	FUNDAMENTOS DE INFORMÁTICA (141211006) INGENIEROS INDUSTRIALES CONVOCATORIA SEPTIEMBRE 2008 (1-septiembre-2008)
	NOMBRE: DNI:

PRIMERA PREGUNTA	1 PUNTO

Suponga el siguiente código:

```
#include <stdio.h>
#define LIM 100
int pitagoras(short, short, short);
void main(void)
{
      short a, b, c;
      for(a = 1 ; a < LIM ; a++)
            for(b = a ; b < LIM ; b++)
                  for(c = b; c < LIM ; c++)
                        if(pitagoras(a, b, c))
                              printf("%hd - %hd - %hd\n", a, b, c);
}
```

Donde la función pitagoras devuelve un entero distinto de cero si la terna a, b y c verifica la conocida relación de Pitágoras, y el valor cero en caso contrario.

Defina la función pitagoras.

```
int pitagoras(short x, short y, short z)
{
      return x * x + y * y == z * z;
}
```

SEGUNDA PREGUNTA	2 PUNTOS

Se necesita una función que indique si los valores de un array cuya longitud se recibe como segundo parámetro son todos ellos distintos entre sí o existe algún valor repetido. La función ha de devolver un valor distinto de cero si todos los elementos del array son distintos.

Se le persenta un programa que declara y usa la función ***distintos*** que es la que usted debe definir.

```
#include <stdio.h>
#define TAM 5
int distintos(long*, short);
void main(void)
{
	long a[TAM];
	short i;
	for(i = 0 ; i < TAM ; i++)
	{
		printf("Valor de a[%hd] = ", i);
		scanf("%ld", a + i);
	}

	if(distintos(a, TAM)) printf("Hay elementos iguales");
	else printf("No hay elementos iguales");
}
```

```
int distintos(long* m, short d)
{
    short i, j, iguales;
    for(i = 0 , iguales = 0 ; i < d && !iguales ; i++)
    {
        for(j = i + 1 ; j < d ; j++)
            if(m[i] == m[j])
            {
                iguales = 1;
                break;
            }
    }
    return iguales;
}
```

TERCERA PREGUNTA	2 PUNTOS

Lea el siguiente código en C:

```
#include <stdio.h>
#include <ctype.h>
#define LONG 100
void main(void)
{
	char c[LONG];
	short i = 0;
	printf("Introduzca la cadena de texto... ");
	gets(c);
	while(i < 100 && c[i] != '\0')
	{
		if(isdigit(c[i])) c[i] = '\0';
		else i++;
	}
	printf("%s", c);
}
```

Indique qué salida ofrecerá por pantalla si el usuario introduce, en la ejecución de la función *gets()*, la cadena de texto “Examen Septiembre 2008”.

Examen Septiembre

CUARTA PREGUNTA	1 PUNTO

Lea el siguiente código y muestre su salida por pantalla:

```
#include <stdio.h>
void intercambiar(long, long);
void main(void)
{
	long a = 5 , b = 2;
	intercambiar(a, b);
	printf("a = %ld - b = %ld", a, b);
}

void intercambiar(long a, long b)
{	a ^= b ; b ^= a ; a ^= b;	}
```

a = 5 - b = 2

QUINTA PREGUNTA	3 PUNTOS

Se llama PRIMO PERFECTO a aquel entero primo n que verifica que $(n-1)/2$ también es primo. Por ejemplo, $n = 11$ es primo y $(n-1)/2 = 5$ también es primo: luego $n = 11$ es un primo perfecto.

Haga un programa que muestre los primos perfectos menores que 1000.

Sugerencia: se simplifica mucho el código si define una función (llamad, por ejemplo, ***int esprimo(long)***, que devuelve un entero distinto de cero cuando el entero que se recibe como parámetro un valor primo).

```
#include <stdio.h>
#include <math.h>
int esprimo(long);
void main(void)
{
       long N;
       for(N = 5 ; N < 1000 ; N += 2)
         if(esprimo(N) && esprimo((N - 1) / 2))
             printf("%ld\n", N);
}
int esprimo(long x)
{
       short d;
       if(x % 2 == 0) return 0;
       for(d = 3 ; d <= sqrt(x) ; d += 2)
         if(x % d == 0) return 0;
       return 1;
}
```

SEXTA PREGUNTA	1 PUNTO

Indique, en cada sentencia, si es correcta o existe algún error sintáctico. Si la expresión está correcta, indíquelo con la palabra “CORRECTO” en el recuadro de su derecha; de los contrario, indique dónde está el error (Tres respuestas erróneas o en blanco supone un cero en la pregunta).

`short x[] = {1, 2, 3, 4, 5};`	CORRECTO.
`short x[];`	Falta la dimensión del array.
`short x[5] = {1, 2, 3, 4};`	Falta un quinto valor en la asignación.
`short x[] = {1, 2, 3, 4}`	Falta el punto y coma final.
`short x[1] = {1};`	CORRECTO.
`short x[5]; x++;`	No se puede modificar el valor de x.
`short x[3][2] = {{1,1,1},{2,2}}`	CORRECTO.

`short x[3];`
`for(i = 0 ; i <= 3; i++)`
`  x[i] = i;`
	Violación de memoria: El índice i accede a una posición del array no permitida.

`short x[3];`
`for(i = 0 ; i < 3 ; i++)`
`  *(x + i) = i`
	CORRECTO.

`short x[5], *p = &x;`	p es puntero a **short**. &x es puntero a puntero a **short**. No son tipos compatibles. No vale la asignación.