

TOTALIZADOR PROGRAMACION II

15.3.2021

Apellido, Nombre: [REDACTED]

Calificación [REDACTED]

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 en este examen y al menos 4,25 deben obtenerse entre los ejercicios 2, 3 y 4.

Cuando este examen está aprobado, la nota FINAL se obtiene así: $\text{CURSADA} * 0.3 + \text{TOTALIZADOR} * 0.7$

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

Ej 1.- Indicar Verdadero o Falso, justificando o ejemplificando adecuadamente (de lo contrario tendrá puntaje cero)

a) La implementación circular de Pila sólo tiene sentido en memoria estática.

b) Luego de una rotación simple en un árbol AVL, si el mismo sigue estando desbalanceado implica que debió aplicarse una rotación compuesta.

Ej 2.- Una escuela gestiona las inscripciones de sus alumnos mediante una lista circular donde cada nodo contiene:

- Letralnicial (caracter, ordenado ascendente)
- Cantidad de inscriptos
- Pila de alumnos (cada elemento es una cadena corresponde al apellido y nombres -ANU25- de un alumno que comienza con Letralnicial)

Definir tipos de la lista y resolver:

i) Se ha recibido el archivo de texto INSCRIPTOS20210312.TXT que posee el apellido y nombre de un alumno por línea (no están ordenados) con los inscriptos en el día especificado en el nombre del archivo. Utilizando el TDA Pila, procesar el archivo agregando cada alumno a la pila correspondiente, actualizando la cantidad de inscriptos. Si no existe la letra inicial en la lista, insertarla.

ii) El alumno A (dato de entrada) pide la baja como inscripto por motivos personales. Extraer de la pila que corresponda solamente al alumno A dejando los demás en el orden que tenían, actualizar cantidad de inscriptos, y si no quedaran inscriptos con la inicial de A, eliminar el nodo de la lista.

iii) Definir el tipo de la Pila y desarrollar los operadores SacaP y VaciaP, suponiendo la pila implementada estáticamente.

Ej 3.- (Utilizar TDA N-Ario) Se tiene un ABB AB (sin elementos repetidos) y un árbol N-ario AN; ambos de enteros. Desarrollar una solución que obtenga qué porcentaje de los elementos de AN que tienen grado impar están en AB en un nivel menor a K (K dato de entrada).

Ej 4.- Dado un grafo con aristas ponderadas implementado en media matriz de adyacencia (el triángulo superior), desarrollar una función entera que retorne el vértice de grado K (dato de entrada) con menor costo total en las aristas que lleguen a él (suponer único).

Ej 1 (1,5 pto.)	Ej 2 (3,5 pto.)	Ej 3 (2,5 pto.)	Ej 4 (2,5 pto.)	NOTA	CURSADA	FINAL

Marzo 2021

1 - V o F

a) La implementación circular de Pila solo tiene sentido en memoria estática FALSO

La implementación circular para una pila no tiene sentido

b) Luego de una rotación simple en un árbol AVL, si sigue estando desbalanceado implica que debió aplicarse una rotación compuesta
No, puede requerir mas de una rotación sin importar si las anteriores fueron simples o compuestas

2- Escuela gestiona las inscripciones de sus alumnos mediante una lista circular

TDAPila.h

```
#define LON 26 #define MAX 100
typedef struct {
    Char nombre[LON]; } TElementoP;
typedef struct {
    TElementoP datos[MAX];
    int tope; } TPila;
void Sacar(TPila *, TElementoP *); }
int VacioP(TPila);
```

TDAPila.c

```
#include <stdio.h> #include "TDAPila.h"
int VacioP(TPila P) {
    return P.tope == -1;
}
void Sacar(TPila *P, TElementoP *P) {
    if (!VacioP(*P)) {
        *X = (*P).datos[P.tope--];
    }
}
```

main.c

```
#include <stdio.h> #include <stdlib.h> #include <string.h>
#include "TDAPila.h"
typedef struct nodo {
    Char LetraInicial;
    unsigned int CantIns;
    TPila Palum;
    struct nodo *sig; } nodo;
typedef nodo * Tlista;
```

i) Procesar el archivo de inscriptos agregando cada uno a la pila corresp.

```
void ProcesarInscriptos(TLista *LI, char NombArch[]) {
```

```
FILE *Arch;
```

```
TLista ant, act, nuevo;
```

```
TElemento Palum;
```

```
if ((Arch = fopen(NombArch, "rt")) == NULL)
```

```
    printf("No pudo abrirse el archivo");
```

```
else {
```

```
    while ((fscanf(Arch, "%s\n", Palum.nombre) == 1) {
```

```
        act = *LI;
```

```
        while (act != NULL && Palum.nombre[0] > act->letra[nicial]) {
```

```
            ant = act;
```

```
            act = act->sig;
```

```
        }
```

```
        if (act == NULL || Palum.nombre[0] != act->letra[nicial]) {
```

```
            nuevo = (TLista) malloc(sizeof(nodo));
```

```
            IniciaP(&(nuevo->Palum));
```

```
            nuevo->CantIns = 1;
```

```
            nuevo->letra[nicial] = Palum.nombre[0];
```

```
            nuevo->sig = act;
```

```
            if (act == *LI)
```

```
                *LI = nuevo;
```

```
            else
```

```
                ant->sig = nuevo;
```

```
            act = nuevo;
```

```
        }
```

```
    else
```

```
        act->CantIns += 1;
```

```
        poneP(&(act->Palum), Palum);
```

```
    }
```

```
    fclose(Arch);
```

```
}
```

```
}
```


ii) El alumno A (dato) pide la baja. Extraer de la pila que corresponda, actualizar la cantidad de inscriptos, y si no quedaron eliminar el nodo

```
void EliminaA(TPila *Palum, Char A[], int *elimine) {
```

```
    TElemento Palum;
```

```
    if (!VaciaP(*P)) {
```

```
        Sacar(P, &Palum);
```

```
        if (strcmp(A, Palum.nombre == 0)
```

```
            *elimine = 1;
```

```
        else {
```

```
            EliminaA(Palum, A, elimine);
```

```
            PonP(Palum, Palum);
```

```
        }
```

```
    }
```

```
void BuscaA(Tlista *LI, Char A[]) {
```

```
    int elimine;
```

```
    Tlista ant, act;
```

```
    act = *LI;
```

```
    while (act != NULL && A[0] > act->LetraInicial) {
```

```
        ant = act;
```

```
        act = act->sig;
```

```
    }
```

```
    if (act != NULL && A[0] == act->LetraInicial) {
```

```
        elimine = 0;
```

```
        EliminaA(&(act->Palum), A, &elimine);
```

```
        act->CantIns -= elimine;
```

```
        if (act->CantIns == 0) {
```

```
            if (act == *LI)
```

```
                *LI = act->sig;
```

```
            else
```

```
                ant->sig = act->sig;
```

```
            free(act);
```

```
        }
```

```
    }
```

```
}
```

3- Se tiene un arbol ABB (sin repetidos) y un arbol N-ario AN. ambos enteros. Obtener qué porcentaje de los elementos de AN que tienen grado impar están en AB en un nivel menor a K.

```
int EstaEnMenorAK(Arbol ABB, int nivel, int K, int elemento){
```

```
    if (ABB != NULL)
```

```
        if (ABB->dato != elemento)
```

```
            if (nivel < K-1)
```

```
                if (ABB->dato > elemento)
```

```
                    return EstaEnMenorAK(ABB->izq, nivel+1, K, elemento);
```

```
                else
```

```
                    return EstaEnMenorAK(ABB->der, nivel+1, K, elemento);
```

```
            else
```

```
                return 0;
```

```
        else
```

```
            return 1;
```

```
    else
```

```
        return 0;
```

```
}
```

```
void Porcentaje(ArbolN AN, Posicion P, int *impares, int *Cumplen, arbol ABB, int K){
```

```
    Posicion C; int grado;
```

```
    if (!nulo(P)) {
```

```
        C = HijoMasIzq(P, A); grado = 0;
```

```
        while (!nulo(C)) {
```

```
            grado ++;
```

```
            Porcentaje(AN, C, impares, cumplen);
```

```
            C = HijoDer(C, A);
```

```
        }
```

```
        if (grado % 2) {
```

```
            (*impares) ++;
```

```
            *Cumplen += EstaEnMenorAK(ABB, 1, K, Info(P, A));
```

```
        }
```

```
    }
```

```
}
```


4- Dado un grafo con aristas ponderadas implementado en matriz (V), desarrollar una función entera que retorne el vertice de grado K con menor costo total en las aristas que lleguen a el

```
int VMenorE (int Mat[ ][MAX], int CantV, int GEK) {
```

```
    int VG[MAX] = {0}, VC[MAX] = {0};
```

```
    int Vmenor, Cmenor, i, j;
```

```
    for (i = 0; i < CantV; i++)
```

```
        for (j = i; j < CantV; j++)
```

```
            if (Mat[i][j] != 0) {
```

```
                VG[j] += 1;
```

```
                VC[j] += Mat[i][j];
```

```
                if (j != i) {
```

```
                    VG[i] += 1;
```

```
                    VC[i] += Mat[i][j];
```

```
            }
```

```
        }
```

```
    Cmenor = 999;
```

```
    for (i = 0; i < CantV; i++)
```

```
        if (VG[i] == GEK && VC[i] < Cmenor) {
```

```
            Vmenor = i + 1;
```

```
            Cmenor = VC[i];
```

```
        }
```

```
    return Vmenor;
```

```
}
```