EXAMEN DE FUNDAMENTOS DE PROGRAMACIÓN – 50% TOTAL

Problema 1: (1 punto)

Escribe un programa en C que lea desde el teclado una secuencia no vacía de líneas de entrada de longitud máxima 90 caracteres y escriba por pantalla la de mayor longitud.

NOTAS: No se puede utilizar la librería <string.h> y se deben definir y utilizar las funciones oportunas para el manejo de las cadenas de caracteres.

Solución:

```
#include <stdio.h>
#define MAX_LENGTH 1000

int longitud(char []);
void copiarLinea(char [], char []);

int main(int argc, const char * argv[])
{

    char laMasLarga[MAX_LENGTH];
    char actual[MAX_LENGTH];
    int longitud_actual = 0;
    int longitud_masLarga = 0;

    printf("LA LINEA DE ENTRADA MAS LARGA\n");
    printf("*****************************\n");

    while (!feof(stdin))
    {
        gets(actual);
        if (longitud(actual)>longitud_masLarga)
        {
            copiarLinea(laMasLarga, actual);
            longitud_masLarga = longitud(actual);
        }
    }
    if (longitud_masLarga>0)
        printf("\nLa linea mas larga fue: %s\n",laMasLarga);

    return 0;
}
```

```
void copiarLinea(char to[], char from[])
{
    int i = 0;

    while ((to[i]=from[i]) !='\0') {
        i++;
    }
}

int longitud(char cadena[])
{
    int cont;

    for (cont=0 ; cadena[cont]; cont++);
    return cont;
}
```

Problema 2: (2 puntos)

Dado un fichero de texto de nombre VOTACIONES_2015 en el que aparecen en las 47 primeras líneas el nombre de los 47 partidos políticos que se presentan a las elecciones generales y, a continuación, en cada línea el número de votos que ha recibido en cada una de las 52 circunscripciones electorales de España (provincias) los 47 cada uno de los partidos en el mismo orden en el que aparecen en el fichero.

```
PP

PSOE

CIUDADANOS

PODEMOS

…..

123 123 123 123 4 5 6 7 8 9 12 13 23 43 54 66 77 89 … 4

…
```

Se pide: Escribir un programa en C que:

1. Abra, cierre y procese los registros del fichero adecuadamente.
2. Defina las estructuras de datos correctas necesarias para recoger la información del fichero y procesarla como requiere el apartado 3.
3. Lea todas líneas del fichero y calcule el número total de votos que ha recibido cada partido en España y muestre por pantalla el nombre de los partidos que obtienen un porcentaje de voto mayor al 5% del total de los votos emitidos con el número total de votos recibidos y el porcentaje que le corresponde del total.

Solución:

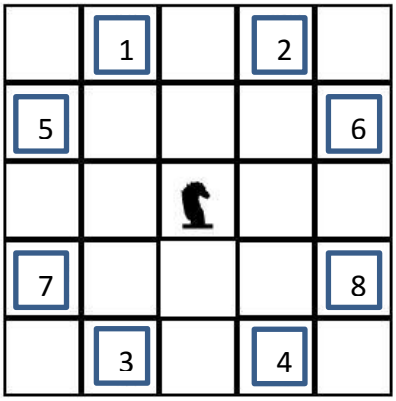
```
#include <stdio.h>
#include <stdlib.h>

#define NUM_PARTIDOS 47
#define NUM_PROVINCIAS 52

struct tPartido
{
    char nombre [15];
    int num_votos;
};

void sumarProvincia(FILE *, struct tPartido[], int *);
```

```
void main(void)
{
    FILE *fp;
    struct tPartido partido[NUM_PARTIDOS];
    int votosEmitidos = 0;

    int i;

    if ((fp = fopen ("VOTACIONES_2015.txt","r")) == NULL)
    {
        printf("Error abriendo fichero\n");
        exit (-1);
    }

    for (i=0; i<NUM_PARTIDOS; i++)
    {
        fscanf(fp,"%s",partido[i].nombre);
        partido[i].num_votos = 0;
    }

    for (i=0; i<NUM_PROVINCIAS; i++)
        sumarProvincia(fp,partido,&votosEmitidos);

    fclose(fp);

    printf("PARTIDO\t\tNUM_VOTOS\tPORCENTAJE OBTENIDO\n");
    printf("************************************************\n");

    for (i=0; i<NUM_PARTIDOS; i++)
    {
        float porcentaje = (float) partido[i].num_votos*100 /
                              votosEmitidos;
        if (porcentaje > 5)
            printf("%s\t\t%d\t\t\t%.2f\n",
                  partido[i].nombre,partido[i].num_votos,porcentaje);
    }
    printf("Total de votos emitidos: %d\n",votosEmitidos);
    printf("\n\nFIN DEL PROCESO ELECTORAL\n");
}
```

```
void sumarProvincia(FILE *fp, struct tPartido partidos[], int * cont)
{
    int i;
    int votos;
    for (i=0; i<NUM_PARTIDOS; i++)
    {
        fscanf(fp,"%d",&votos);
        partidos[i].num_votos += votos;
        *cont += votos;
    }
}
```

Problema 3: (2 puntos)

En el ajedrez, el caballo que se sitúa en la casilla (i,j) de un tablero se puede mover a cualquiera de las siguientes 8 casillas del tablero.



Si simulamos el tablero de ajedrez con una matriz de números enteros que inicialmente están a 0 todos ellos. Situamos el caballo en la casilla (0,0) y, cada vez que el caballo se mueve a otra casilla, la casilla de destino se marca con un 1 indicando que el caballo está en ella o ha estado previamente en ella.

Comenzamos a mover el caballo eligiendo un movimiento en el orden en que se han presentado en la imagen de la manera siguiente: el primer movimiento a considerar es el marcado como 1, si esa casilla no ha sido visitada previamente por el caballo, se marca como visitada poniendo un 1 y la posición del caballo (i , j) pasa a ser (i-1, j+2), si esa posición ya hubiese sido visitada se elige la casilla siguiente, en el orden establecido, hasta que se encuentre una posición que no haya sido visitada, en cuyo caso el caballo se mueve a esa casilla; o en el caso de no encontrar ninguna casilla a 0 (no visitada) el movimiento del caballo para.

Se pide:

1. Definir la función ***sePuedeMover(int tablero [8][8], int *i, int *j),*** que devuelva un 1 si el caballo ha podido desplazarse desde la casilla (i,j) a otra casilla que no había visitado previamente y actualice las posiciones i, j a las posiciones que ocupa el caballo después de ejecutar un movimiento; o 0 si no se puede mover el caballo.
2. Escribir un programa que utilizando la función anterior mueva el caballo por el tablero hasta que sea imposible realizar otro movimiento y, después, escriba si el caballo ha visitado todas las casillas del tablero.

Solución:

```
#include <stdio.h>

#define TAMANIO 8

int sePuedeMover(int [][TAMANIO], int *, int *);
int coordenadaValida(int*, int *, int);
void presentarTablero (int [][TAMANIO]);
```

Publicación de notas, martes 22 de diciembre
Revisión de notas martes 22 de diciembre en clase

```
int main(void) {

    int tablero[TAMANIO][TAMANIO] = {0};
    int i=0, j=0;


    tablero[i][j] = 1;

    while (sePuedeMover(tablero, &i, &j));

    for (i=0; i<TAMANIO; i++)
    {
        for (j=0; j<TAMANIO; j++)
            if (!tablero[i][j])
            {
                printf("El caballo no ha recorrido el tablero
                       completo\n");
                return 0;
            }
    }
    printf("El caballo ha recorrido el tablero completo\n")
    return 0;
}

int sePuedeMover(int tablero [8][8], int *i, int *j)
{
    int movimiento = 1;
    int r = *i, s = *j;
    int valeMovimiento;

    valeMovimiento = coordenadaValida(&r,&s,movimiento);
    while (k<9 && (!valeMovimiento || tablero[r][s]))
    {
        k++;
        r=*i;
        s=*j;
        valeMovimiento = coordenadaValida(&r,&s,movimiento);
    }
    if (valeMovimiento && !tablero[r][s])
    {
        tablero[*i][*j] = 1;
        *i = r;
        *j = s;
        printf("%d %d\n",r,s);
    }
    return valeMovimiento && !tablero[r][s];
}
```

```
int coordenadaValida(int *i, int *j, int movimiento)
{
    switch (movimiento) {
        case 1:
            *i = *i-2;
            *j = *j-1;
            break;
        case 2:
            *i = *i-2;
            *j = *j+1;
            break;
        case 3:
            *i = *i+2;
            *j = *j-1;
            break;
        case 4:
            *i = *i+2;
            *j = *j+1;
            break;
        case 5:
            *i = *i-1;
            *j = *j-2;
            break;
        case 6:
            *i = *i-1;
            *j = *j+2;
            break;
        case 7:
            *i = *i+1;
            *j = *j-2;
            break;
        case 8:
            *i = *i+1;
            *j = *j+2;
            break;
    }
    return *i>=0 && *i<TAMANIO && *j>=0 && *j<TAMANIO;
}
```