Nombre:
Apellidos:
Grupo: **Nº. Matrícula**

Ejercicio 1 (2,5 puntos – 30 minutos)

Al ingresar capital en un banco, pongamos 1000€, éste nos ofrece un porcentaje anual de interés, por ejemplo el 4%. Al cabo de un año nuestro capital se ha convertido en:

1000+(4/100)*1000 =1000 * (1+4/100) =1000 * 1.04 = 1040.

Al finalizar el segundo año, el 4% se aplica sobre los 1040€, y obtenemos:

1040+1040*(4/100) = 1040*(1+4/100) = 1040 * 1.04

Si colocamos estas cantidades seguidas:

1000, 1000*1.04, 1000*1.04*1.04, …

se corresponden a los términos de una progresión geométrica de razón 1.04. Cada término se obtiene a partir del anterior multiplicado por 1.04. Se pide:

- En primer lugar, desarrollar una función 'leerdatos', que pida por teclado los valores de capital, interés y años. Esta función tendrá tres parámetros por referencia y devolverá 'void'.
- Escribir una función (llamada 'calcapital') que calcule el capital que tendremos al cabo de 'x' años, si ingresamos una cantidad inicial 'c' euros a un interés (en %) 'i' si no se toca el dinero durante los 'x' años. La función tendrá como parámetros 'x', 'c' e 'i', y devolverá el valor del capital al cabo de los 'x' años (con sólo dos decimales).
- Escribir el programa principal, el cual llamará a las funciones 'leerdatos' y 'calcapital', y escribirá el valor retornado por 'calcapital' en pantalla. Esta operación se realizará tantas veces como quiera el usuario.

Ejemplo de ejecución:

```
>Introduzca el capital: 1000
>Introduzca el interés (%): 4
>Introduzca el número de años: 5
>El capital al cabo de 5 años es: 1216,65€.
>Quiere realizar la operación otra vez (S/N)?: S
>Introduzca el capital: 2500
>Introduzca el interés (%): 2
>Introduzca el número de años: 10
>El capital al cabo de 5 años es: 3047,49€.
>Quiere realizar la operación otra vez (S/N)?: N
>Fin del programa
```

Nombre:
Apellidos:
Grupo: **Nº. Matrícula**

Ejercicio 2 (2,5 puntos – 30 minutos)

Se debe calcular la nómina mensual de un empleado que trabaja por horas, el pago de cada hora trabajada depende de su categoría:

categoría	pago x hora (€.)
A	30.00
B	20.50
C	18.50

Además si el empleado trabaja más de 150 horas mensuales tiene una bonificación del 5 % de nómina.

El programa que calcula la nómina está compuesto por dos funciones, una que solicita los datos del empleado por teclado y otra función que calcula la nómina mensual. El programa principal debe mostrar la nómina mensual calculada del empleado introducido.

Para los datos del empleado se debe usar la siguiente estructura:

```
typedef struct
{    char nom[40], cat;
     int horas;
     float nomina, pHora, bonf;
} empleado;
```

Los prototipos de las funciones que se tienen que utilizar son los siguientes:

```
void Pedirdato(empleado* emp);

float Nomina (empleado, float  *);
```

Ejemplo de ejecución:

```
Introduzca nombre empleado: Pepe Campo
Introduzca categoría del empleado: B
Introduzca horas trabajadas: 25

La nómina neta del empleado Pepe Campo es: 512.50 euros
```

Nombre:
Apellidos:
Grupo: **Nº. Matrícula**

Ejercicio 3 (2,5 puntos – 30 minutos)

La constante de Brun es el valor al que converge la suma de los inversos de los números primos gemelos. Su aproximación se calcula mediante la expresión siguiente:

$$B_2 = \left(\frac{1}{3}+\frac{1}{5}\right)+\left(\frac{1}{5}+\frac{1}{7}\right)+\left(\frac{1}{11}+\frac{1}{13}\right)+\left(\frac{1}{17}+\frac{1}{19}\right)+\left(\frac{1}{29}+\frac{1}{31}\right)+\cdots \sim 1{,}902160583104$$

Nota: Dos números primos se denominan gemelos cuando su diferencia es igual a 2. Por ejemplo, son primos gemelos el 3 y el 5, el 5 y el 7, el 11 y el 13, etc.

Se pide: Elaborar un programa que solicite al usuario un número entero "n" entre 1 y 1000 y calcule la aproximación a la constante de Brun utilizando los primos gemelos que existan en el subconjunto de los "n" primeros números naturales. El programa deberá tener al menos las siguientes funciones:

`int es_primo (int n);` // Recibe un entero "n" y determina si es primo o no.

`int get_primos(int v[],int n);` // Recibe un vector de enteros y un entero "n". Almacena en el vector los números primos que existen entre 1 y "n" y devuelve como resultado la cantidad de primos encontrados en ese rango.

`int constante_brun (int v[],int m, float *b);` // Recibe un vector de enteros con "m" números primos y el valor de m. Calcula la aproximación a la constante de Brun y devuelve la cantidad de primos gemelos encontrados en el vector.

Ejemplo de ejecución:

```
Introduzca un número entero entre 1 y 1000: -1
ATENCIÓN: Valor introducido fuera de rango (1-1000)
Introduzca un número entero entre 1 y 1000: 20
Hay 4 pares de números primos gemelos entre 1 y 20
La constante de Brun calculada vale: 1.155478
Presione una tecla para continuar . . .
```

Nombre:
Apellidos:
Grupo: **Nº. Matrícula**

Ejercicio 4 (2,5 puntos – 30 minutos)

Se necesita un programa para calcular la puntuación de cada una de las solicitudes para asignar las becas del curso 2014/2015.

La información de las solicitudes se encuentra en un fichero de texto cuyo nombre será introducido por el usuario desde el teclado. Dicho fichero contiene en cada línea la siguiente información: DNI con letra, la renta per cápita de la familia, el número de progenitores/tutores en situación de desempleo y el número de menores de 12 años de la unidad familiar. Por ejemplo:

```
52898324M 3009.32  0 1
51737212K 6500.50 1 0
…
```

Los baremos de puntuación aplicables son:

- Ingresos familiares:

Renta per cápita hasta	2.500,99 €	4 puntos
Renta per cápita de 2.501,00 €	a 4.000,99 €	3 puntos
Renta per cápita de 4.001,00	a 5.000, 99 €	2 puntos
Renta per cápita de 5.001,00 €	a 6.000, 00 €	1 punto
Renta per cápita superior a 6.000,01 €		0 puntos

- Por la situación de desempleo de uno o de ambos progenitores o tutores: 2 puntos por cada uno.
- Por cada menor de 12 años de la unidad familiar: 1 punto.

El programa debe generar un fichero de salida denominado “resultados.txt” que tendrá una primera línea con cinco columnas: DNI, P1, P2, P3, Total. Y a continuación habrá una línea por cada una del fichero de entrada, donde se escriba el DNI, las puntuaciones obtenidas por cada uno de los datos y el total de la puntuación debidamente calculada. Por ejemplo:

Ejemplo de archivo de salida “resultados.txt”:

```
DNI          P1   P2   P3   Total
52898324M    3    0    1    4
51737212K    0    2    0    2
…
```

Se debe comprobar la correcta apertura del fichero y en caso de que no pueda ser abierto se debe informar al usuario con un mensaje y cerrar el programa.

Nombre:
Apellidos:
Grupo: **Nº. Matrícula**

SÓLO A REALIZAR POR LOS ALUMNOS DE EVALUACIÓN FINAL

Ejercicio 5 (2,5 puntos – 30 minutos)

Se desea realizar un programa que permita comprobar si una palabra es un palíndromo, es decir, si se lee igual de izquierda a derecha que de derecha a izquierda. Por ejemplo, radar y reconocer son palíndromos.

Para comprobarlo, el programa debe hacer una copia invertida de la palabra (escribirla al revés) y luego comprobar si la palabra original y la copia invertida son iguales, en cuyo caso se trata de un palíndromo.

Se pide completar el programa con los prototipos y las definiciones de las cuatro funciones siguientes:

EsPalindromo	Comprueba si la palabra es un palíndromo.
CopiaInvertida	Copia la palabra original en otra, pero escribiéndola de derecha a izquierda.
ComparaPalabras	Comprueba que dos palabras son iguales (se supondrá que ambas tienen la misma longitud).
LongitudPalabra	Calcula el número de caracteres de una palabra sin contar el carácter nulo.

Nota: No pueden utilizarse funciones de la librería string.h

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

//Completar los prototipos de las funciones
int EsPalindromo

void CopiaInvertida

int ComparaPalabras

void LongitudPalabra

void main()
{
     char palabra[20];

     printf("Introduzca una palabra: ");
     gets(palabra);

     if(EsPalindromo(palabra))
            printf("Es palindromo\n");
     else
            printf("No es palindromo\n");

     system("pause");
}

//Completar las definiciones de las funciones
```

SOLUCIÓN EJERCICIO 1

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

float calcapital(float capital, float interes, int anos);
void leerdatos(float* capital, float* interes, int* anos);

int main()
{
   float capital;
   float interes;
   int anos;
   char letra='s';

   while(letra=='s')
   {
      leerdatos(&capital, &interes, &anos);

      printf("El capital al cabo de %d años es:%.2f€\n",anos,calcapital(capital,interes,anos));

      printf("Desea continuar(s/n)?");
      scanf("%c",&letra);
   }
        printf("Fin del programa");
}

void leerdatos(float* capital, float* interes, int* anos)
{
   printf("Introduzca el capital: ");
   scanf("%f",capital);
   printf("\nIntroduzca el interes: ");
   scanf("%f",interes);
   printf("\nIntroduzca los anos: ");
   scanf("%d",anos);
}

float calcapital(float capital, float interes, int anos)
{

   float resul=capital*pow((1+(interes/100)),anos);
   return resul;

}
```

SOLUCIÓN EJERCICIO 2

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

typedef struct
{    char nom[40], cat;
    int horas;
    double nomina, pHora, bonf;
}empleado;

//leer datos del empleado
void Pedirdato(empleado* emp);

//cálculo de la nómina del empleado

float Nomina (empleado, float *);


int main()
{
   float nom=0.0;
   empleado emp;


          Pedirdato(&emp);

     emp.nomina=Nomina(emp, &nom);

          printf("La nomina neta del empleado %s es %.2f euros \n", emp.nom, emp.nomina);

          fflush(stdin);

          system("pause");
}

// se introducen los datos del empleado

void Pedirdato(empleado* emp)
{
          printf("Introduce nombre empleado:\n ");
          gets(emp->nom);

          printf("Introduzca categoria: \n");
          scanf("%c",&emp->cat);

          printf("Introduzca cantidad de horas trabajadas: \n");
          scanf("%d",&emp->horas);

}


float Nomina (empleado dato, float *nom)
{

          *nom=0.0;
            switch(dato.cat)
            {
          case 'A': dato.pHora=30.00;   break;
          case 'B': dato.pHora=20.50;   break;
          case 'C': dato.pHora=18.50;   break;
                              default:
                                        {printf("Categoria no correcta \n");
                                        dato.pHora=0.00;}
                                        break;
```

```
    }
    dato.nomina=dato.horas*dato.pHora;
    if(dato.horas>150)
        dato.bonf=0.05*dato.nomina;
    else
        dato.bonf=0;
    dato.nomina=dato.nomina+dato.bonf;
          *nom=dato.nomina;
          return *nom;
}
```

SOLUCIÓN EJERCICIO 3

```
#include <stdio.h>
#include <stdlib.h>
#define MAXIMO 1000
int es_primo (int);                    // Determina si un número es primo
int get_primos(int [],int);            // Obtiene los primos entre 1 y n
int constante_brun (int [],int, float *); // Calcula Brun y pares encontrados

int main(void)
{
  int vector[MAXIMO]; // Contenedor para los números primos.
  int n,m;            // Rango de números y cantidad de primos encontrados
  int pares;          // Cantidad de pares de primos gemelos encontrados.
  float brun;         // Aproximación a la constante de Brun.
  int i;              // Variable de control de bucle.

  do{     // Bucle para la entrada de datos con comprobación de rango.
   printf ("Introduzca un número entero entre 1 y %d: ",MAXIMO);
   scanf ("%d",&n);
   if ((n<1) || (n>MAXIMO))
     printf ("ATENCIÓN: Valor introducido fuera de rango (1-%d)\n",MAXIMO);
  }while ((n<1) || (n>MAXIMO));

  m=get_primos(vector,n);                  // Obtener primos entre 1 y n
  pares = constante_brun (vector,m,&brun);  // Llamada a la función de Brun
                                  // Visualización de resultados
  printf ("Hay %d pares de números primos gemelos entre 1 y %d\n",pares,n);
  printf ("La constante de Brun calculada vale: %f\n",brun);
  system("PAUSE");
  return;
}

int es_primo (int numero)  // Determina si "numero" es primo
{
    int i,primo=1;        // Variable control de bucle y flag detectar primo
    for (i=2;i<numero;i++)   // Recorrido buscando divisores de numero
      if ((numero % i) == 0) // Si es divisible por alguno
         primo=0;           // no es primo
    return primo;
}

int get_primos(int v[],int n)  // Obtiene los primos entre 1 y n
{
    int i,total=0;        // Control de bucle y primos encontrados
    for (i=1;i<=n;i++)     // Recorrido de los n primeros números naturales
      if (es_primo(i)==1)  // Si i es primo
      {
         v[total]=i;      // Lo almacena en el vector
         total++;         // Incrementa la cantidad de primos encontrados
      }
    return total;         // Devuelve cantidad de primos encontrados
}

int constante_brun (int v[],int m, float *b) // Calcula Brun y pares encontrados
{
    int n_pares=0,i;      // Pares de primos gemelos detectados y control bucle

    *b=0;         // Se inicializa la cte de Brun a cero para acumular.
    for (i=0;i<m-1;i++)       // Para cada número primo excepto el último.
     if (v[i+1]-v[i]==2)      // Se comprueba si es gemelo del siguiente.
     {
        n_pares++;    // Se incrementa el número de pares de primos gemelos.
        *b=*b + (((float)1.0/v[i])+((float)1.0/v[i+1]));  // Se calcula...
        }
    return n_pares;               // Retorna el número de pares detectados.
}
```

SOLUCIÓN EJERCICIO 4

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

void main() {

	FILE *ficheroEntrada;
	FILE *ficheroSalida;
	char nombreFicheroEntrada[50];
	char dni[11];
	float renta;
	int nTutoresParados, nMenores;
	int puntosTotales, puntosRenta;

	printf("Introduce el nombre del fichero de entrada\n");
	scanf("%s", nombreFicheroEntrada);

	ficheroEntrada = fopen(nombreFicheroEntrada, "r");
	if (ficheroEntrada == NULL) {
		printf("No se ha podido leer el fichero de entrada\n");
		system("pause");
		return;
	}
	ficheroSalida = fopen("resultado.txt", "w");
	fprintf(ficheroSalida, "DNI\tP1\t\P2\tP3\tTotal\n");

	while (fscanf(ficheroEntrada, "%s %f %d %d", dni, &renta, &nTutoresParados, &nMenores) != EOF) {
		puntosTotales = 0;
		puntosRenta = 0;
		if (renta < 2501) {
			puntosRenta = 4;
		} else if (renta < 4001) {
			puntosRenta = 3;
		} else if (renta < 5001) {
			puntosRenta = 2;
		} else if (renta < 6001) {
			puntosRenta = 1;
		}
		puntosTotales = puntosRenta + nTutoresParados * 2 + nMenores;
		fprintf(ficheroSalida, "%s\t%d\t%d\t%d\t%d\n", dni, puntosRenta, nTutoresParados * 2,
nMenores, puntosTotales);
	}
	fclose(ficheroEntrada);
	fclose(ficheroSalida);
}
```

SOLUCIÓN EJERCICIO 5

//Prototipos

```
int EsPalindromo (char palabra[]);
void CopiaInvertida (char original[], char copia[]);
int ComparaPalabras (char palabra1[], char palabra2[]);
void LongitudPalabra (char palabra[], int *n);
```

//Definiciones

```
int EsPalindromo (char palabra[])
{
	char reves[20];
	int resultado;

	CopiaInvertida(palabra, reves);
	resultado=ComparaPalabras(palabra, reves);

	return resultado;
}

void CopiaInvertida (char original[], char copia[])
{
	int i, j, longitud;

	LongitudPalabra(original, &longitud);
	j=0;

	for (i=longitud-1; i>=0; i--)
	{
		copia[j]=original[i];
		j++;

	//Finaliza la copia con el carácter nulo
	copia[longitud]='\0';
}

void LongitudPalabra(char palabra[], int *n)
{
	int i=0;
	while(palabra[i]!='\0')
		i++;
	*n=i;
}

int ComparaPalabras(char palabra1[], char palabra2[])
{
	int i=0;

	while(palabra1[i]!='\0')
	{
		if(palabra1[i]!=palabra2[i])
			return 0;
		i++;
	}
	return 1;
}
```