
struct
{ int a;
  char *nom;
} e1,*p=&e1;
p->a=50; p->nom="septiembre";
```

01) printf("%d",*p->a); imprime 50
02) printf("%d",(++p)->a); imprime 51
04) printf("%s",++p->nom); imprime eptiembre
08) printf("%c",*++p->nom); imprime e

---

4. Con las siguientes declaraciones, y suponiendo que los datos **int** ocupan 4 bytes:

```
typedef struct { int d,m,a; } tfecha;
typedef struct
{ char nom[40];
  tfecha fecha;
  int edad;
} tipo;
tipo var, x[10], *p=x;
```

01) La instrucción **printf("%d %d\n", sizeof(tipo), sizeof(tfecha));** imprime **56 12**
02) Las siguientes instrucciones pueden ser correctas:
    gets(p->nom);
    scanf("%d %d %d %d", p->fecha.d, p->fecha.m, p->fecha.a, p->edad);
04) Las siguientes instrucciones pueden ser correctas:
    gets(x[0].nom);
    scanf("%d %d %d %d", &p->fecha.d, &p->fecha.m, &p->fecha.a, &p->edad);
08) Las siguientes instrucciones pueden ser correctas:
    p++; p[0].edad++;

---

5. En el siguiente programa, indicar qué sentencias hay que sustituir por el COMENTARIO en la funcion **fun** para que imprima todos los nombres y notas leidos y despues incremente en uno la nota de todos los alumnos.

```
typedef struct
{  char nombre[21];
   float nota;
} alu;
void fun(alu *x, int n);

main()
{  int i=0, n;
   alu a[10];
   while( i<10)
   {   printf("Dame nombre: ");  if (gets(a[i].nombre)==NULL)  break;
       printf("Dame nota: ");  scanf("%f", &a[i].nota);  fflush(stdin);  i++;
   }
   n=i;
   fun(a , n);
}

void fun(alu *x, int n)
```

```
{ int i;
  /*  COMENTARIO A SUSTITUIR  */
}
```

```
01) for (i=0; i<n; i++)   printf("\n%s  %f", x[i].nombre, x[i].nota++);
02) for (i=0; i<n; i++,x++)   printf("\n%s  %f", x->nombre, x->nota++);
04) for (i=0; i<n; i++,x++)   printf("\n%s  %f", (*x).nombre, (*x).nota++);
08) for (i=0; i<n; i++)   printf("\n%s  %f", (x+i).nombre, (x+i).nota++);
```

---

6. Con las siguientes declaraciones:

```
struct complejo
{
      float real;
      float imag;
};
struct complejo  x[10];
struct complejo *p=x
```

Indicar qué sentencia es **INCORRECTA**

```
01) scanf("%f %f", &x[0].real, &x[0].imag);
02) scanf("%f %f", p->real, p->imag);
04) scanf("%f %f", &p->real, &p->imag);
08) printf("%f %f", (*p).real, (*p).imag);
```

PUNTEROS GENÉRICOS

1. Indicar qué sentencias hay que sustituir por el comentario para que, al ejecutar el siguiente programa, se imprima el valor de **x**.

```
main( )
{  int x=3;
   void *p=&x;   //← Puntero genérico
   //  COMENTARIO A SUSTITUIR
}
```

```
01) printf ("%d",*p);
02) printf ("%d",*((int *)p));
04) printf ("%d",void *p);
08) printf ("%d",(int *)p);
```

---

2. Indicar qué sentencias hay que sustituir por el comentario para que, al ejecutar el siguiente programa, se tome por teclado el valor de **x**.

```
main( )
{  int x=3;
   void *p=&x;   //← Puntero genérico
   printf("Dame valor de x: ");
   //  COMENTARIO A SUSTITUIR
}
```

```
01) scanf ("%d",p);
02) scanf ("%d",(void *)p);
```

04) scanf ("%d",&p);
08) scanf ("%d",(int *)p);

---

3. Indicar qué sentencias hay que sustituir por el comentario para que, al ejecutar el siguiente programa, se imprima correctamente el valor de **x**.

```
main( )
{  int x=3;
   void *p=&x;
   //  COMENTARIO A SUSTITUIR
}
```

01) printf ("%d", *p);
02) printf ("%d", (int *)p);
04) printf ("%d", void *p);
08) printf ("%d", *(int *)p);

ARRAYS DE PUNTEROS

1. Indicar qué sentencias hay que sustituir por el comentario para que el siguiente programa imprima por pantalla todos los datos del array **a.**

```
struct alu
{ char nombre[30];
  float nota;
};
main()
{  struct alu a[3], *p[3], **pp=p;   int i ;
   for (i=0; i<3; i++)  p[i]=&a[i];
   printf("Introduce nombres y notas:\n") ;
   for (i=0; i<3; i++)
   {   gets(a[i].nombre); scanf("%f", &a[i].nota);
       getchar();   // para limpiar el buffer del teclado
   }
   printf("\nLos datos introducidos son:\n") ;
   for (i=0; i<3; i++)
   {  /* COMENTARIO A SUSTITUIR */  }
}
```

01) printf("Nombre: %s\tNota: %g\n", p[i]->nombre, p[i]->nota);
02) printf("Nombre: %s\tNota: %g\n", pp[i]->nombre, pp[i]->nota);
04) printf("Nombre: %s\tNota: %g\n", a[i].nombre, a[i].nota);
08) printf("Nombre: %s\tNota: %g\n", a[i]->nombre, a[i]->nota);

---

2. El prototipo de una función es: **char *fun(char **,char *,char);**
Suponiendo que además se ha declarado:

```
char *p[]={"uno","dos","tres","cuatro"};
char **pp=p;
char car='r';
```

Indique qué llamadas a **fun()** pueden ser correctas:

01) p[0]=fun(p,p[0],p[0][0]);
02) fun(p,p[0],car);
04) p[1]=fun(&p[0],p[1],'w');
08) fun(pp,"nombre",'w');

---

3. ¿Qué imprime este programa?

```
main()
{
  char *p[ ]={"lunes","martes","miercoles","jueves","viernes", "sabado","domingo",NULL};
  char **pp=p;
  p[1]++;
  while(*pp!=NULL )  printf("%s",*pp++);
}
```

01) martesmiercolesjuevesviernessabadodomingo
02) lunesmartesmiercolesjuevesviernessabadodomingo
04) lunesartesmiercolesjuevesviernessabadodomingo
08) munesnartesoiercoleskueveswiernestabadoeomingo

---

4. Indicar qué sentencias hay que sustituir por el comentario para que el siguiente programa lea desde teclado todos los datos del array **a.**

```
struct alu
{ char nombre[30];
  float nota;
};
main()
{  struct alu a[3], *p[3], **pp=p;
  int i ;
  for (i=0; i<3; i++)  p[i]=&a[i];
  printf("Introduce nombres y notas:\n") ;
  for (i=0; i<3; i++)
  {  /*  COMENTARIO A SUSTITUIR   */
     getchar();  // para limpiar el buffer del teclado
  }
}
```

01) **scanf("%s %f", p[i]->nombre, p[i]->nota)**;
02) **{ gets(a[i].nombre); scanf("%f", &a[i].nota); }**
04) **{ gets(pp[i]->nombre); scanf("%f", &pp[i]->nota); }**
08) **scanf("%s %f", &p[i]->nombre, &p[i]->nota)**;

---

5. Indique que instrucciones son las que hay que incluir en COMENTARIO para imprimir todas las cadenas de caracteres.

```
main()
{
 char *p[ ]={"lunes", "martes", "miercoles", "jueves", "viernes", "sabado", "domingo", NULL};
 char **pp=p;
 // COMENTARIO
}
```

01) while (*pp!=NULL)
    { printf("%s ", *pp); pp++; }
02) while (*pp==NULL)
    { printf("%s ", *pp); pp++; }
04) while (pp!=NULL)
    { printf("%s ", pp); *pp++; }
08) while (*pp!=NULL)
    { printf("%s ", pp); *pp++; }

---

6. ¿Qué imprime el siguiente programa?

```
void main()
{
  char *dias[ ] = {"Lunes", "Martes", "Miercoles", "Jueves", "Viernes", "Sabado", "Domingo"};
  char *p = dias[0];
  int i;
  for (i=0; i<6; i++)
     dias[i] = dias[i+1];
  dias[i] = p;
  for (i=0; i<7; i++)
     printf("%s  ", dias[i]);
}
```

01) Lunes Martes Miercoles Jueves Viernes Sabado Domingo
02) Martes Miercoles Jueves Viernes Sabado Domingo Lunes
04) Munes Nartes Niercoles Kueves Wiernes Tabado Eomingo
08) uunes aartes iiercoles uueves iiernes aabado Lomingo

PUNTEROS A PUNTEROS

1. ¿Qué se imprime al ejecutar este programa?

```
void fun(int **);
main()
{ int a[2][2]={10,20,30,40}, *b[2]={a[0],&a[1][0]}, **p=b;
  printf("%d, %d, %d, ",**p, b[1][1], **(p+1));
  fun(b);
}
void fun(int **q)
{
 printf("%d",**(q+1));
}
```

01) 10, 20, 30, 30
02) 10, 30, 40, 30
04) 10, 20, 30, 40
08) 10, 40, 30, 30

---

2. Una función tiene el siguiente prototipo: **char *fun(char **, char *, char);**
Suponiendo además las siguientes declaraciones:

char *p=”examen”; char cad[ ]=”ordinario”; char **pp=&p;

Indique qué llamadas a la función **fun()** pueden ser correctas:

01) cad=fun(pp,&p[0],’A’);
02) p=fun(pp,p,”A”);
04) p=fun(p,cad+1,’B’);
08) p=fun(&p,cad,’B’);

---

3. Con las siguientes instrucciones:

float x[ ]={4,-8,9,56,-13.6,32.1};

```
float *p=x;
float **pp=&p;
```

01) La instrucción printf("%g ",**pp); imprime 4
02) La instrucción printf("%g ",*++p); imprime -8
04) La instrucción printf("%g ",*(p+3)); imprime 56
08) La instrucción printf("%g ",*(*pp+3)); imprime 56

---

4. Dadas estas declaraciones: **int a[2][3]={10,20,30,40,50,60};**
int *p[2]={a[0],a[1]};
int **pp=p;

01) **printf("%d",*p[1]); imprime 40**
02) **printf("%d",p[0][5]); imprime 60**
04) **printf("%d",*(*(pp+1))); imprime 40**
08) **printf("%d",p[1][1]); imprime 50**

---

5. Con las siguientes declaraciones, indique qué instrucciones pueden ser correctas:

int x[10], *p=x, **pp;

01) pp=&p;
02) pp=p;
04) pp=*p;
08) p=x+2;

---

6. Con las siguientes declaraciones, indique qué instrucciones pueden ser correctas:

int x[10], *p=x, **pp;

01) p++;
02) x++;
04) pp=p++;
08) pp=&p;

---

7. Dadas estas declaraciones: **int a[2][3]={10,20,30,40,50,60};**
int *p[2]={a[0],a[1]};
int **pp=p;
Indicar que afirmación es INCORRECTA.

01) printf("%d", *p[1]+1); imprime 41
02) printf("%d", pp[0][2]); imprime 30
04) printf("%d", **(pp+1)); imprime 40
08) printf("%d", p[1][1]); imprime 50

PASO DE ARGUMENTOS POR REFERENCIA

1. El siguiente programa, ¿qué imprime en pantalla?

```
fun1(int *v1, int v2);
main( )
{
   int v1=10,v2=20;
   fun1(&v1,v2);
```

```
    printf("v1=%d,v2=%d",v1,v2);
}
fun1(int *v1, int v2)
{  *v1=30;   v2=50;  }
```

01) v1=10,v2=20
02) v1=30,v2=50
04) v1=30,v2=20
08) v1=10,v2=50

---

2. ¿Qué imprime este programa, si al ejecutarlo se introducen los valores **3** y **4** para las variables **a** y **b**?

```
void fun( double s1, double *s2);
main()
{  double a,b;
   scanf("%lf %lf", &a,&b);
   fun(a,&b);
   printf("%.1lf,%.1lf\n",a,b);
}
void fun( double s1, double *s2)
{  double x=s1-*s2;
   s1+=*s2;  *s2=x;
}
```

01) 3.0,4.0
02) 3.0,-1.0
04) 7.0,-1.0
08) 7.0,4.0

---

3. ¿Qué imprimirá el siguiente programa?

```
int f1(int *a)
{ *a = *a * *a;  return *a;  }
void f2(int *a, int b)
{ b = f1(a) * f1(&b); }
main()
{  int a=2,b=2;
   f2(&a,b);
   printf("%d, %d",a,b);
}
```

01) 4, 16
02) 2, 16
04) 4, 2
08) 2, 2

---

4. Dado el siguiente programa, ¿qué valor toman las variables si tecleamos los valores **7 8 9**?:

```
main()
{ typedef struct fecha { int dia,mes,anyo; } fec;
  fec a={0,0,0};
  printf("Dame fecha: ");
  fun(1,&a.dia); fun(2,&a.mes); a.anyo=fun(3,&a.mes);
}
int fun(int tipo, int *dato)
{ printf("%s: ", tipo==1?"Dia":(tipo==2?"Mes":"Año"));
  scanf("%d", dato); return *dato;
```

```
}
```

01) **a.dia=7 a.mes=8 a.anyo=9**
02) **a.dia=7 a.mes=9 a.anyo=9**
04) **a.dia=7 a.mes=8 a.anyo=0**
08) **Da error de compilación**

---

5. ¿Qué imprime el siguiente programa?

```
int fun(int x, int *p)
{  x++; ++*p; return x;  }

main()
{  int n=10, m=20;
   printf("%d %d %d", fun(n,&m), n, m);
}
```

01) 11 11 20
02) 11 10 20
04) 11 10 21
08) 11 11 21

---

6. Dada la siguiente función:

```
int funarg(int *a)
{
        *a = *a+5;
        return (*a)*(*a);
}
void main(void)
{
        int a=4, b=7;
        printf("La funcion devuelve %d --> ", funarg(&b));
        printf("La variable vale %d\n", b);
}
```

Al ejecutar este programa nos dará:

01) La función devuelve 144 –> La variable vale 12
02) La función devuelve   81 –> La variable vale 7
03) La función devuelve 144 –> La variable vale 7
04) La función devuelve   81 –> La variable vale 12

---

7. El siguiente programa, ¿qué imprime en pantalla?

```
void fun1(int *a, int b);

main( )
{
   int a=10, b=20;
   fun1(&a, b);
   printf("a=%d, b=%d", a, b);
}
void fun1(int *a, int b)
{
  *a=30;
```

```
    b=50;
}
```

01) a=31, b=51
02) a=30, b=50
04) a=30, b=20
08) a=31, b=21

---

8. El siguiente programa, ¿qué imprime en pantalla?

```
void funcion(int *num1, int num2)
{
      *num1=10;
      num2=20;
      printf("En la función num1=%d, num2=%d", *num1, num2);
}

void main( )
{
      int num1=1, num2=2;
      funcion(&num1, num2);
      printf(" y en el main num1=%d, num2=%d\n", num1, num2);
}
```

01) En la función num1=10, num2=20 y en el main num1=1, num2=20
02) En la función num1=10, num2=20 y en el main num1=1, num2=2
04) En la función num1=10, num2=20 y en el main num1=10, num2=20
08) En la función num1=10, num2=20 y en el main num1=10, num2=2

ASIGNACIÓN DINÁMICA DE MEMORIA

1. En el programa siguiente, dada la estructura **struct alumno**, indicar qué sentencias se han de sustituir por el COMENTARIO en la función **leer_datos** para que se lea el nombre y la nota de un alumno, si dicha función recibe como argumento un puntero a dicha estructura.

```
struct alumno  {  char *nombre;   int nota;  };
void leer_datos (struct alumno *alu);
main()
{
  struct alumno al;
  leer_datos(&al);
}
void leer_datos (struct alumno *alu)
{  char nom[80];  puts("Nombre: ");  gets(nom);
  /* COMENTARIO */
  puts("Nota: "); scanf("%d",&alu->nota);
}
```

01) strcpy(alu->nombre,nom);
02) alu->nombre=(char *)malloc((strlen(nom)+1));
      if (alu->nombre==NULL) { puts("No hay memoria"); exit(0); }
      strcpy(alu->nombre,nom);
04) gets(alu->nombre);
08) alu->nombre=(char *)malloc((strlen(nom)+1));
      if (alu->nombre==NULL) { puts("No hay memoria"); exit(0); }
      alu->nombre=nom);

---

2. Dado el array dinámico **a** de 5 elementos de tipo **int**, indicar qué instrucciones se deben sustituir por el COMENTARIO para ampliar a 10 su número de elementos, sin perder la información que ya existía almacenada en el array original y rellenando a **0** los nuevos elementos. Suponer que no existirán problemas de falta de memoria.

```
main()
{  int *a=(int *)malloc(5*sizeof(int)), *b, i;
   for (i=0; i<5; i++) a[i]=i*2;     // Estos son los valores iniciales del array
   /* COMENTARIO A SUSTITUIR */
   for (i=0; i<10; i++) printf("%d  ", a[i]);
   free(a);
}
```

01) b = (int *)malloc(10*sizeof(int)); for (i=0; i<5; i++) b[i]=a[i];
    for (i=5; i<10; i++) b[i]=0;
    free(a); a=b;
02) b = (int *)calloc(10, sizeof(int)); for (i=0; i<5; i++) b[i]=a[i];
    free(a); a=b;
04) b = (int *)malloc(10*sizeof(int)); for (i=0; i<5; i++) b[i]=a[i];
    for (i=5; i<10; i++) b[i]=0;
    free(a); b=a;
08) a = (int *)realloc(a, 10*sizeof(int)); for (i=5; i<10; i++) a[i]=0;

---

3. Con las siguientes declaraciones, indique qué instrucciones pueden ser correctas:

```
struct ficha
{ char nom[40];
  int edad;
};
struct ficha *pun=NULL;
```

01) pun=(struct ficha *)malloc(sizeof(struct ficha *));
02) pun=(struct ficha *)malloc(4*sizeof(struct ficha));
04) pun=(struct ficha *)calloc(3,sizeof(struct ficha));
08) pun=(struct ficha *)realloc(pun,5*sizeof(struct ficha));

---

4. Con las siguientes declaraciones, indique qué instrucción es INCORRECTA.

```
struct ficha
{
  char nom[40];
  int edad;
};
struct ficha *pun=NULL;
```

01) pun=(struct ficha *) malloc(4*sizeof(struct ficha));
02) pun=(struct ficha *) calloc(3,sizeof(struct ficha));
04) pun=(struct ficha *) realloc(pun,5*sizeof(struct ficha));
08) pun=(struct ficha *) malloc(sizeof(struct ficha *));

---

5. Con las siguientes declaraciones, indique qué instrucción es **INCORRECTA**:

```
struct producto_t
{
      char nombre[40];
```

```
    int cantidad;
    float precio;
};

struct producto_t *puntero=NULL;
```

01) puntero=(struct producto_t *)malloc(sizeof(struct producto_t *));
02) puntero=(struct producto_t *)malloc(4*sizeof(struct producto_t));
04) puntero=(struct producto_t *)calloc(3,sizeof(struct producto_t));
08) puntero=(struct producto_t *)realloc(puntero,5*sizeof(struct producto_t));

PASO DE ARRAYS A FUNCIONES

1. Dada la función **fun** siguiente, y si además se ha declarado **int x[ ]={3,6,9,2};**

```
int fun(int *x, int n)
{
  int a=0;
  while(n-->0)  a+=*x++;
  return a;
 }
```

01) La instrucción **printf("%d",fun(x,4));** imprime 20
02) La instrucción **printf("%d",fun(x+1,3));** imprime 17
04) La instrucción **printf("%d",fun(x++,3));** imprime 17
08) La instrucción **printf("%d",fun(&x[0],sizeof(x)/sizeof(int)));** imprime 20

---

2. La función:

```
int fun(int *x, int nmax)
 {  int i=0,j=0;
    for (i=1; i<nmax; i++)
      if ( *(x+i)>*(x+j) )  j=i;
   return (j);
 }
```

01) Retorna el valor de x[i].
02) Retorna el valor de x[j].
04) Retorna la posición del mayor elemento del array x.
08) Retorna el valor del mayor elemento del array x.

---

3. ¿Qué funciones son correctas para que el siguiente programa permita intercambiar los dos valores del array **a**?

```
main()
{
  int a[2]={10,6};
  fun(a);
}
```

01) void fun(int a[ ])
    { int aux = a[0]; a[0]=a[1]; a[1]=aux; }
02) void fun(int *a)
    { int aux = *a; *a = *(a+1); *(a+1)=aux; }
04) void fun(int *a)

```
    { int aux = *a; *a = *(++a); *(++a)=aux; }
08) void fun(int *x)
    { int aux = x[0]; x[0] = *(x+1); *(x+1)=aux; }
```

---

4. En la siguiente función, indique qué instrucciones pueden ser correctas:

```
int fun(char cad[ ], int n)
{
  // instrucciones
}
```

```
01) printf("%c", *cad++);
02) printf("%c", *(cad)++);
04) printf("%c", *(cad+3));
08) printf("%c", *(cad+3)='W');
```

PASO DE ESTRUCTURAS A FUNCIONES

1. Indicar que sentencias hay que sustituir por el comentario en la función **fun** para que el siguiente programa imprima todos los datos del array **a.**

```
struct alu
{ char nombre[30];
  float nota;
};
void fun(struct alu *s, int num);
main()
{
  struct alu a[3]= {{"Juan", 7.5}, {"Luis", 5.5}, {"Ana", 8.25}};
  fun(a,3);
}
void fun(struct alu *s, int num)
{
  int i;
  for(i=0; i<num; i++)
  /*  COMENTARIO A SUSTITUIR   */
}
```

```
01) printf("Nombre: %s, Nota: %.2f\n", s[i]->nombre, s[i]->nota);
02) { printf("Nombre: %s, Nota: %.2f\n", s->nombre, s->nota);  s++;  }
04) printf("Nombre: %s, Nota: %.2f\n", s->nombre, s->nota);
08) printf("Nombre: %s, Nota: %.2f\n", s[i].nombre, s[i].nota);
```

---

2. Indicar qué resultado produce el siguiente programa:

```
struct t
{  int a;
   char b;  };
void fun1 (struct t *p);
void fun2 (struct t q);
main( )
{   int i;
    struct t reg;
    reg.a=10;   reg.b='D';
    for( i=0; i<3; i++)
    {  fun1(&reg);   fun2(reg);
```

```
        printf("%d,%c\n",reg.a,reg.b);    }
}
void fun1 (struct t *p)
{   if (p->a >= 12) p->a++;
    else p->b++;   }
void fun2 (struct t q)
{  q.a=20;   q.b++;   }
```

01) 20,F
21,G
22,H

02) 10,F
11,G
12,H

04) 20,E
21,E
22,E

08) 10,E
10,F
10,G

---

3. ¿Qué imprime el programa siguiente?

```
typedef  struct
{   char *nombre;
    float nota;
} reg;
void transferir (reg *p);
main( )
{   reg  alumno = {"Luis Sanchez", 7.50};
    transferir (&alumno);
    printf ("%s, %.2f \n", alumno.nombre, alumno.nota);
}
void transferir (reg *p)
{   p->nombre = "Juan Garcia";  p->nota = 9.80;   return;   }
```

01) Luis Sanchez, 7.50
02) Juan Garcia, 9.80
04) [valor basura], [valor basura]
08) Juan, 9.80

---

4. ¿Qué imprime el siguiente programa?

```
typedef struct ficha
{ char nom[40]; int edad; } tficha;
void fun1(tficha x)
{  strcpy(x.nom,"SARA"); x.edad++;  }
void fun2(tficha *x)
{  x->nom[0]='A'; x->edad++;  }
main()
{  tficha dato={"JUAN", 19};
   fun2(&dato); fun1(dato);
   printf("%s %d", dato.nom, dato.edad);
}
```

01) AARA 21
02) SARA 20
04) AUAN 19
08) AUAN 20

---

5. En el programa siguiente, dada la estructura **struct alumno**, indicar qué sentencias se han de sustituir por el COMENTARIO en la función **leer_datos** para que se lea desde teclado el nombre y la nota de un alumno.

```
struct alumno  {  char *nombre;   int nota;  };
void leer_datos (struct alumno *alu);
main()
```

```
{
  struct alumno al;
  leer_datos(&al);
}
void leer_datos (struct alumno *alu)
{  char nom[80];  puts("Nombre: ");  gets(nom);
  /* COMENTARIO */
  puts("Nota: "); scanf("%d",&alu->nota);
}
```

```
01) strcpy(alu->nombre,nom);
02) alu->nombre=(char *)malloc((strlen(nom)+1));
     if (alu->nombre==NULL) { puts("No hay memoria"); exit(0); }
     strcpy(alu->nombre,nom);
04) gets(alu->nombre);
08) alu->nombre=(char *)malloc((strlen(nom)+1));
    if (alu->nombre==NULL) { puts("No hay memoria"); exit(0); }
    alu->nombre=nom;
```

---

6. En el programa siguiente, dada la estructura **struct producto_t**, indicar qué sentencias se han de sustituir por el COMENTARIO en la función **rellenar_stock** para que se rellene desde teclado los datos del producto.

```
struct producto_t
{
      char *nombre;
      int cantidad;
      float precio;
};

void rellenar_stock (struct producto_t *producto)
{
      char nombre[80];
      printf("Nombre: ");
      gets(nombre);

      /* COMENTARIO */

      printf("Cantidad: ");
      scanf("%d", &producto->cantidad);
      printf("Precio: ");
      scanf("%f", &producto->precio);
}

void main()
{
      struct producto_t producto;
      rellenar_stock(&producto);
}
```

```
01) strcpy(producto->nombre,nombre);

02) producto->nombre=(char *)malloc((strlen(nombre)+1));
    if (producto->nombre==NULL)
    {
       printf("No hay memoria"); exit(0);
    }
    strcpy(producto->nombre,nombre);

04) gets(producto->nombre);
```

08) producto->nombre=(char *)malloc((strlen(nombre)+1));
if (producto->nombre==NULL)
{
printf("No hay memoria"); exit(0);
}
producto->nombre=nombre;

---

7. ¿Qué imprime el siguiente programa?

```
typedef struct
{
	char nombre[40];
	int cantidad;
	float precio;
} producto_t;

void funcion1(producto_t x)
{
	strcpy(x.nombre,"LAPIZ");
	x.cantidad++;
	x.precio/=2;
}

void funcion2(producto_t *x)
{
	x->nombre[0]='T';
	x->cantidad++;
	x->precio/=2;
}

void main()
{
	producto_t dato={"Borrador", 19, 2.0};
	funcion1(dato);
	funcion2(&dato);
	printf("%s %d %.2f\n", dato.nombre, dato.cantidad, dato.precio);
}
```

01) Borrador 19 2.00
02) LAPIZ 20 1.00
04) TAPIZ 21 0.50
08) Torrador 20 1.00

RECURSIVIDAD

1. Dada la función:

```
int fact(int x)
{
  if (x==0) return (1);
  x = x*fact(x-1);
  printf (" %d",x);
  return (x);
}
```

01) Al llamar a fact(3), escribe en pantalla **1 2 6**

02) Al llamar a fact(2), escribe en pantalla **1 2**
04) Al llamar a fact(1), escribe en pantalla **1**
08) Al llamar a fact(0), escribe en pantalla **1**

---

2. Indicar qué imprime esta función cuando se ejecuta la instrucción **fun(5);**

```
void fun(int x)
{
  if (x==1) return;
  printf("%d", x--);
  fun(x);
  printf("%d", x);
}
```

01) 54321234
02) 43214321
04) 543211234
08) 5432101234

---

3. ¿Qué se imprime al ejecutar este programa?:

```
void recur(char **);
main()
{
 char *nom[10]={"hoy","es","lunes",NULL};
 recur(nom);
}
void recur(char **pun)
{
 if (*pun==NULL) return;
 recur(pun+1);
 printf("%s  ",*pun);
}
```

01) lunes es hoy
02) hoy es lunes
04) senul se yoh
08) hoy es lunes NULL

---

4. Al ejecutar el siguiente programa:

```
void fun(void);

main()
{ fun(); }

void fun(void)
{
 int n;
 scanf("%d",&n);
 if (n==0) return;
 fun();
 printf("%d  ",n);
}
```

01) Si se introduce **0** seguido de <intro> imprime 0
02) Si se introduce **0 4** seguido de <intro> no imprime nada
04) Si se introduce **1 2 3 4 0** seguido de <intro> imprime 4 3 2 1
08) Si se introduce **4 3 0** seguido de <intro> imprime 0 3 4

---

5. En el siguiente programa:

```
void escribir (char *n);
main()
{
  escribir("Hola");
}
```

Indicar cuál de las siguientes funciones es la que presenta por pantalla la cadena referenciada por **n**, pero escrita al revés.

01) void escribir (char *n)
{ if (*n!=0) { escribir (n+1); putchar (*n); } }
02) void escribir (char *n)
{ if (*n!=0) { escribir (n++); putchar (*n); } }
04) void escribir (char *n)
{ if (*n!='\0') { escribir (n+1); putchar (*n); } }
08) void escribir (char *n)
{ if (*n!=0) { escribir (n+1); putchar (n); } }

---

6. Indicar qué sentencias hay que sustituir por el **COMENTARIO** para que la función **fun()** efectue el cambio de base del número entero **num** a la base **b** (base menor que 10) e imprima el número resultante.
Ejemplos: si num=12 y b=2, imprimirá 1100
si num=12 y b=8, imprimirá 14

```
void fun(int num, int b)
{
  /* COMENTARIO A SUSTITUIR  */
}
```

01) **if (num%base>0)**
fun(num/base, base);
printf("%d", num%base);
02) **if (num/base>0)**
fun(num/base, base);
printf("%d", num%base);
04) **if (num/base==0)**
fun(num/base, base);
printf("%d", num%base);
08) **if (num/base<=0)**
printf("%d", num%base);
fun(num/base, base);

---

7. Indicar qué sentencias hay que sustituir por el comentario para que el siguiente programa imprima "**casa coche mueble puerta".**

```
void impr(char **pcad)
{  /* COMENTARIO A SUSTITUIR */
   printf("%s ",*pcad++);
   impr(pcad);
}
main()
{  char *cad[ ]={"casa","coche","mueble","puerta",NULL};
   impr(cad);
```

}

01) if (*pcad !=NULL) return;
02) while (*++pcad!=NULL);
04) if (pcad==NULL) return;
08) if (*pcad==NULL) return;

---

8. ¿Qué imprime el siguiente programa?

```
void fun(int i);
char p[ ][10]={"lunes","martes","miercoles","jueves","viernes","sabado","domingo"};
main()
{
  fun(0);
}
void fun(int i)
{
 if (strcmp(p[i],"sabado")==0) return;
 fun(i+1);  printf("%s  ", p[i]);
}
```

01) lunes martes miercoles jueves viernes sabado
02) lunes martes miercoles jueves viernes sabado domingo
04) lunes martes miercoles jueves viernes
08) viernes jueves miercoles martes lunes

---

9. ¿Qué imprime el siguiente programa?

```
void fun(int i);
char p[ ][10]={"lunes","martes","miercoles","jueves","viernes","sabado","domingo"};
main()
{
  fun(0);
}
void fun(int i)
{
 if (i==6) return;
 fun(i+1);  printf("%s  ", p[i]);
}
```

01) lunes martes miercoles jueves viernes sabado
02) lunes martes miercoles jueves viernes sabado domingo
04) sabado viernes jueves miercoles martes lunes
08) viernes jueves miercoles martes lunes

---

10. Indicar qué sentencias hay que sustituir por el comentario para que el siguiente programa imprima la representación en binario del numero entero **n** que se recibe como argumento.

```
void imprime_binario ( int n)
{
  if (n>= 2)
  {
    /*  COMENTARIO A SUSTITUIR  */
```

```
  }
  else printf(" %d" , n);
}
main()
{
   imprime_binario(12);
}
```

01) printf(" %d" , n%2); imprime_binario (n/2);
02) imprime_binario (n/2); printf(" %d" , n%2);
04) imprime_binario (n/2); printf(" %d" , n/2);
08) imprime_binario (n%2); printf(" %d" , n%2);

---

11. ¿Qué se imprimirá al ejecutar este programa?

```
void fun(int *x, int n)
{  if (n==0) return;
   n--;
   printf("  %d", *(x+n));
   fun(x,n);
}

main()
{  int x[6]={25,4,3,8,10,11};
   fun(x,6);
}
```

01) 25 4 3 8 10 11
02) 4 3 8 10 11
04) 11 10 8 3 4 25
08) 10 8 3 4 25

---

12. ¿Qué resultado imprime la función **main** tras la llamada a la función recursiva?

```
int recursiva(int a, int b);

void main(void)
{      int x = 20, y=10, result=0;
       result = recursiva(x, y);
       printf("%d\n", result);
}

int recursiva(int a, int b)
{      if (b == 1)  return a;
       else   return  a + recursiva(a, b-1);
}
```

01) 180
02) 20
04) 200
08) 210

---

13. ¿Qué imprime el siguiente programa?

```
void func(int n)
{
  printf("%d ", n);
```

```
  if (n==0)
     return;
  func(n-1);
  printf("%d ", n);
}

main()
{
  func(4);
}
```

01) 4 4 3 3 2 2 1 1 0
02) 4 3 2 1 0 0 1 2 3 4
04) 4 3 2 1 0 1 2 3 4
08) 4 3 2 1 0

---

14. ¿Qué devuelve la función **fun** si se le pasa como argumento el valor **4**?:

```
int fun(int x)
{
  if (x==0)
  {
      return (1);
  }

  x += fun(x-1);

  return (x);
}
```

01) 10
02) 1
04) 11
08) 4

ARGUMENTOS EN LÍNEA DE ÓRDENES

1. Si el resultado de compilar y enlazar este programa se llama **prog**, ¿qué imprimirá la orden **prog lunes martes** ?

```
main(int argc, char **argv)
{
  printf("%s", *++argv);
  printf("%c%s", argv[0][0], *argv);
}
```

01) Nada porque da error de compilación
02) lunesmartes
04) lunesllunes
08) luneslmartes

---

2. El siguiente programa, una vez compilado, se ejecutaria así: **./prog numero cadena**

```
main( int argc, char *argv[ ])
{
 int n ;
```

```
 if (argc < 3)
 { printf("ERROR. Forma de uso: %s Numero Cadena\n", argv[0]);  exit(0); }
 n=atoi(argv[1]);
 if (strlen(argv[2])<=n) puts(argv[2]);
 else  while(n>=1) putchar(argv[2][--n]);
}
```

01) Si ponemos **./prog 4 HOLA** imprime **HOLA**
02) Si ponemos **./prog 6 HOLA** imprime **HOLAAA**
04) Si ponemos **./prog 2 HOLA** imprime **OH**
08) Si ponemos **./prog 0 HOLA** imprime **ERROR. Forma de uso: prog Numero Cadena**

---

3. En un programa con el siguiente encabezamiento: **main(int argc, char *argv[ ])**
y que, una vez compilado con **gcc**, genera el programa ejecutable **a.out**, si damos la orden:
./a.out 543 mas 789

01) la instrucción **printf("%d+%d=%d", argv[1], argv[3], argv[1]+argv[3]);** imprime 543+789=1332
02) la instrucción **printf("%s%s=%s", argv[1], argv[2], argv[3]);** imprime 543mas789=1332
04) la instrucción **printf("%d%s%d=%d", argv[1], argv[2], argv[3], argv[1]+argv[3]);** imprime 543+789=1332
08) la instrucción **printf("%c,%s", argv[3][1], &argv[1][1]);** imprime 8,43

---

4. En el siguiente programa, indicar qué sentencias hay que sustituir por el comentario, para que al ejecutar el programa se visualicen todos los argumentos que se introduzcan por la linea de órdenes, incluyendo el nombre del fichero ejecutable.

```
void imprime(char *p);
main(int argc, char*argv[ ])
{
  printf("El numero de argumentos es: %d\n", argc-1);
  /* COMENTARIO A SUSTITUIR  */
}
void imprime(char *p)
{
  printf("%s\n", p);
}
```

01) **while (*argv) imprime(*argv++);**
02) **while (*argv++) imprime(*argv);**
04) **while (--argc >0) imprime(*argv++);**
08) **while (argc-- >0) imprime(*argv++);**

---

5. Suponiendo que se ha dado la orden de compilación **gcc prog.c**, indicar qué imprime el siguiente programa cuando se ejecuta la orden: **./a.out lunes martes miercoles**

```
main(int argc, char *argv[ ])
{
 int i=0;
 while (argv[++i])
    printf("  %s", argv[i]);
 printf("  %s",argv[i-1]);
}
```

01) a.out lunes martes miercoles
02) lunes martes miercoles
04) martes miercoles
08) lunes martes miercoles miercoles

---

6. Suponiendo que el programa ejecutable que se obtiene al compilar este programa se llama **PROG.EXE**.

```
main(int argc, char *argv[ ])
{
   int c=0;
   if ( (argc==3) && (!strcmp(argv[1],"examen")) )
   {  c=atoi(argv[2]);
      printf("c=%d",c);
   }
   else printf("c=%d", c);
}
```

01) al dar la orden **PROG examen 5** imprime c=0
02) al dar la orden **PROG examen 5** imprime c=5
04) al dar la orden **PROG 5 examen** imprime c=0
08) al dar la orden **PROG examen 5** imprime c=2

---

7. Suponiendo que el programa **prog.c** se ha compilado sin errores en el laboratorio con la orden **gcc prog.c,** indicar qué imprime el siguiente programa cuando se ejecuta la orden:
./a.out 24 de junio de 2014 examen

```
main(int argc, char *argv[ ])
{
  while (*++argv)
     printf(""%s"", *argv);
  printf(""%s"", *--argv);
  printf(""%s"", *--argv);
}
```

01) a.out 24 de junio de 2014 examen
02) 24 de junio de 2014 examen
04) 24 examen 2014
08) 24 de junio de 2014 examen examen 2014

---

8. Suponiendo que se ha dado la orden de compilación **gcc prog.c -o prog** y ha funcionado correctamente, indicar qué imprime el siguiente programa cuando en el laboratorio se ejecuta la orden: **./prog 15 de mayo**

```
main(int argc, char **argv)
{ while(argc)
     printf("  %s", argv[--argc]);
  printf("  %s", *++argv);
}
```

01) mayo de 15 prog
02) mayo de 15
04) mayo de 15 prog 15
08) valores basura

---

9. En un programa de nombre **prog** con argumentos en línea de órdenes, al ejecutar la orden
./prog lunes martes la instrucción:

01) **printf("%c", argv[2][1]);** imprime m
02) **printf("%c", argv[1][1]);** imprime u
04) **printf("%s", &argv[1][2]);** imprime nes

08) **printf("%s", argv[1]);** imprime lunes

---

10. Queremos que al compilar el programa **prog.c** se obtenga un fichero ejecutable **prog.exe** con posibilidad de argumentos en la línea de órdenes, de modo que al ejecutar la orden ./**prog.exe 1 1** el programa imprima **iguales**, y al ejecutar ./**prog.exe 1 2** imprima **distintos**. Indicar qué sentencias hay que sustituir por el comentario en el programa **prog.c**

```
main(int argc, char *argv[])
{
  if (argc!=3) { puts("Mal numero de argumentos"); exit(1); }
  /*  COMENTARIO A SUSTITUIR  */
}
```

01) **if (argv[1]==argv[2]) puts("iguales"); else puts("distintos");**
02) **if (!strcmp(argv[1], argv[2])) puts("iguales"); else puts("distintos");**
04) **if (argv[1][0]==argv[2][0]) puts("iguales"); else puts("distintos");**
08) **if (*argv[1]==*argv[2]) puts("iguales"); else puts("distintos");**

---

11. Si el resultado de compilar y enlazar este programa se llama **prog**, ¿qué imprimirá la orden **./prog lunes martes** ?

```
int main(int argc, char **argv)
{
   printf("%s ", argv[1]); argv++;
   printf("%s %s", *argv, argv[1]);
}
```

01) lunes prog lunes
02) lunes lunes martes
04) lunes lunes lunes
08) lunes prog martes

---

12. Dado el siguiente programa cuyo ejecutable se llamará **p**:

```
int main(int argc, char *argv[ ])
{
   int i=0;
   if (argc > 1)
   {
      for(i=1; i<argc; i++)
      {
         if ( strcmp(argv[i],"rojo")==0 &&  ( i!=2 ) )
            printf("\n\n\t Escribimos en ROJO\n\n");
         else if ( strcmp(argv[i],"verde")==0 )
            printf("\n\n\t Escribimos en VERDE\n\n");
         else if ( strcmp(argv[i],"azul")==0  && ( i!=1) )
            printf("\n\n\t Escribimos en AZUL\n\n");
      }
   }
   else  printf("\n\n\tNo seas aburrido, introduce argumentos\n\n");
 }
```

01) Al ejecutarlo con **./p azul** imprime:
No seas aburrido, introduce argumentos
02) Al ejecutarlo con **./p verde rojo** imprime:
Escribimos en VERDE

04) Al ejecutarlo con **./p verde rojo** imprime:
Escribimos en VERDE
Escribimos en ROJO

08) Al ejecutarlo con **./p azul verde** imprime:
Escribimos en AZUL
Escribimos en VERDE

---

13. Si el resultado de compilar y enlazar este programa se llama **prog**, ¿qué imprimirá la orden **./prog lunes martes** ?

```
void main(int argc, char **argv)
{
   printf("Argumentos: %d son %s %s", argc, argv[1], argv[2]);
}
```

01) Argumentos: 3 son prog lunes
02) Argumentos: 3 son lunes martes
04) Argumentos: 2 son prog lunes
08) Argumentos: 2 son lunes martes

FICHEROS

1. Se tiene en el disco un fichero de nombre **t1.txt** que contiene únicamente los 4 caracteres siguientes: **velo**
¿Qué se obtiene en pantalla al ejecutar el siguiente programa?

```
main()
{
  char car;
  FILE *punf;
  punf=fopen("t1.txt","r");
  car=fgetc(punf);
  while (!ferror(punf) && !feof(punf))
  {
    putchar(car+1); car=fgetc(punf);
  }
  fclose(punf);
}
```

01) velo
02) udkn
04) wfmp
08) elo

---

2. Con el siguiente trozo de programa, diga qué instrucciones pueden ser correctas para abrir un archivo de nombre **fichero.txt** para escritura:

```
main()
{
  FILE *p;
  char cad[ ]="fichero.txt";
  char op[ ]="w";
  /* Instrucción a poner */
```

01) p=fopen(cad,"w");
02) p=fopen(cad,op);

04) p=fopen(&fichero.txt,"w");
08) p=fopen(&cad[0],op);

---

3. Suponiendo que se ha declarado: **FILE *pf;**
Indique qué instrucciones son correctas para abrir el fichero **datos.dat** para añadirle nuevos valores al final.

01) if (pf=fopen("datos.dat","a")) { puts("error"); exit(0); }
02) char nom[ ]="datos.dat";
if (!(pf=fopen(nom,"w")) { puts("error"); exit(0); }
04) char nom[ ]="datos.dat", x[ ]="a";
if (!(pf=fopen(nom,x)) { puts("error"); exit(0); }
08) char *nom="datos.dat";
if (!(pf=fopen(&nom[0],"a")) { puts("error"); exit(0); }

---

4. Se tiene en el disco un fichero de nombre **t1.txt** que contiene únicamente los 4 caracteres siguientes: **velo**
¿Qué se obtiene en pantalla al ejecutar el siguiente programa?

```
main()
{
  int n=0:
  char car;
  FILE *punf;
  punf=fopen("t1.txt","r");
  while (!ferror(punf) && !feof(punf))
  { car=fgetc(punf);
    n++;
  }
  fclose(punf);
  printf(“%d”, n);
}
```

01) 4
02) 5
04) 3
08) un valor basura

---

5. Si al ejecutar el programa indicado introducimos por teclado la palabra **"mifi"**, y existe un fichero de texto de nombre MIFI.TXT que contiene únicamente la palabra **"murcielago"**, ¿que resultado se imprime?

```
main( )
{   int con=0;  char nom[10];  FILE *p;
    gets(nom);
    strcat(nom,".txt");
    if ((p=fopen(nom,"r"))==NULL)
      {  puts("mal"); exit(1);  }
    while(!feof(p))
    {   nom[0]=fgetc(p);
        switch (nom[0])
        {  case 'a': case 'e': case 'i': case 'o': case 'u': ++con;  }
    }
    printf("%d",con);
    fclose(p);
}
```

01) Nada
02) 5
04) 6
08) 1

---

6. Si ejecutamos el siguiente programa:

```
main( )
{  char nom[10],  *op="wb";  FILE *p;
   gets(nom);
   p=fopen(nom,op);
   fputc('A',p);
   fclose(p);
}
```

01) Hay que introducir el nombre del archivo que se quiere crear y cuyo contenido va a ser el caracter A
02) No se puede abrir el archivo
04) Hay que introducir el nombre del archivo que se quiere crear y luego los caracteres a grabar en binario
08) Da error de compilacion

---

7. Suponiendo que en un programa con argumentos en línea de órdenes se ha pasado como primer argumento el texto "**mifi"**, indicar qué instrucciones son correctas para abrir en modo escritura el fichero de nombre **mifi**.

01) FILE pun=fopen(“mifi”,”w”);
02) FILE *pun=fopen(mifi,”w”);
04) FILE *pun=fopen("argv[1]",”w”);
08) FILE *pun=fopen(argv[1],”w”);

---

8. Con las siguientes declaraciones: **char cad[40]; FILE *pf=fopen(“mifi”,”w+”);**
Indicar qué instrucciones pueden ser correctas:

01) fgets(“Una línea”,39,pf);
02) fgets(cad+1,30,pf);
04) fputs(“Una línea”,pf);
08) fputs(“Una línea”,20,pf);

---

9. En el siguiente programa, indicar qué sentencias hay que sustituir por el comentario, para que al ejecutar el programa imprima **HOY ES MARTES** si desde el teclado se introduce **HOY ES MARTES [Intro]**. Para finalizar el programa pulsamos **[Intro]**.

```
main()
{ int i=0;
  char linea[25];
  /* COMENTARIO A SUSTITUIR  */
}
```

01) while (fgets(linea, 10, stdin)!=NULL && linea[0]!='\n')
    fputs(linea,stdout);
02) while (fgets(linea, 5, stdin)!=NULL && linea[0]!='\n')
    fputs(linea,stdout);
04) while (fgets(linea, 15, stdin)!=NULL && linea[0]!='\n')
    fputs(linea,stdout);
08) while (fgets(linea, 25, stdin)!=NULL && linea[0]!='\n')
    fprintf(stdout,"%s",linea);

---

10. Indique cómo podría ser una función que permita abrir un fichero cuyo nombre se da como primer argumento, para añadirle al final una línea de texto guardada en una cadena cuyo nombre constituya el segundo argumento.

01) void fun(char *nomfich,"a")
{ FILE *pun=fopen(nomfich,"a");
fputs(cad,pun); fputc('\n',pun);
fclose(pun); }
02) void fun(char *nomfich,char *cad)
{ FILE *pun=fopen(nomfich,"a");
fputs(cad,pun); fputc('\n',pun);
fclose(pun); }
04) void fun(char *nomfich,char cad[])
{ FILE *pun=fopen(nomfich,"a");
fputs(cad,pun); fputc('\n',pun);
fclose(pun); }
08) void fun(char *nomfich,char *cad)
{ FILE *pun=fopen(&nomfich[0],"a");
fputs(&cad[0],pun); fputs("\n",pun);
fclose(pun); }

---

11. ¿Qué imprime este programa?

```
main()
{ char nombres[][20]={"Escuela","Universitaria","Politecnica"};
  int i,n=sizeof(nombres)/20, a=strlen(nombres[1]);
  FILE *pf;
  if ((pf=fopen("fnom","w+")) == NULL)  { perror(" No se puede abrir fichero"); exit(0); }
  for (i=0; i<n && !ferror(pf); i++)  fputs(*(nombres+i) , pf);
  rewind(pf);  i=0;
  fgets(nombres[i],a,pf);
  while(!feof(pf))
  { printf("%s  ",nombres[i++]);  fgets(nombres[i],a,pf);  }
  fclose(pf);
}
```

01) Escuela Universitaria Politecnica
02) EscuelaUnive rsitariaPoli
04) Escuela Escuela Escuela
08) EscuelaUniversitariaPolitecnica

---

12. Un fichero que contiene estructuras de tipo **struct pp** acaba de ser abierto para lectura y es referenciado mediante un puntero **pf**. Si **x** es una variable de ese tipo de estructuras, indique qué instrucciones son correctas para leer el segundo registro del fichero:

01) fseek(pf,2,SEEK_SET); fread(&x,sizeof(struct pp),1,pf);
02) fseek(pf,1,SEEK_SET); fread(&x,sizeof(struct pp),1,pf);
04) fseek(pf,2*sizeof(struct pp),SEEK_SET); fread(&x,sizeof(struct pp),1,pf);
08) fseek(pf,1*sizeof(struct pp),SEEK_SET); fread(&x,sizeof(struct pp),1,pf);

---

13. Indicar qué grupo de sentencias hay que sustituir por el comentario, para que se pueda leer la información del fichero **AAA** y visualizarla.

```
main( )
{   int  y[5], i, x[5]={1001,1002,1003,1004,1005};
    FILE *pf;
    if ((pf=fopen("AAA","wb"))==NULL) {  perror("Error de apertura");  exit(0); }
    fwrite(x,sizeof(x),1,pf);
    fclose(pf);
    if ((pf=fopen("AAA","rb"))==NULL) {  perror("Error de apertura");   exit(0); }
    /*  Comentario a sustituir      */
    fclose(pf);
```

}

01) fread(y,sizeof(y),1,pf);
for(i=0;i<5;i++) printf("%d ",y[i]);
02) fread(&y[0],sizeof(y[0]),5,pf);
for(i=0;i<5;i++) printf("%d ",y[i]);
04) fread(&y[0],sizeof(y),1,pf);
for(i=0;i<5;i++) printf("%d ",y[i]);
08) while (!feof(pf)) { i=fgetc(pf); printf("%d ",i); }

---

14. Un fichero cuyo puntero de referencia es **FILE *pf;** ha sido abierto en modalidad de lectura y está trabajando con estructuras de tipo **struct pp**. Si se ha declarado **struct pp x[10];**

01) fread(x,sizeof(struct pp),1,pf); lee un registro y lo mete en x[0].
02) fread(x+3,sizeof(struct pp),1,pf); lee un registro y lo mete en x[3].
04) fread(&x[3],sizeof(struct pp),1,pf); lee un registro y lo mete en x[3].
08) rewind(pf); printf("%ld",ftell(pf)); imprime el numero de bytes del fichero.

---

15. Se supone que en un programa existen las instrucciones siguientes:

int con=0; FILE *pun=fopen(nom,"r");

donde **nom** es la dirección de comienzo de la cadena de caracteres que contiene el nombre de un fichero. Ese fichero tiene grabados 5 registros correspondientes a unas estructuras que se van a ir leyendo de una en una sobre una variable **var** de 6 bytes.
¿Con qué valor queda la variable **con** según se ejecute el segmento A o el segmento B?

```
/* segmento A */              /* segmento B */
fread(&var,6,1,pun);          while(!feof(pun))
while(!feof(pun))             { fread(&var,6,1,pun);
{
  con++;                        con++;
  fread(&var,6,1,pun);        }
}
```

01) 0 en ambos casos
02) 6 en el A y 5 en el B
04) 5 en el A y 6 en el B
08) 5 en ambos casos

---

16. Dado el siguiente programa, indicar qué sentencias hay que sustituir por el **COMENTARIO** para que se visualice por pantalla el fichero de nombre **datos**.

```
main()
{ FILE *pf;
  int x[5], y[5], i=0;
  for (i=0; i<5; i++) x[i]=10*i;
  if ((pf=fopen("datos","w+"))==NULL)
     { puts("\n Error de escritura\n"); exit(0); }
  fwrite(x,sizeof(x),1,pf);
  rewind(pf);
  /* COMENTARIO A SUSTITUIR */
  fclose (pf);
}
```

01) fread(y,sizeof(y),1,pf);
for(i=0; i<5; i++) printf(" %d", y[i]);
02) fread(y,sizeof(y[0]),5,pf);
for(i=0; i<5; i++) printf(" %d", y[i]);
04) fread(&y[0],sizeof(y[0]),1,pf);

```
     for(i=0; i<5; i++) printf(" %d", y[i]);
08) for(i=0; i<5; i++)
     { y[i]=fgetc(pf); printf(" %d", y[i]); }
```

---

17. Dado el siguiente programa, indicar cuáles de los **COMENTARIOS** a sustituir son correctos.

```
typedef struct
{ char nombre[20]; int nota; } TIPOREG;

main()
{
  TIPOREG reg;
  TIPOREG talum[5] = {{"JUAN",5},{"ANA",7},{"LUIS",9},{"EVA",3},{"PACO",6}};
  FILE *fp = fopen("DATOS.DAT","wb");
  fwrite(talum, sizeof(TIPOREG), 5, fp);
  fclose(fp);
  fp = fopen("DATOS.DAT","rb");
  /*  COMENTARIO1 A SUSTITUIR  */
  printf("%d, %s \n", reg.nota, reg.nombre);
  /*  COMENTARIO2 A SUSTITUIR  */
  printf("%d, %s \n", reg.nota, reg.nombre);
  /*  COMENTARIO3 A SUSTITUIR  */
  printf("%d, %s \n", reg.nota, reg.nombre);
  /*  COMENTARIO4 A SUSTITUIR  */
  printf("%d, %s \n", reg.nota, reg.nombre);
  fclose(fp);
}
```

```
01) // COMENTARIO1 Lee el primer registro
     fread(&reg,sizeof(reg),1,fp);
02) // COMENTARIO2 Me posiciono en el penúltimo registro y lo leo
     fseek(fp,-2L * sizeof(reg),SEEK_END); fread(&reg,sizeof(reg),1,fp);
04) // COMENTARIO3 Me muevo dos registros hacia atrás y lo leo
     fseek(fp,-2L * sizeof(reg),SEEK_CUR); fread(&reg,sizeof(reg),1,fp);
08) // COMENTARIO4 Me sitúo después del primer registro y lo leo
     fseek(fp,1L * sizeof(reg),SEEK_SET); fread(&reg,sizeof(reg),1,fp);
```

---

18. Dado el siguiente programa, indicar qué sentencias hay que sustituir por el **COMENTARIO** para que se visualice por pantalla el contenido del fichero **datos** en modo texto.

```
main()
{ FILE *pf;
  char x[15], y[15]; int i=0;
  for (i=0; i<15; i++) x[i]='A'+i;
  if ((pf=fopen("datos","w+"))==NULL)
    { puts("\n Error de apertura \n"); exit(0); }
  fwrite(x,1,15,pf);
  rewind(pf);
  /* COMENTARIO A SUSTITUIR */
  fclose (pf);
}
```

```
01) fread(y,1,15,pf);
     for(i=0; i<15; i++)  printf("%c", y[i]);
02) fread(y,sizeof(y[0]),sizeof(y),pf);
     for(i=0; i<15; i++)  printf("%c", y[i]);
04) fread(&y[0],1,15,pf);
     for(i=0; i<15; i++)  printf("%d", y[i]);
```

08) for(i=0; i<15; i++)
{ y[i]=fgetc(pf); printf("%c", y[i]); }

---

19. Dadas estas declaraciones:

```
FILE *p=fopen("fich","w");
struct pp
{ char c[40];
  int n;
  char  *p;
} x[40];
```

Indique qué instrucciones son correctas para almacenar en el fichero de nombre **fich** el array **x.**

01) **fwrite(x,sizeof(x),1,p);**
02) **fwrite(x,sizeof(struct pp),40,p);**
04) **fwrite(&x,sizeof(x),1,p);**
08) **fwrite(&x[0],sizeof(x),1,p);**

---

20. Un fichero que contiene estructuras de tipo **struct pp** acaba de ser abierto para lectura y es referenciado mediante un puntero a FILE **pf**. Si **x** es una variable de ese tipo de estructuras, indique qué instrucciones son correctas para leer el penúltimo registro del fichero:

01) fseek(pf,-1,SEEK_END); fread(&x,sizeof(struct pp),1,pf);
02) fseek(pf,-1*sizeof(struct pp),SEEK_END); fread(&x,sizeof(struct pp),1,pf);
04) fseek(pf,-2*sizeof(struct pp),SEEK_END); fread(&x,sizeof(struct pp),1,pf);
08) fseek(pf,0,SEEK_END); fseek(pf,-2*sizeof(struct pp),SEEK_CUR); fread(&x,sizeof(struct pp),1,pf);

---

21. Un fichero que contiene estructuras de tipo **struct pp** acaba de ser abierto para lectura y es referenciado mediante un puntero **pf**. Indique qué instrucciones son correctas para obtener en la variable **n** el número de registros que tiene el fichero.

```
struct pp { int r; float s; double t; };
struct pp x;  int n;
```

01) n=0; fseek(pf,0,SEEK_SET); fread(&x,sizeof(struct pp),1,pf);
while (!feof(pf) && !ferror(pf)) { n++; fread(&x,sizeof(struct pp),1,pf); }
02) n=0; rewind(pf); while (fread(&x,sizeof(x),1,pf) == 1) n++;
04) n=0; rewind(pf);
while (!feof(pf) && !ferror(pf)) { fread(&x,sizeof(struct pp),1,pf); n++; }
08) fseek(pf,0,SEEK_END); n = ftell(pf)/sizeof(struct pp);

---

22. Indicar qué se imprime si, al ejecutar este programa, se introduce por teclado el texto:
Escuela Politécnica

```
main()
{
  char cad[4];
  fgets(cad,4,stdin);
  fputs(cad,stdout);
}
```

01) Escuela Politécnica
02) Esc
04) Esc (seguido de basura)
08) Escu

---

23. Si el resultado de compilar y enlazar este código fuente es el programa **prog**, indicar qué se imprime al ejecutar la orden: **./prog datos r**
Suponer que el contenido del fichero **datos** es el texto: **Examen de Programación**

```
main(int argc, char *argv[ ])
{  char cad[30];
   FILE *p=fopen(argv[1], argv[2]);
   if (p==NULL) printf("El fichero %s no existe\n", argv[1]);
   printf("Vamos a leer el fichero"); fscanf(p, "%s", cad);
   fclose(p);
}
```

01) El fichero datos no existe
Vamos a leer el fichero
02) El fichero datos no existe
Examen de Programación
04) Vamos a leer el fichero
Examen de Programación
08) Vamos a leer el fichero

---

24. Se dispone de un fichero compuesto por estructuras del tipo:

struct alumno { char nombre[30]; int nota; };

Indicar qué sentencias hay que sustituir por el comentario, para que la función **fun** devuelva el campo **nombre** de la estructura leída, donde **pf** es un puntero al fichero abierto en modo lectura y el entero **p** es el número de registro de la estructura a la que queremos acceder (el primer registro del fichero es el de número 0).

```
char *fun(FILE *pf, int p)
{  struct alumno  a;
   /*  COMENTARIO A SUSTITUIR  */
}
```

01) fseek(pf, (long)p * sizeof(struct alumno), SEEK_SET);
fread (&a, sizeof(struct alumno), 1 ,pf);
return a.nombre;
02) fseek(pf, (long)p * sizeof(struct alumno), SEEK_SET);
fread (&a, sizeof(struct alumno), 1 ,pf);
return a->nombre;
04) fseek(pf, (long)p * sizeof(struct alumno), 0);
fread (&a, sizeof(struct alumno), 1 ,pf);
return a.nombre;
08) fseek(pf, (long)p * sizeof(struct alumno), SEEK_SET);
fread (a, sizeof(struct alumno), 1 ,pf);
return a.nombre;

---

25. Dado el siguiente programa, indicar qué **COMENTARIOS** a sustituir son correctos.

```
typedef struct
{ char nombre[20]; int nota; } TIPOREG;

main()
{
 TIPOREG reg;
 TIPOREG talum[5] = {{"JUAN",5},{"ANA",7},{"LUIS",9},{"EVA",3},{"PACO",6}};
 FILE *fp = fopen("DATOS.DAT","w");
 fwrite(talum, sizeof(TIPOREG), 5, fp);
 fclose(fp);
 fp = fopen("DATOS.DAT","r");

 /*  COMENTARIO1 A SUSTITUIR  */
```

```
  printf("%d, %s \n", reg.nota, reg.nombre);

  /*  COMENTARIO2 A SUSTITUIR  */
  printf("%d, %s \n", reg.nota, reg.nombre);

  /*  COMENTARIO3 A SUSTITUIR  */
  printf("%d, %s \n", reg.nota, reg.nombre);

  /*  COMENTARIO4 A SUSTITUIR  */
  printf("%d, %s \n", reg.nota, reg.nombre);
  fclose(fp);
}
```

01) COMENTARIO1: Me posiciono en el primer registro y lo leo
fseek(fp,0L,SEEK_SET); fread(&reg,sizeof(reg),1,fp);

02) COMENTARIO2: Me sitúo en el cuarto registro y lo leo
fseek(fp,3L*sizeof(reg),SEEK_SET); fread(&reg,sizeof(reg),1,fp);

04) COMENTARIO3: Me muevo dos registros hacia atrás y leo el registro
fseek(fp,-2L*sizeof(reg),SEEK_CUR); fread(&reg,sizeof(reg),1,fp);

08) COMENTARIO4: Me posiciono en el último registro y lo leo
fseek(fp,0L,SEEK_END); fread(&reg,sizeof(reg),1,fp);

---

26. ¿Qué imprime el siguiente programa, suponiendo que el fichero de texto **fich.txt** existe y que contiene un texto de **100** caracteres?

```
main()
{
  int n=0;
  FILE *pf = fopen("fich.txt", "r+");
  while (!feof(pf) && !ferror(pf))
  {
        fgetc(pf);   n++;
  }
  printf("%d", n);
  fclose(pf);
}
```

01) 100
02) 99
04) Nada, da error de compilación
08) 101

---

27. ¿Qué imprime el siguiente programa, suponiendo que el fichero de texto **fich.txt** existe y que contiene el texto "**EXAMEN DE PROGRAMACION**" sin ningún carácter **\n** en su interior?

```
main()
{
  char cad[10];
  FILE *pf = fopen("fich.txt", "r+");
  fgets(cad, 5, pf);
  while (!feof(pf) && !ferror(pf))
  {
        printf("%s", cad);
        fgets(cad, 5, pf);
  }
```

```
  fclose(pf);
}
```

01) EXAME
02) EXAMEN DE PROGRAMACI
04) EXAMEN DE PROGRAMACION
08) EXAMEN DE

---

28. En un fichero de tipo base de datos abierto mediante el puntero **pf** y formado por **100** registros de tipo **struct reg**, indicar cuáles son las instrucciones necesarias para leer el último registro del fichero en una variable de memoria de nombre **dato** de tipo **struct reg**.

01) fseek(pf, 0, SEEK_END); fread(&dato, sizeof(struct reg), 1, pf);
02) fseek(pf, 0, SEEK_END); fread(&dato, sizeof(dato), 1, pf);
04) fseek(pf, 99*sizeof(struct reg), SEEK_SET); fread(&dato, sizeof(struct reg), 1, pf);
08) fseek(pf, 100*sizeof(dato), SEEK_SET); fread(&dato, sizeof(dato), 1, pf);

---

29. Tenemos un fichero de texto llamado **notas.txt** conteniendo la información de todos los alumnos con el siguiente formato:

Nombre: John Apellido: Doe Asignatura: Programación Nota: 6.66

Si queremos leer un registro, suponiendo que el fichero se ha abierto para lectura correctamente y dadas las siguientes variables, indicar qué instrucción es la correcta:

char nombre[50], char apellido[50], char asignatura[50];
float nota=0.0;

```
01) fscanf(pf, “Nombre: %s Apellido: %s Asignatura: %s Nota: %f”,
               nombre, apellido, asignatura, nota);
02) fscanf(pf, “Nombre: %s Apellido: %s Asignatura: %s Nota: %f”,
               &nombre, &apellido, &asignatura, &nota);
04) fscanf(pf, “Nombre: %s Apellido: %s Asignatura: %s Nota: %f”,
               nombre, apellido, asignatura, &nota);
08) fscanf(pf, “Nombre: %s Apellido: %s Asignatura:  %s Nota: %f”,
               &nombre, &apellido, &asignatura, nota);
```

---

30. Suponiendo que se ha declarado **FILE *pf**, indique qué instrucciones son correctas para abrir el fichero **datos.dat** para añadirle nuevos valores al final.

```
01)  pf=fopen("datos.dat","a");
     if (pf!=NULL)
       {   puts("Error");   }
     else
       {   /* Escribimos */   }

02)  char nom[ ]="datos.dat";
     pf=fopen(nom, "w");
     if (pf==NULL)
       {   puts("Error");  }
     else
       {   /* Escribimos */   }

04)  pf=fopen("datos.dat","a");
     if (pf==NULL)
       {   puts("Error");  }
     else
```

```
        {   /* Escribimos */   }

08)  char nom[ ]="datos.dat";
     pf=fopen(nom, "w");
     if (pf!=NULL)
        {   puts("error");  }
     else
        {   /* Escribimos */   }
```

---

31. Un fichero que contiene estructuras de tipo **struct dato_t** acaba de ser abierto para lectura y es referenciado mediante un puntero a FILE **pf**. Si **x** es una variable de ese tipo de estructuras, indique qué instrucciones son correctas para leer el **penúltimo** registro del fichero:

```
01)   fseek(pf,-2, SEEK_END);
      fread(&x, sizeof(struct dato_t), 1, pf);

02)   fseek(pf,-2*sizeof(struct dato_t), SEEK_CUR);
      fread(&x, sizeof(struct dato_t), 1, pf);

04)   fseek(pf,-2*sizeof(struct dato_t), SEEK_END);
      fread(&x, sizeof(struct dato_t), 2, pf);

08)   fseek(pf,0,SEEK_END);
      fseek(pf,-2*sizeof(struct dato_t), SEEK_CUR);
      fread(&x, sizeof(struct dato_t), 1, pf);
```

LISTAS ENLAZADAS

1. Una lista enlazada está formada por estructuras con el siguiente formato:

```
struct pp
{ int num;
  struct pp *sig;
};
```

Se suponen además las siguientes declaraciones:

```
typedef struct pp tipo;
tipo *p,*q;
```

Indique qué instrucciones son necesarias para borrar el elemento siguiente al elemento apuntado por el puntero **p,** suponiendo que sí existe tal elemento siguiente.

```
01) q=p->sig; p->sig=q->sig; free(q);
02) p->sig=(tipo *)malloc(sizeof(tipo)); p->sig->sig=p->sig; free(p);
04) q=p->sig; p->sig=q; free(p);
08) q=p->sig; *p=*q; free(q);
```

---

2. Una lista enlazada está formada por estructuras con el siguiente formato:

```
struct nodo
{ int num;
  struct nodo *sig; };
```

Se suponen además las siguientes declaraciones:

```
typedef struct nodo tipo;
tipo *p,*q;
```

¿Qué instrucciones se deben poner para insertar un elemento de valor **50** después del elemento apuntado por el puntero **p** ?

```
01) q=p->sig;
    p->sig=(tipo *)malloc(sizeof(tipo));
    p->sig->num=50; p->sig->sig=q;
02) p->sig=(tipo *)malloc(sizeof(tipo));
    p->sig->num=50; p->sig->sig=p->sig;
04) q=p->sig;
    p->sig=(tipo *)malloc(sizeof(tipo));
    p=p->sig; p->num=50; p->sig=q;
08) q=(tipo *)malloc(sizeof(tipo));
    *q=*p; q->num=50; p->sig=q;
```

---

3. La función **fun** busca un elemento en una lista lineal enlazada, en la que cada elemento tiene el siguiente tipo de datos:

```
typedef struct nodo
{ int dato;
  struct nodo *sig;
} lista;
```

El primer parámetro **p** es un puntero al primer elemento de la lista y el segundo parámetro **num** es el número que se busca. La función **fun** debe devolver la dirección del elemento de la lista, si encuentra el numero a buscar, y devuelve NULL si no se encuentra el número en la lista. Indicar qué sentencias hay que sustituir por el comentario.

```
lista *fun(lista *p, int num)
{ int sw=0;
  /* COMENTARIO A SUSTITUIR  */
}
```

```
01) while((p != NULL) && (sw == 0))
    if( p->dato==num) sw=1; else p++;
    if (sw) return(p); else return NULL;
02) while((p != NULL) && (sw == 0))
    if( p->dato==num) sw=1; else p=p->sig;
    if (sw) return(p); else return NULL;
04) while((p->sig != NULL) && (sw == 0))
    if( p->dato==num) sw=1; else p=p->sig;
    if (sw) return(p); else return NULL;
08) while((p != NULL) && (sw == 0))
    if( p->dato==num) sw=1; else p=p->sig;
    if (sw) return(--p); else return NULL;
```

---

4. Una lista enlazada está formada por estructuras con el siguiente formato:

```
struct pp
{ int num;
  struct pp *sig;
};
```

Se suponen además las siguientes declaraciones:

```
typedef struct pp tipo;
tipo *p,*q;
```

¿Qué instrucciones son necesarias para borrar el elemento apuntado por el puntero **p,** suponiendo que no es el último de la lista?.

```
01) q=p->sig;  p->sig=q->sig; free(p);
02) p->sig=(tipo *)malloc(sizeof(tipo)); p->sig->sig=p->sig; free(p);
04) q=p->sig; p->sig=q; free(p);
08) q=p->sig; *p=*q; free(q);
```

---

5. Una lista enlazada está compuesta por estructuras del siguiente tipo:

```
struct pp
{  int num;
   struct pp *sig;
};
```

Indique la instrucción que falta para que la función **recorrer()** recorra e imprima la lista enlazada cuya dirección de comienzo se le da como argumento de entrada.

```
void recorrer(struct pp *pun)
{
  while(pun!=NULL)
  {
    printf("%d  ", pun->num);
    /*  Instrucción que falta    */
  }
}
```

01) pun++;
02) pun=pun+1;
04) pun=pun->sig;
08) pun->sig=pun;

---

6. Se ha realizado un listín telefónico por medio de una lista enlazada formada por estructuras con el siguiente formato:

```
typedef struct pp
{  char nom[20]; // nombre
   unsigned int tel; // numero de telefono
   struct pp *sig;
} ficha;
```

Indique cómo podría ser la función **buscar()**, a la que se le pasa como argumentos la dirección de comienzo **p** de la lista enlazada y un nombre **nom**, y devolverá como resultado el teléfono correspondiente a dicho nombre. Si el nombre a buscar no se encuentra en la lista, devolverá **0**.

01)
```
unsigned buscar(ficha *p,char *nom)
{  unsigned tel=0;
   while (p!=NULL)
   {  if (strcmp(nom,p->nom)==0) return p->tel;
      p=p->sig;  }
   return tel;
}
```

02)
```
unsigned *buscar(ficha *p,char *nom)
{  unsigned tel=0;
   while (p!=NULL)
   {  if (strcmp(nom,p->nom)==0) return p->tel;
      p=p->sig;   }
   return *tel;
}
```

04)
```
unsigned buscar(ficha *p,char *nom)
{  unsigned tel=0;
   while (p->sig!=NULL)
   {  if (strcmp(nom,p->nom)==0) return p->tel;
      p=p->sig;   }
   if (strcmp(nom,p->nom)==0) return p->tel;
   return tel;
}
```

08)
```
unsigned buscar(ficha *p,char *nom)
{  unsigned tel=0;
   while (p!=NULL)
```

```
    {  if (strcmp(&nom[0],&p->nom[0])==0) return p->tel;
       p=p->sig;   }
    return tel;
  }
```

---

7. En una lista lineal simplemente enlazada formada por elementos del tipo **struct elem** y apuntada por un puntero global **lista** que apunta al primer elemento de la lista, indicar cuál es la función **insertar** adecuada para insertar un nuevo elemento al final de la lista, incluso si la lista inicial está vacía.

```
typedef struct  elem
{
  int dato1;
  char dato2;
  struct elem *siguiente
} elemento;

elemento *lista;
```

```
01)  void insertar(int d1, char d2)
     {
       elemento *p = lista, *q = (elemento *) malloc(sizeof(elemento));
       if (q!=NULL)
       {
         q->dato1=d1;  q->dato2=d2; q->siguiente=NULL;
         while (p->siguiente!=NULL)
                p = p->siguiente;
         p->siguiente = q;
       }
     }
02)  void insertar(int d1, char d2)
     {
       elemento *p = lista, *q = (elemento *)malloc(sizeof(elemento));
       if (q==NULL)  return;
       q->dato1=d1; q->dato2=d2; q->siguiente=NULL;
       if (p==NULL)  { lista=q; return; }
       while (p->siguiente!=NULL)
              p = p->siguiente;
       p->siguiente = q;
     }

04)  void insertar(int d1, char d2)
     {
      elemento *p = lista, *q = (elemento *)malloc(sizeof(elemento));
       if (q==NULL)  return;
       q->dato1=d1;  q->dato2=d2; q->siguiente=NULL;
       if (p==NULL)  lista=q;
       else
       {
         while (p!=NULL)
                p = p->siguiente;
          p->siguiente = q;
       }
     }
08)  void insertar(int d1, char d2)
     {
       elemento *p = lista, *q;
       if ((q = (elemento *)malloc(sizeof(elemento)))==NULL)  return;
       q->dato1=d1;  q->dato2=d2; q->siguiente=NULL;
       if (p==NULL)
```

```
    {
        lista=q; return;
    }
    while (p->siguiente!=NULL)
            p = p->siguiente;
    p->siguiente = lista;
}
```

---

8. Una lista enlazada está formada por estructuras con el siguiente formato:

```
struct nodo
{
  int num;
  struct nodo *sig;
};
```

Se suponen además las siguientes declaraciones:

```
typedef struct nodo nodo_t;
nodo_t *p,*q;
```

Indique qué instrucciones son necesarias para añadir un elemento con el valor **8** después del elemento apuntado por el puntero **p**.

```
01) q=(nodo_t *) malloc(sizeof(nodo_t));
    p->sig = q;
    q->sig = p->sig;
    q->num = 8;
02) q=(nodo_t *) malloc(sizeof(nodo_t));
    q->sig = p->sig;
    q->num = 8;
    p->sig = q;
04) p->sig=(nodo_t *)malloc(sizeof(nodo_t));
    q=p->sig;
    q->sig=p->sig;
    q->num = 8;
08) q=(nodo_t *) malloc(sizeof(nodo_t));
    p->sig=q;
    q->sig=p;
    q->num = 8;
```

PILAS Y COLAS

1. En una pila se han ido almacenando las siguientes informaciones por orden de llegada: **MA,JA,CO,GR,SE,BU**, siendo **MA** la información que lleva más tiempo en la pila. Si se hacen dos operaciones de sacar (**pop**) y una operación de meter (**push**) en la que se mete el dato **AL**,

01) Con la siguiente operación pop obtendremos AL.
02) Con la siguiente operación pop obtendremos AL y con la siguiente pop obtendremos GR.
04) Con la siguiente operación pop obtendremos CO.
08) Con la siguiente operación pop obtendremos BU y con la siguiente pop obtendremos SE.

---

2. La pila de la figura es controlada mediante el puntero de ámbito global **pi**.

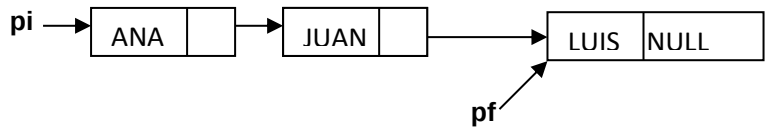
El puntero **pi** apunta al último elemento que se introdujo en la pila. Se quiere realizar una función llamada **fun**, que permita extraer un elemento de la pila y lo imprima. Indicar qué sentencias hay que sustituir por el comentario, para construir dicha función **fun**. Cada elemento de la pila es del tipo:

```
struct nodo { char nombre[40];
              struct nodo *sig; }
void fun(void)
{ struct nodo *pt;
  char nom[40];
  /* COMENTARIO A SUSTITUIR */
  free(pt);
}
```

01) strcpy(nom,pi->nombre); printf("%s\n ",nom);
pt=pi; pi=pi->sig;
02) printf("%s\n ",pi->nombre);
pi=pi->sig; pt=pi;
04) nom = pi->nombre; printf("%s\n ", nom);
pt=pi; pi=pi->sig;
08) strcpy(nom,pi->nombre);
pt=pi; pi=pi->sig;
printf("%s\n ",nom);

---

3. La cola de la figura es controlada mediante dos punteros de ámbito global: **pi** y **pf**.



El puntero **pi** apunta al primer elemento que se introdujo en la cola, y el puntero **pf** apunta al último que se ha introducido. Se quiere realizar una función llamada **insertar**, que permita añadir un elemento en la cola, siendo el parámetro **cad** el que recibe el nombre a añadir. Indicar qué sentencias hay que sustituir por el comentario para construir dicha función **insertar**. Cada elemento de la cola es del tipo:

```
struct nodo { char nombre[40];
              struct nodo *sig; };

void insertar (char *cad)
{
  struct nodo *pt = (struct nodo *)malloc(sizeof(struct nodo));
  /* COMENTARIO A SUSTITUIR  */
}
```

01) strcpy(pt->nombre ,cad); pt->sig= NULL;
if (pf==NULL) pi=pt; else pf->sig=pt;
pf=pt;
02) pt->nombre=cad) ;
if (pf==NULL) pi=pt; else pf->sig=pt;
pt->sig= NULL; pf=pt;
04) strcpy(pt->nombre ,cad) ;
if (pi==NULL) pf=pt; else pi->sig=pt;
pt->sig= NULL; pi=pt;
08) strcpy(pt->nombre ,cad) ;
if (pf==NULL) pi=pt; else pf->sig=pt;
pt->sig= NULL; pf=pt;

---

4. En una pila se han ido almacenando los siguientes números por orden de llegada:
27, 33, 12, 20, 8, 10, siendo **27** el dato que lleva más tiempo en la pila. Si se hacen dos operaciones de sacar (**pop**) y una operación de meter (**push**) en la que se mete el dato **15**,

01) Con la siguiente operación pop obtendremos 15.
02) Con la siguiente operación pop obtendremos 15 y con la siguiente pop obtendremos 20.
04) Con la siguiente operación pop obtendremos 12.
08) Con la siguiente operación pop obtendremos 12 y con la siguiente pop obtendremos 20.

---

5. En una cola se han ido almacenando los siguientes números por orden de llegada:
27, 33, 12, 20, 8, 10, siendo **27** el dato que lleva más tiempo en la cola. Si se hacen dos operaciones de sacar (**pop**) y una operación de meter (**push**) en la que se mete el dato **15**,

01) Con la siguiente operación pop obtendremos 15.
02) Con la siguiente operación pop obtendremos 15 y con la siguiente pop obtendremos 20.
04) Con la siguiente operación pop obtendremos 12.
08) Con la siguiente operación pop obtendremos 12 y con la siguiente pop obtendremos 20.

---

6. Indicar qué función sirve para extraer un dato de una pila constituida por una lista enlazada simple con estructuras del tipo **elemento** y gobernada por el puntero global **pila**.

```
typedef struct elem {  int dato; struct elem *psig;  } elemento;
elemento *pila=NULL;
```

```
01)  int pop(void)
     { elemento *q; int x;
       if (pila==NULL) { printf("Pila vacía\n"); return EOF; }
       q = pila; x = q->dato; q = q->psig; free(q); return x;
     }
02)  int pop(void)
     { elemento *q; int x;
       if (pila==NULL) { printf("Pila vacía\n"); return EOF; }
       x = pila->dato; q = pila; pila = pila->psig; free(q); return x;
     }
04)  int pop(void)
     { elemento *q; int x;
       if (pila==NULL) { printf("Pila vacía"); x=EOF; }
       else { x = pila->dato; q = pila; pila = q->psig; free(q); }
       return x;
     }
08)  int pop(void)
     { elemento *q; int x;
       if (pila==NULL) { printf("Pila vacía\n"); return EOF; }
       q = pila; x = q->dato; pila = q->psig; free(q); return x;
     }
```

---

7. En una lista lineal doblemente enlazada formada por elementos del tipo **struct elem** y apuntada por un puntero global **lista** que apunta al primer elemento de la lista, indicar cuáles son las instrucciones necesarias para eliminar el ultimo elemento de la lista.

```
struct elem
{
  int dato1;
  char dato2;
  struct elem *anterior;
  struct elem *siguiente
};
struct elem *lista, *p, *q;
```

```
01)  p=lista;
     while (p!=NULL)
```

```
        p = p->siguiente;
    q = p->anterior;
    if (q==NULL)  lista = NULL;
    else  q->siguiente = NULL;
    free(p);
02) p=lista;
    while (p->siguiente!=NULL)
        p = p->siguiente;
    q = p->anterior;  q->siguiente = NULL;
    free(p);
04) p=lista;
    while (p->siguiente!=NULL)
         p = p->siguiente;
    q = p->anterior;
    if (q==NULL)  lista = NULL;
    else   q->siguiente = NULL;
    free(q);
08) p=lista;
    while (p->siguiente!=NULL)
          p = p->siguiente;
    q = p->anterior;
    if (q==NULL)  lista = NULL;
    else   q->siguiente = NULL;
    free(p);
```

---

8. En una cola se han ido almacenando los siguientes números por orden de llegada:
27, 33, 12, 20, 8, 10, siendo **27** el dato que lleva más tiempo en la cola.

Si se hacen tres operaciones de sacar (**pop**) y dos operaciones de meter (**push**) en la que se introducen los dato **15** y **22** (en ese orden),

01) Con la siguiente operación pop obtendremos 15.
02) Con la siguiente operación pop obtendremos 10.
04) Con la siguiente operación pop obtendremos 22.
08) Con la siguiente operación pop obtendremos 20.

ÁRBOLES BINARIOS

1. Se crea un árbol binario de búsqueda con los siguientes datos recibidos por el teclado:
ene, feb, mar, abr, may, jun, jul, ago, sep, oct, nov, dic.
Indicar qué afirmaciones son correctas:

01) De los dos recorridos estándar del árbol en Preorden, uno es el siguiente:
    ene, feb, mar, may, sep, oct, nov, jun, jul, abr, ago, dic
02) El árbol tiene altura 7
04) El nodo "dic" tiene nivel 3
08) De los dos recorridos estándar del árbol en Postorden, uno es el siguiente:
    jul, jun, nov, oct, sep, may, mar, feb, dic, ago, abr, ene

---

2. Un árbol binario de búsqueda está formado por estructuras cuyas claves son números enteros y ha sido creado con la siguiente información comenzando por la raíz: **10 7 6 8 15 9 3 21 14 20 2**
Indique qué afirmaciones son correctas:

01) Si se recorre en Inorden se obtiene: 2 3 6 7 8 9 10 14 15 20 21

02) Si se recorre en Preorden se obtiene: 10 7 6 3 2 8 9 15 14 21 20
04) Si se recorre en Postorden se obtiene: 2 3 6 9 8 7 14 20 21 15 10
08) Si se recorre en Inorden se obtiene: 21 20 15 14 10 9 8 7 6 3 2

---

3. Si creamos un árbol binario de búsqueda con las cadenas siguientes: **uno dos tres cuatro cinco seis siete ocho nueve**
en el que cada nodo está compuesto por una estructura del tipo:

```
struct nodo
{ char d[10];
  struct nodo *iz, *de; };
```

¿Qué imprime la función **fun** si recibe como argumento **r** la dirección del nodo que contiene la cadena **"uno"**?

```
void fun(struct nodo *r)
{ if(r!=NULL)
  { fun(r->de); printf("%s  ",r->d)); fun(r->iz); }
}
```

01) uno dos tres cuatro cinco seis siete ocho nueve
02) nueve ocho siete seis cinco cuatro tres dos uno
04) cinco cuatro dos nueve ocho seis siete tres uno
08) cuatro dos cinco uno nueve ocho tres seis siete

---

4. Si creamos un árbol binario de búsqueda con las siguientes cadenas una a una:
dabale arroz a la zorra el abad en un arbol
siendo **dabale** el primer dato recibido y en el que cada nodo es una estructura del tipo:

```
struct nodo
{ char cad[40];
  struct nodo *iz,*de; };
```

indique qué afirmaciones son correctas:

01) "abad" no es el nodo raíz
02) Un recorrido en Postorden es: arbol abad a arroz en el un zorra la dabale
04) "la" es un nodo terminal
08) Un recorrido en Inorden es: a abad arbol arroz dabale el en la un zorra

---

5. Si creamos un árbol binario de búsqueda que almacena claves de tipo cadenas de caracteres, y con las siguientes cadenas introducidas en este orden:
rojo verde azul amarillo naranja blanco negro rosa marron
siendo **rojo** el nodo raíz y en el que cada nodo es una estructura del tipo:

```
struct nodo
{ char cad[40];
  struct nodo *iz,*de; };
```

indique qué afirmaciones son correctas:

01) la altura o profundidad del árbol es 4
02) un recorrido en Postorden es:
rosa verde negro marron blanco naranja amarillo azul rojo
04) "blanco" es un nodo terminal
08) un recorrido en Inorden es:
verde rosa rojo negro naranja marron blanco azul amarillo

---

6. Se ha creado un árbol binario ordenado formado por estructuras del siguiente tipo:

```
typedef struct pp
{ int clave;
  struct pp *izq;   // Puntero al subarbol izquierdo
```

```
  struct pp *der;  // Puntero al subarbol derecho
} tipo;
```

El árbol se ha creado con los siguientes valores tomados por teclado, empezando por la raiz:

6,12,9,7,8,2,3,5,4

¿Qué imprimirá la función **fun()** si es llamada con un argumento de entrada igual al puntero raíz del árbol?

```
void fun(tipo *r)
{
 if (r!=NULL)
 {  fun(r->der);  printf("%d  ", r->clave);  fun(r->izq);  }
}
```

01) 2 3 4 5 6 7 8 9 12
02) 8 7 9 12 4 5 3 2 6
04) 12 9 8 7 6 5 4 3 2
08) 4 5 3 2 8 7 9 12 6

---

7. Se ha creado un árbol binario ordenado formado por estructuras del siguiente tipo:

```
typedef struct pp
{ int clave;
  struct pp *iz;
  struct pp *der;
} tipo;
```

El árbol se ha creado con los siguientes valores tomados por teclado, empezando por la raiz:

8,12,9,7,6,2,3,5,4

¿Qué imprimirá la función **fun()** si es llamada con un argumento de entrada igual al puntero raíz del árbol?

```
void fun(tipo *r)
{
  if (r!=NULL)
  {  fun(r->iz);  printf("%d  ", r->clave);   fun(r->der);  }
}
```

01) 2 3 4 5 6 7 8 9 12
02) 8 7 12 6 2 3 4 5 9
04)12 9 8 7 6 5 4 3 2
08)4 5 3 2 6 7 8 9 12

---

8. Se crea un árbol binario ordenado con las siguientes claves numéricas recibidas por teclado:
50, 28, 15, 75, 18, 60, 81, 42, 65, 70. Indicar qué imprime la función **imprimirArbol** si recibe como argumento un puntero **p** que apunta al nodo "**50**".

```
struct nodo
{
  int dato;
  struct nodo *dcho, *izqdo;
}
void imprimirArbol(struct nodo *p)
{
  if (p!=NULL)
  {
     imprimirArbol(p->dcho);  imprimirArbol(p->izqdo);  printf("%d  ", p->dato);
   }
}
```

01) 70 65 60 81 75 42 18 15 28 50
02) 42 18 15 28 81 70 65 60 75 50
04) 81 70 65 60 75 42 18 15 28 50
08) 15 18 28 42 50 60 65 70 75 81

---

9. Si creamos un árbol binario ordenado con los siguientes datos: **30 50 70 20 10 25 40 60 12 15 55** en el que cada nodo está compuesto por una estructura del tipo:

```
struct nodo
{
  int dato;
  struct nodo *dcho, *izqdo;
};
```

¿Qué imprime la función **imprimirArbol** si recibe como argumento **p** la dirección del nodo que contiene el valor **20**?

```
void imprimirArbol(struct nodo *p)
{
  if (p!=NULL)
  {
     imprimirArbol(p->dcho);
     printf("%d  ", p->dato);
     imprimirArbol(p->izqdo);
   }
}
```

01) 70 60 55 50 40 30 25 20 15 12 10
02) 10 12 15 20 25 30 40 50 55 60 70
04) 25 20 15 12 10
08) 10 12 15 20 25

SOLUCIONES

PUNTEROS A ARRAYS Y A CADENAS DE CARACTERES

1. 02
2. 00
3. 11
4. 11
5. 08
6. 06
7. 15
8. 15
9. 14
10. 08
11. 01
12. 04
13. 08
14. 01

PUNTEROS A ESTRUCTURAS

1. 05
2. 04
3. 12
4. 13
5. 07
6. 02

PUNTEROS GENÉRICOS

1. 02
2. 08
3. 08

ARRAYS DE PUNTEROS

1. 07
2. 15
3. 04
4. 06
5. 01
6. 02

PUNTEROS A PUNTEROS

1. 08
2. 08
3. 15
4. 15
5. 09
6. 09
7. 00

PASO DE ARGUMENTOS POR REFERENCIA

1. 04
2. 02
3. 04

4. 02
5. 04
6. 01
7. 04
8. 08

ASIGNACIÓN DINÁMICA DE MEMORIA

1. 02
2. 11
3. 14
4. 08
5. 01

PASO DE ARRAYS A FUNCIONES

1. 11
2. 04
3. 11
4. 15

PASO DE ESTRUCTURAS A FUNCIONES

1. 10
2. 08
3. 02
4. 08
5. 02
6. 02
7. 08

RECURSIVIDAD

1. 07
2. 01
3. 01
4. 06
5. 05
6. 02
7. 08
8. 08
9. 04
10. 02
11. 04
12. 04
13. 04
14. 04

ARGUMENTOS EN LÍNEA DE ÓRDENES

1. 04
2. 05
3. 08
4. 09
5. 08
6. 06
7. 08
8. 04
9. 14
10. 14
11. 02
12. 02

13. 02

FICHEROS

1. 04
2. 11
3. 12
4. 02
5. 02
6. 01
7. 08
8. 06
9. 15
10. 14
11. 02
12. 08
13. 07
14. 07
15. 04
16. 03
17. 15
18. 11
19. 11
20. 12
21. 11
22. 02
23. 08
24. 05
25. 07
26. 08
27. 02
28. 04
29. 04
30. 04
31. 08

LISTAS ENLAZADAS

1. 01
2. 13
3. 02
4. 08
5. 04
6. 13
7. 02
8. 02

PILAS Y COLAS

1. 03
2. 09
3. 09
4. 03
5. 12
6. 14
7. 08
8. 08

ÁRBOLES BINARIOS

1. 05
2. 15
3. 00
4. 11
5. 11
6. 04
7. 01
8. 04
9. 04