# TOTALIZADOR PROGRAMACION II

**6.3.2023**

**Apellido, Nombre:** GARCIA GENGA GZEQuiel

*Serán considerados al calificar este examen la eficiencia de las soluciones y la utilización adecuada de las características del lenguaje C y de la programación estructurada.*

*Para aprobar es necesario obtener al menos 5 puntos. Y al menos el 25% de cada uno de los ejercicios 2, 3 y 4. Y al menos 4,25 entre los ejercicios 2, 3 y 4.*

*Cuando este examen está aprobado, la nota FINAL se obtiene así: CURSADA * 0.3 + TOTALIZADOR * 0.7*

*En todos los ejercicios que corresponda, mostrar las invocaciones (incluyendo su contexto: declaraciones de variables y tipos, inicializaciones y acciones posteriores) de las soluciones desarrolladas.*

| Ej 1 (1,5 p) | Ej 2 (4 p) | Ej 3 (2 p) | Ej 4 (2,5 p) | NOTA | CURSADA | FINAL (*) |
|---|---|---|---|---|---|---|
| — | 0,80 | 1,25 | 1 | 2 | — | 2 |

**Ej 1.-** Indicar V o F, justificando o ejemplificando adecuadamente (de lo contrario tendrá puntaje cero)

**a)** La implementación de una pila en memoria estática podría hacerse inicializando el primero en MAX en lugar de en -1 (siendo MAX la dimensión física del vector).

**b)** Si se aplica el recorrido en profundidad sobre un digrafo no conexo el mismo podría fallar al vaciarse la pila y no conseguir llegar a un vértice no visitado de otra componente conexa.

**Ej 2.-** Una consultora de RRHH para proyectos tecnológicos maneja una lista simplemente enlazada con Sublistas con la siguiente información en cada nodo:

- Id Persona (ordenado, no se repite, cadena de 6)
- Rol (0..9, 0 indica que no tiene proyecto, sino es el rol en el proyecto actual, una persona no puede estar en más de un proyecto a la vez o en más de un rol en el mismo proyecto a la vez)
- Fecha inicio (cadena de 8, aaaammdd, vacío si no tiene proyecto)
- Sublista de proyectos (puede estar vacía, son los roles/proyectos en los que concluyó su participación), en cada nodo: *Empresa* (puede repetirse, ordenada, cadena de 10, si inicia con # es extranjera), *Rol* (1..9), *Meses que trabajó* (entero corto)

Se pide resolver, modularizando:

**a)** En un archivo de texto **FEBRERO.TXT**, se encuentra la información de las personas (dato correcto) que participan en proyectos en el mes de febrero de 2023. En cada línea (separando cada dato por un blanco) se tiene: *IdPersona* (ordenado, cadena de 6), *Empresa* (cadena de 10), *Rol*, *Terminó [S/N]*. Con la información del archivo, actualizar la lista y sublistas según corresponda.

NOTA: Suponer la existencia de una función CANTM, en la librería fechas.h que recibe dos cadenas de formato *aaaamm* y retorna la diferencia en meses entre ambas. NO DESARROLLAR CANTM

S/R **b)** Dados un P y E, eliminar todos los proyectos asociados a la empresa E de la persona P (puede o no existir). Al finalizar, indicar en cuántos proyectos de E la persona P ha concluido su participación. NO eliminar el nodo de la lista si quedara sin proyectos

S/R **c)** Generar una lista doble que en cada nodo tenga *Empresa* (ordenada, cadena de 10) y *CantP* en la que figuren solo las empresas extranjeras siendo *CantP* la cantidad de personas distintas que ha contratado.

**Ej 3.-** (*Utilizar TDA N-Ario*) Dados un árbol AN N-Ario de enteros y un árbol binario AB que proviene de la transformación de un bosque de enteros, desarrollar una función *int* que retorne la cantidad de árboles del bosque que contenían una cantidad de claves que estaba en AN en el nivel K (K dato). (O sea, si la cantidad de claves en un árbol de AB era X, X está en AN en el nivel K)

- Si la solución se resuelve mediante una función void, el puntaje obtenido no superará la mitad del asignado

**Ej 4.-** (*Utilizar TDA Pila*) Dada una matriz de adyacencia que representa un digrafo acíclico de N vértices (numerados de 1 a N), con aristas ponderadas y una Pila P que contiene en cada elemento V (un vértice 1 a N, ordenada) y un G, determinar recursivamente sobre al menos una de las estructuras si todos los V tienen grado de salida G. Definir el tipo de la Pila estática y desarrollar las funciones utilizadas. NOTA: P puede perderse

S/R

Marzo 2023

1 - V o F

a) Si podría implementarse una pila estática inicializando el tope en MAX. En ese caso la pila estaría vacía. Luego, se decrementaría el tope al poner elementos, incrementandolo al sacar

b) Verdadero, si el grafo no es conexo, el recorrido en profundidad fallará ya que no se podrán visitar todos los vertices

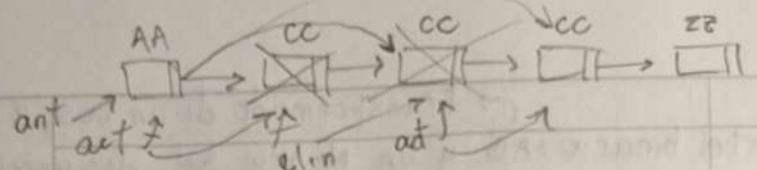
2- Consultora de recursos humanos. Lista SE con sublistas

nodoL:
- IdPersona ordenado, no se repite, cadena de 6
- Rol (0..9) 0 → no tiene proyecto, sino → rol en el proy. actual
- Fecha inicio cadena de 8
- Sublista de proyectos roles/proyectos concluidos
  - → nodos:
    - Empresa puede repetirse, ordenada, cad 10, E=>#
    - Rol
    - Meses que trabajo

main.c

```
#include "fechas.h"
#define ID 7   #define FEC 9   #define EMP 11
typedef struct nodito{          // SUBLISTA PROYECTOS
   Char Empresa[EMP];
   Unsigned Short int Rol, Meses;
   struct nodito *sig; } nodito;
typedef nodito *TSub;
typedef struct nodo{            // LISTA DE PERSONAS
   Char IdPersona[10];
   Unsigned short int Rol;
   Char FechaInicio[FEC];
   TSub SubProyectos;
   struct nodo *sig; } nodo;
typedef nodo *Tlista;
typedef struct nodoD{
   Char Empresa[EMP];
   int CantP;
   struct nodoD *ant, *sig; } nodoD;
typedef nodoD *PnodoD;
typedef struct {
   PnodoD Pri, Ult; } TlistaD;
```

a)
```
void Actualiza(Tlista L){
    FILE *arch;
    Tlista actL;
    Tsub ants, acts, nuevito;
    Char IdPersona[ID], Empresa[EMP], Termino; Unsigned int Rol;
    if ((arch = fopen("FEBEREO.TXT", "rt")) == NULL)
        Printf("No pudo abrirse el archivo");
    else{
        while ((fscanf(arch,"%s %s %u %c", IdPersona, Empresa, Rol, Termino) ==4){
            actL= L;
            While (strcmp(actL→IdPersona, IdPersona)!=0)
                actL = actL→sig;
            if (Termino =='N')
                actL→Rol = Rol;
            else{
                nuevito = (Tsub) malloc(sizeof(nodito));
                strcpy(nuevito→Empresa, Empresa);
                nuevito→Rol = Rol;
                nuevito→Meses = CANTM(actL→FichaInicio, "202302");
                acts = actL→SubProyectos;
                While(acts!=NULL && strcmp(acts→Empresa, Empresa)<0){
                    ants = acts;
                    acts = acts→sig;
                }
                if (acts == actL→SubProyectos)
                    actL→SubProyectos = nuevito;
                else
                    ant→sig = nuevito;
                nuevito→sig = actL;
            }
        }
        fclose(arch);
    }
}
```




b) void EliminaProyectos(Tlista L, char P[], char E[], int *Concluidos){
```
    Tlista actL;
    TSub ants, acts, elim;
    *Concluidos = -1;   // si vuelve con -1 significa que no existe la persona
    actL = L;
    while(actL!=NULL && strcmp(P, actL->IdPersona)>0)
        actL = actL->sig;
    if (actL!=NULL && strcmp(P, actL->IdPersona)==0){
        *Concluidos = 0;
        acts = actL->subproyectos;
        while (acts!=NULL && strcmp(E, acts->Empresa)>0){
            ants = acts;
            acts = acts->sig;
        }
        while(acts!=NULL && strcmp(E, acts->Empresa)==0){
            elim = acts;
            acts = acts->sig;
            free(elim);
            ants->sig = act;
            *Concluidos += 1;
        }
    }
}
```

c) void GenraLD(TlistaD *LD, Tlista L){
```
    PnodoD act; int Trabajo
    TSub acts;
    while (L!=NULL){
      acts = L->SubProductos;
      while (acts!=NULL){
        if (acts->Empresa[0] == '#'){
          act = (*LD).pri;
          while (act!=NULL && strcmp(act->Empresa, acts->Empresa)<0)
              act = act->sig;
          if (acts!=NULL && strcmp(act->Empresa, acts->Empresa)==0)
              act->CantP += 1;
          else
              CreaInserta(LD, acts->Empresa)
          while(acts!=NULL && strcmp(acts->Empresa, EmpresaAnt)==0)
              acts = act->sig;
        else
          acts = acts->sig;
        }
        L = L->sig;
      }
}
```

ya que no debo contar la persona mas de una vez

↳ transformado de un bosque

3- Dados un arbol binario AB y un N-Ario AN, desarrollar una función int que retorne la cantidad de arboles del bosque que contenian una cantidad de claves que estaba en el AN en el nivel K.

```
int EstaEnK(ArbolN AN, posicion P, int CantClaves, int nivel, int K){
    if (!Nulo(P))
        if (Nivel == K)
            if (CantClaves == Info(P,AN));
                return 1;
            else
                return EstaEnK(AN, HnoDer(P,AN), CantClaves, nivel, K);
        else
            return EstaEnK(AN, HijoMasIzq(P,AN), cantclaves, nivel+1, K) ||
                   EstaEnK(AN, HnoDer(P,AN), CantClaves, nivel, K);
    else
        return 0;
}
int CuentaClaves(arbol AB){
    if (AB != NULL)
        return 1+ CuentaClaves(AB->izq) + CuentaClaves(AB->der);
    else
        return 0;
}
int CantArboles(arbol bosque, ArbolN AN, int K){
    arbol aux
    int Cont, CantClaves;
    Cont = 0;
    CantClaves = 0;
    While (bosque != NULL){
        CantClaves = 1 + CuentaClaves(AB->izq);
        Cont += EstaEnK(AN, Raiz(AN), CantClaves, 1, K);
        bosque = bosque->der;
    }
    return cont;
}
```

4- Digrafo acíclico con aristas ponderadas y una pila que contiene un vertice y un G en cada elemento, determinar si todos los v tienen grado de salida G

```
void TodosCumplen(int Mat[][MAX], int i, int j, int N, TPila *P, int GS, int *Cumplen){
    TelementoP reg;
    if (i == N)
        *Cumplen=1;
    else
        if (j == N){
            Sacar(P, &reg);
            if ( GS = reg.G)
                TodosCumplen(Mat, i+1, 0, N, P, 0, cumplen);
            else
                *Cumplen = 0;
        }
        else
            TodosCumplen(Mat, i, j+1, N, P, GS+Mat[i][j]!=0, cumplen);
}
```