Universidad Nacional de Salta - Facultad de Ciencias Exactas
Carrera: LAS TUP - Asignatura: REDICTADO INTENSIVO DE PROGRAMACIÓN

SEGUNDO EXAMEN PARCIAL
07/03/2020

Apellido y nombres:	L.U.:	Comisión:

El examen se aprueba con un mínimo de 60 puntos

Ejercicio	1	2	Nota
Puntaje asignado	50	50	
Puntaje obtenido			

IMPROTANTE: Tome un tiempo para leer las consignas, comprenderlas y consultar las dudas que tenga. Los ejercicios solicitan el desarrollo de contenidos que se vieron en clases, por lo que estamos seguros que pueden hacerlos.

Ejercicio 1: TAD + RECURSIÓN

En las recientes elecciones legislativas de la ciudad de “Vayuno”, resultaron electos N diputados. Cada diputado participa en una sola comisión de trabajo. Cada comisión posee como máximo K diputados. Inicialmente se registran los siguientes datos: DNI, Apellido y Nombre completo y Partido (se deja en cero la comisión). Al dar de alta un nuevo diputado, se debe tener en cuenta que el diputado no esté previamente registrado. La Comisión de trabajo se asigna posteriormente y puede ser modificada.

Se solicita implementar los TADs con **módulos recursivos** que permita:

a) Cargar los N nuevos diputados teniendo en cuenta que no se repitan

b) Ordenar la lista por el método de ordenación por mezcla.

Además, asumiendo que la lista se encuentra ordenada, debe permitir realizar las siguientes tareas:

c) Eliminar un diputado

d) Asignar una comisión a un diputado teniendo en cuenta que no se debe superar el cupo k establecido para las comisiones.

Desarrolle de forma completa los archivos de cabecera y únicamente, del archivo fuente de la lista, los módulos correspondientes a los incisos b), c) y el módulo *cuentaCom()*;

Finalmente implemente un programa (no recursivo) que usando el TAD implementado se cargue la lista de los N nuevos diputados, la ordene y luego mediante un menú permita el acceso al resto de las funcionalidades.

En el TAD Diputado (diputados.c) tiene desarrollados (entre otros) los siguientes módulos:

```
int opCom(){
   int op;
   do{
         printf("1-Obras Publicas\n");
         printf("2-Presupuesto e Impuestos\n");
         printf("3-Salud Publica\n");
         printf("4-Turismo\n");
         printf("5-Deporte\n");
         printf("Opcion\n");
         scanf("%d",&op);
   } while (op<1||op>5);
   return op;
}
```

```
void mostrarCom(int c) {
   switch (c){
   case 1:
         printf("Obras Publicas\n"); break;
   case 2:
         printf("Presupuesto e Impuestos\n");break;
   case 3:
         printf("Salud Publica\n");break;
   case 4:
         printf("Turismo\n");break;
   case 5:
         printf("Deporte\n");break;
   }
}
```

A continuación se muestra una parte del TAD listaDip.c

Archivo listaDip.c

```
#include "diputado.h"
```

```
#include "listaDip.h"
#include  <stdio.h>

/*CARGAR LISTA*/
tLista nuevaLista(){
….NO SE PIDE DESARROLLO DE ESTE MODULO
}
```

/*ORDENAR*/
DESARROLLAR MODULOS PARA ORDENAR

/*ELIMINAR*/
DESARROLLAR MODULOS PARA ELIMINAR

/*ASIGNA COMISIÓN*/

```
int cuentaCom(……) {
….. DESARROLLAR ESTE MODULO
}
void asignaComDip(tLista L, int k){
        long DNI;
        int p,op;
        printf("Ingrese DNI Diputado a eliminar: ");
        scanf("%ld",&DNI);
        p=buscarDip(L.V, 1, L.N, DNI);
        if (p!=0) {
                op=opCom();
                if (cuentaCom(L.V, L.N,op)<k)
                        modiComDip(&L.V[p],op);
                else printf("Cupo cubierto para esta Comision\n");
        } else printf("El Diputado no se encuentra en la lista\n");
}
```

Ejercicio 2: LISTA ENLAZADA (Programación Dinámica – Punteros)

Una escuela organiza anualmente la fiesta de la familia en el que realizan diferentes actividades, entre ellas el juego de la lota. Todos los años cuando se realiza el juego la persona encargada del sorteo va anotando en un papel cada uno de los números obtenidos. Cuando una persona indica que tiene cartón ganador, se debe verificar que los 15 números del cartón se encuentren en lista que se anotó previamente. Esto resulta muy lento y poco práctico.

Para la edición de la fiesta de este año, el prof. de informática de la escuela quiere colaborar en la organización del juego y decide desarrollar un programa que permita facilitar la tarea de verificar si un cartón es ganador.

El prof. quiere que el programa administre de forma eficiente la memoria de la máquina, pues las PCs de la escuela poseen pocos recursos de hardware. Hace mucho que no desarrolla programas con manejo dinámico de memoria, por lo que decide pedirle ayuda a usted para el desarrollo y le manda un mail con el siguiente detalle:

El programa a desarrollar debe estar correctamente modularizado y utilizar listas enlazadas (programación dinámica). Tener en cuenta que debe permitir iniciar la lista de números sorteados en vacío y mediante un menú brindar las siguientes alternativas:

1.- Registrar un número sorteado en una lista, agregándolo de manera ordenada.

2.- Verificar que los 15 números del cartón estén en la lista de números sorteados (siempre que la lista posea al menos 15 números)

3.- Eliminar un número de la lista de números sorteados para brindar la posibilidad de subsanar errores en la carga.