

Examen II (35%)

1) (6 pts) Para el siguiente programa, efectúe la corrida “en frío”. Las anotaciones, cálculos y seguimiento algorítmico deben corresponderse con las respuestas

```
#include <stdio.h>
#define N 3
int i, j, count = 0;
int A [N][N];
int main() {
    for (i = 0; i < N; i++)
        for (j = i; j < N; j++) {
            A[i][j] = i + j - 1;
            A[j][i] = -A[i][j];
        }
    for (i = N; i > 0; i--)
        for (j = i; j < N - i; j++) {
            A[i][j] = -A[i][j] % N;
            j = j + i + 1;
            count = count + j;
        }
    printf("+ i = %d, j = %d, count = %d\n", i, j, count);
}
```

y determine cual de las siguientes opciones representa la salida

- a.- $i = 1, j = 2, count = 0$
- b.- $i = 3, j = 3, count = 1$
- c.- $i = 0, j = 4, count = 3$
- d.- $i = 1, j = 1, count = 25$
- e.- $i = 1, j = 3, count = 4$

inicio	i= 3, j= 3, count= 0
externo entra	i= 3, j= 3, count= 0
externo sale	i= 3, j= 3, count= 0
externo entra	i= 2, j= 3, count= 0
externo sale	i= 2, j= 2, count= 0
externo entra	i= 1, j= 2, count= 0
interno entra	i= 1, j= 1, count= 0
interno sale	i= 1, j= 3, count= 3
externo sale	i= 1, j= 4, count= 3
final	i= 0, j= 4, count= 3
Salida:	
- i = 0, j = 4, count = 3	

2) (8 pts) Defina un tipo de datos llamado **fecha_t**, con los campos adecuados para almacenar (en ese orden) el año, mes, día, hora, y minuto.

a) Haga un procedimiento **DiffFecha** que reciba tres valores de tipo **fecha_t** y devuelva en el tercer argumento (parámetro) la diferencia temporal entre ellas, también como **fecha_t**. Asuma meses de 30 días sin años bisiestos.

b) Haga un procedimiento **SumFecha** que reciba tres argumentos todos de tipo **fecha_t**: la fecha base, un desplazamiento temporal, y una fecha resultante que será devuelta como la suma del desplazamiento temporal a la fecha base.

```
typedef struct {
    short anyo;
    char mes; // de 0 a 12
    char dia; // de 0 a 31
    char hora; // de 0 a 24
    char min; // de 0 a 60
} fecha_t;

fecha_t A, B, C, D;

... // se leen valores para A y B
DiffFecha(A, B, D); // Obtenemos D como B - A
SumFecha(A, D, C); // Obtenemos C como A + D
// luego de esto, B y C deben ser la misma fecha

// Este es el que haría todo el mundo
void DiffFecha (fecha_t A, fecha_t B, fecha_t *D) {
    // pre (B>=A)
    *D.min = B.min - A.min;
    *D.hora = B.hora - A.hora;
    if (*D.min < 0) {
        *D.hora--;
        *D.min += 60;
    }
    *D.dia = B.dia - A.dia;
    if (*D.hora < 0) {
        *D.dia--;
        *D.hora += 24;
    }
    *D.mes = B.mes - A.mes;
    if (*D.dia < 0) {
        *D.mes--;
        *D.dia += 30;
    }
    *D.anyo = B.anyo - A.anyo;
    if (*D.mes < 0) {
        *D.anyo--;
        *D.mes += 12;
    }
    _post( *D.anyo >= 0); // si el año dá negativo, fecha A > fecha B;
}
```

```

void SumFecha (fecha_t A, fecha_t B, fecha_t* D) {
    *D.min = A.min + B.min;
    *D.hora = A.hora + B.hora;
    if (*D.min > 60) {
        *D.hora++;
        *D.min -= 60;
    }
    *D.dia = A.dia + B.dia;
    if (*D.hora > 24) {
        *D.dia++;
        *D.hora -= 24;
    }
    *D.mes = A.mes + B.mes;
    if (*D.dia > 30) {
        *D.mes++;
        *D.dia -= 30;
    }
    *D.anyo = A.anyo + B.anyo;
    if (*D.mes > 12) {
        *D.anyo++;
        *D.mes -= 12;
    }
}

/* Este es el que haria yo ... */

// convertir todo a minutos
void DifFecha (fecha_t A, fecha_t B, fecha_t* D) {
    int ma = A.min + 60*(A.hora + 24*(A.dia + 30*(A.mes + 12*A.anyo)));
    int mb = B.min + 60*(B.hora + 24*(B.dia + 30*(B.mes + 12*B.anyo)));
    // min + 60*hora + 1440*dia + 43200*mes + 518400*anyo;
    int md = mb - ma;

    _pre (md >= 0); // por si quieren chequear la precondition
    *D.anyo = md / (12*30*24*60); // 518400
    md = md % (12*30*24*60);
    *D.mes = md / (30*24*60); // 43200
    md = md % (30*24*60);
    *D.dia = md / (24*60); // 1440
    md = md % (24*60);
    *D.hora = md / (60);
    *D.min = md % (60);
}

void SumFecha (fecha_t A, fecha_t B, fecha_t* D) {
    int ma = A.min + 60*(A.hora + 24*(A.dia + 30*(A.mes + 12*A.anyo)));
    int mb = B.min + 60*(B.hora + 24*(B.dia + 30*(B.mes + 12*B.anyo)));
    int md = mb + ma;

    *D.anyo = md / (12*30*24*60); // 518400
    md = md % (12*30*24*60);
    *D.mes = md / (30*24*60); // 43200
    md = md % (30*24*60);
    *D.dia = md / (24*60); // 1440
    md = md % (24*60);
    *D.hora = md / (60);
    *D.min = md % (60);
}

```

3) (8 pts) Corrida en “frío”: Diga cual es la salida al ejecutar el siguiente programa. Las anotaciones, cálculos y seguimiento algorítmico deben corresponderse con las respuestas

```
#include <stdio.h>

int y = 1;

void i (int k, int* j) {
    *j = (*j + 1)*3;
}

void j (int* a, int k) {
    i(k, a);
}

int top (int x) {
    return (x + 1);
}

void otro (int j, float x) {
    y = -j*x;
    j++;
}

int main ( ) {
    int k = -1;
    int j = -2;
    float p = 3.14159265;

    otro(top(k), p);
    printf ("y = %i, j = %i, k = %i \n", y, j, k);

    i(y,&j);
    printf ("y = %i, j = %i, k = %i \n", y, j, k);
}


```

Salida:

y = 0, j = -2, k = -1

y = 0, j = -3, k = -1

Salida:

4) (13 pts) Se necesita elaborar un programa que maneje los *pensa* de cada carrera, de la manera usada en la USB. Para esto, Ud. debe diseñar las estructuras de datos necesarias, leer de archivo los datos necesarios y construir el pensum de una carrera de la universidad.

Por cada carrera existe un archivo contentivo del pensum de nombre "*xxx.pen*", donde *xxx* es el código de la carrera (ej. *iel.pen*). Cada archivo del pensum contiene la siguiente información en cada línea, correspondiente a cada materia del pensum, ordenado por trimestre:

<trimestre (num de 1 a 15), código materia (6 caracteres, los 2 primeros el código del departamento y los 4 últimos dígitos), horas teoría, horas práctica, horas lab, créditos (enteros), num pre-requisitos (enteros), códigos de materia requisito (tantos como el anterior diga, máximo 2), tiene_co-requisitos?, código co-requisito (en caso de que el anterior sea verdadero)>.

Usted debe:

- Declarar las estructuras de datos necesarias para guardar un pensum.
- Escribir una función que reciba un nombre de archivo y devuelva el arreglo de estructuras de datos correspondientes al pensum.
- Escribir función de búsqueda que dado un código de materia, localice en el arreglo de estructuras del pensum el registro correspondiente y devuelva un apuntador a ese registro.
- Escribir una función que dado un trimestre cualquiera del pensum, calcule el acumulado de horas (T, P, y L) y la suma de créditos para ese trimestre.

SOLUCION

```
#include <stdio.h>

typedef struct {
    char Teoria;
    char Practica;
    char Laboratorio;
} horas_t;

typedef struct {
    char Trimestre;
    char[6] CodMateria;
    horas_t HorasSemana;
    char NumCreditos;
    char NumPrereq;
    codmat_t[2] Prereq;
    char Tiene_Correq;
    codmat_t Correq;
} materia_t;

// suponiendo que las cosas viene separadas por comas ","
// 10 CI2125 2 2 3 4 2 2 MA1111 FI1112 1 MA3115
```

```

int leerPensum(char* elNombre, materia_t* elPensum) {
    char sep; // para leer espacios en blanco
    FILE *archivo;
    int cuenta = 0, k;
    archivo = fopen( elNombre, "r" );
    if (archivo) {
        while (!feof(archivo)) {
            fscanf(archivo, "%i", &elPensum[cuenta].Trimestre);
            fscanf(archivo, "%c", &sep);
            fscanf(archivo, "%s", &elPensum[cuenta].CodMateria);
            fscanf(archivo, "%i", &elPensum[cuenta].HorasSemana.Teoria);
            fscanf(archivo, "%i", &elPensum[cuenta].HorasSemana.Practica);
            fscanf(archivo, "%i", &elPensum[cuenta].HorasSemana.Laboratorio);
            fscanf(archivo, "%i", &elPensum[cuenta].NumCreditos);
            fscanf(archivo, "%i", &elPensum[cuenta].NumPrereq);
            for (k = 0; k < elPensum[cuenta].NumPrereq; k++) {
                fscanf(archivo, "%c", &sep);
                fscanf(archivo, "%s", &elPensum[cuenta].Prereq[k]);
            }
            fscanf(archivo, "%i", &elPensum[cuenta].Tiene_Correq);
            if (elPensum[cuenta].Tiene_Correq) {
                fscanf(archivo, "%c", &sep);
                fscanf(archivo, "%s", &elPensum[cuenta].Correq[k]);
            }
            cuenta++;
        }
        return cuenta; // Ultimo = cuenta
    } else {
        printf( "Error: archivo %s NO se encuentra\n" );
        return -1;
    }
}

int buscarMateria(char[6] codigo) {
    int cuenta = 0;
    int fin = false;
    while (!fin) && (cuenta < Ultimo) {
        switch strcmp(codigo, elPensum[cuenta].CodMateria) {
            case 0: fin = true; return (cuenta);
            case 1: fin = true; return -1;
            default: cuenta++;
        }
    };
    return -1; // si llega aquí, es un error
}

void calcularAcumulado(char trimestre, int* totalHoras, int* totalCreditos) {
    int cuenta = 0;
    *totalHoras = 0;
    *totalCreditos = 0;

    for (cuenta = 0; cuenta < Ultimo; cuenta++) {
        if (elPensum[cuenta].Trimestre == trimestre) {
            *totalHoras += elPensum[cuenta].HorasSemana.Teoria +
                elPensum[cuenta].HorasSemana.Practica +
                elPensum[cuenta].HorasSemana.Laboratorio;
            *totalCreditos += elPensum[cuenta].NumCreditos;
        }
    }
    return;
}

```