TOTALIZADOR PROGRAMACION II

19.12.2022

Apellido, Nombre: Russo Fernando

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 (*)
1	1,25	0,50	1,25	4	—	2

Ej 1.- Indicar V o F, justificando o ejemplificando adecuadamente (de lo contrario tendrá puntaje cero)
a) Al finalizar la ejecución del algoritmo de Floyd sobre un digrafo, si el mismo es no conexo en la diagonal habrá posiciones con valor 0 y otras con valor *infinito*.
b) Al insertar una clave en un árbol AVL, se deben calcular los f.e. desde la raíz hasta la clave insertada (*a la ida*) en el proceso de inserción.

Ej 2.- Una consultora de certificación de normas ISO maneja una lista circular con las normas (0..20) y las empresas (0 a 39) que ha certificado (una empresa puede haber certificado una o más normas). Cada nodo de esta lista contiene:

- Norma (ordenada, no se repite)
- NombreNorma (Cadena8)
- Sublista de empresas certificadas, en cada nodo:
  - NúmeroEmpresa
  - FechaCertificación (Cadena de formato aaaamm, ordenada por este campo)

Se pide:

a) Generar una matriz de 21x40 caracteres, tal que en la posición [i,j] contenga S o N según si la norma i fue certificada o no por la empresa j. Hallar de forma recursiva mediante una función int la cantidad de empresas que han certificado todas las normas.

b) Dados un mes MM (cadena) y una norma N, eliminar todas las empresas que hayan certificado N en el mes MM. Si la norma quedara sin empresas certificadas, eliminarla también. A medida que se efectúa el proceso de eliminación, generar un archivo binario ELIM.DAT de registros con la siguiente información para las empresas eliminadas: *NumeroEmpresa, Norma, FechaCertificación*

Ej 3.- (*Utilizar TDA N-Ario*) Se tiene un árbol N-Ario de enteros positivos, hallar mediante una función recursiva int el valor de la menor clave no hoja del nivel, si no existe ninguna devolver -1. (k dado).

Ej 4.- Se tienen: un grafo G con N vértices y aristas ponderadas almacenado en la mitad inferior de una matriz de adyacencia M y una Cola C en la que en cada elemento se tiene *id1 e id2* (1 a N) *y cos*, siendo *id1* e *id2* vértices de G y cos un valor mayor a 0. Se pide,
a) dejar en C solo los elementos que verifiquen que *id1* e *id2* están unidos por una arista con costo a lo sumo *cos*.
b) asumiendo que C está implementada de modo circular, escribir el tipo y los operadores utilizados en la solución

Diciembre 2022

1- V o F

a) FALSO. La diagonal solo contendrá 0's.

b) FALSO. Se recalculan los factores de equilibrio a la vuelta.

2- Consultora de certificación de normas Iso

LISTA CIRCULAR { Norma (0..20, ordenada)
NombreNorma (cadena de 8)
Sublista de empresas certificadas { Numero empresa (0..39)
Ficha certif (aaaammm, ordenado) }

a) Generar una matriz de 21 x 40 caracteres tal que en la posicion [i,j] contenga S o N según si la norma i fue certificada por la empresa j

```
void GeneraMatriz(TlistaC LC, int N, char Mat[][MAX], int E){
    TSub auxE; //empresa
    TlistaC auxN; // norma
    Inicializa(Mat, N, E); // Pone un 'N' en cada posición de la matriz
    if (LC!=NULL){
        auxN = LC->sig;
        do{
            auxE = auxN->SubEmpresas;
            while(auxE!=NULL){
                Mat[auxN->Norma][auxE->NumeroEmpresa] = 'S';
                auxE = auxE->sig;
            }
            auxN = auxN->sig;
        }while(auxN!=LC->sig);
    }
}
```

Hallar de forma recursiva mediante una funcion int la cantidad de empresas que han certificado todas las normas

```
int CantCertTodas(Char Mat[][MAX], int i, int j, int E){
    if (i==-1)
      return 0;
    else
      if (j==-1)
        return 1+CantCertTodas(Mat, i-1, E, E);
      else
        if (Mat[i][j]=='s')
          return CantCertTodas(Mat, i, j-1, E);
        else
          return CantCertTodas(Mat, i-1, E, E);
}
//Cont = CantCertTodas(Mat, N, E, E);
```

b) Dados un mes MM y una norma N, eliminar todas las empresas que hayan certificado N en el mes MM. Si la norma quedara sin empresas certificadas eliminarla tambien. A medida que se efectua el proceso de eliminacion, generar un archivo binario ELIM.DAT con: NumeroEmpresa, Norma, FechaCertificacion, para las empresas eliminadas

```
#include <stdio.h> #include <stdlib.h> #include <string.h>
#define MAX 100 #define NOM 9 #define FEC 7 #define MES 3

typedef struct nodito{
   int NumroEmpresa;
   Char FecCert[FEC];
   struct nodito *sig;} nodito;

typedef nodito *Tsub;

typedef struct{
   int Norma, NumroEmpresa;
   Char FecCert[FEC];} Treg;

typedef struct nodo{
      int Norma;
      Char NombreNorma[NOM];
      Tsub SubEmpresas;
      Struct nodo *sig;}nodo;

typedef nodo *TlistaC;
```

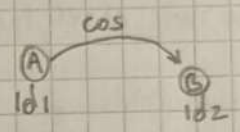
```
void EliminaEmpresas(TlistaC *LC, char MM[], int Norma){
  FILE *archb;
  Treg Relim;
  TlistaC antC, actC;
  TSub antS, actS, elimS;
  if ((archb=fopen("ELIM.DAT","wb"))==NULL)
     Printf("No pudo abrirse el archivo");
  else{
     if(*LC!=NULL){
        antC=*LC;
        actC=(*LC)->sig;
        do{
          if(actC->Norma==Norma){
             actS=actC->SubEmpresas;
             while(auxS!=NULL){
                if(auxS->FecEmp[4]==MM[0] && auxS->FecEmp[5]==MM[1]){
                   Relim.NumeroEmpresa=auxS->NumeroEmpresa;
                   Relim.Norma=Norma;
                   strcpy(Relim.FecCurt,auxS->FecCurt);
                   fwrite(&Relim, sizeof(Treg), 1, archb);
                   elimS=actS;
                   actS=actS->sig;
                   if(elimS==actC->SubEmpresas)
                      actC->SubEmpresas=actS;
                   else
                      antS->sig=actS;
                   Free(elimS);
                }else{
                   antS=actS;
                   actS=actS->sig;
                }
             }
             if(actC->SubEmpresas==NULL){ // Debo eliminar la norma
                if(actC==*LC)
                   if(actC==actC->sig)
                      *LC=NULL;
                   else
                      *LC=antC;
                antC->sig=actC->sig; Free(actC);
             }else{antC=actC; actC=actC->sig;
          }while(*LC!=NULL && actC!=(*LC) && Norma>actC->Norma);
        fclose(archb);
     }
  }
}
```

3- Se tiene un arbol N-ario de enteros positivos, hallar mediante una funcion recursiva int el valor de la menor clave no hoja del nivel K, Si no existe ninguna devolver -1.

```
int MenorNoHojaK (ArbolN A, Posicion P, int nivel, int K){
    Posicion C,h
    int resultado, min, clave, trae;
    if (!Nulo(P)){
       C = HijoMasIzq(P,A);
       min = 9999;
       if (nivel == K-1){
          while(!Nulo(C)){
             clave = info(P,A);
             if (clave < min && !Nulo(HijoMasIzq(C,A))
                min = clave;
             C = HnoDer(C,A);
          }
          resultado = min != 9999 ? clave : -1;
       }
       else{
          while(!nulo(C)){
             trae = MenorNoHojaK(A, C, nivel+1, K);
             if (trae < min && trae != -1)
                min = trae;
             C = HnoDer(C,A);
          }
          resultado = min != 9999 ? trae : -1;
       }
    }
    else
       resultado = -1;
}
```

4- Grafo con N vértices y aristas ponderadas almacenado en la mitad inferior de una matriz y una cola en la que cada elemento: id1, id2 (1 a N) y cos (>0)

a) dejar en C solo los elementos que verifiquen que id1 e id2 están unidos por una arista con costo a lo sumo cos




```
void ActualizaC(int Mat[][MAX], int N, TCola *C){
   TElementoC rinfo; int i,j;
   rinfo.cos = 0;
   PonerC(C, rinfo);
   SacaC(C, &rinfo);
   while(rinfo.cos != 0){
      if (rinfo.id1 > rinfo.id2){
         i = rinfo.id1 - 1;
         j = rinfo.id2
      }
      else{
         i = rinfo.id2 - 1;
         j = rinfo.id1 - 1;
      }
      if (Mat[i][j] < rinfo.cos)
         PonerC(C, rinfo);
      }
      SacaC(C, &rinfo);
   }
}
```