Primer parcial – 26 de abril de 2024

Nombre y Apellido: Szapowalo Joan **Calificación:** 7,25 (Prom)

IMPORTANTE: NO SE CORREGIRA LO QUE NO SEA LEGIBLE: PUEDE ENTREGAR EN LAPIZ SIEMPRE QUE SEA OSCURO
Serán considerados al calificar este examen la eficiencia y legibilidad de las soluciones y el uso de las características del lenguaje C y de la programación estructurada. Para facilitar el seguimiento del código, se sugiere numerar las llaves de cada bloque, o marcar con una línea las llaves asociadas.
Para aprobar es necesario obtener al menos 5p, de los cuales al menos 4,25p deben obtenerse en el inciso i)
Para acceder al coloquio de promoción, es necesario obtener al menos 6p, de los cuales al menos 5p deben obtenerse en el inciso i)

i) Inciso a) (2.5 p)	*i) Inciso b) (3.5 p)*	*i) Inciso c) (2.5 p)*	*ii) y iii) (1.5 p)*
2,25	2	1,50	1,50

Una planta de VTV de automóviles gestiona sus turnos mediante una lista simple *LT*, donde cada nodo corresponde a un turno solicitado y contiene los siguientes datos: *dominio del vehículo* (cadena de 7, ordenado), *fecha del turno* (cadena de 8, formato AAAAMMDD), *hora* (cadena de 5, formato 24horas HH:MM)

Además, se tiene en una lista doble *PA* con los puestos de atención (no es dato la cantidad de puestos), un nodo por puesto, con la siguiente estructura: *puesto* (cadena de 3), *sublista de automóviles asignados al puesto*; para cada automóvil: *dominio*, *cantidad de observaciones encontradas*.

Una cola *C* registra los automóviles que van arribando a la planta, cada elemento de la cola es un dominio.

Se pide, resolver en lenguaje C:

i) utilizando los operadores del TDA Cola, mediante un subprograma por cada ítem resuelva:

a) Eliminar de *LT*, los turnos incorrectos. Un turno es incorrecto si el último número del *dominio* no coincide con el número del mes del turno, considerando que en octubre se atienden los dominios finalizados en 0, y en noviembre y diciembre se acepta cualquier dominio. Los dominios pueden tener el formato actual (AA999AA) o el anterior (AAA999). Finalmente, informar el porcentaje de turnos depurados (considerando el total de turnos asignados).

b) Simular la atención de los automóviles de *C* para el mes de abril, verificando que posean turno, y asignando a algún puesto aleatoriamente (utilizar la función *rand()* de *stdlib.h* de la siguiente manera: rand() % N + 1, siendo N la cantidad de puestos, dará la posición del puesto en *PA*) Cada automóvil a atender deberá insertarse al final de la sublista correspondiente. Los dominios de C que tengan turno para otro mes quedarán en la cola. Listar los dominios de los autos rechazados por no tener turno.

c) Se tiene un archivo de texto TESTEOS.TXT (no ordenado por ningún criterio) con los automóviles observados y rechazados por los puestos de atención, en cada línea la siguiente información: *Puesto, Dominio, Cantidad de observaciones, Estado ([O]bservado, [R]echazado)*. Eliminar de la sublista del puesto que corresponda, cada automóvil rechazado y completar la cantidad de observaciones para los observados.

ii) escribir el *main()* completo que realice las invocaciones a los subprogramas definidos en i) además de las operaciones de inicialización de estructuras necesarias, e invocación a funciones de carga de las mismas **(que no debe desarrollar)**.

iii) definir el tipo de la cola estática utilizado en el programa. Desarrollar **SacaC()**. Indicar en qué archivo/s iría cada definición/desarrollo.

Szapowalo Joan 1 de 3

colaest.h

```
#define PAT 8   #define MAX 100
typedef struct{
    char dominio[PAT];} Telementoc;
typedef struct{
    Telementoc datos[MAX];
    int pri, ult;} TCola;
```

colaest.c

```
#include <stdio.h>
#include "colaest.h"
void Sacac(TCola *C, Telementoc x){
    if ((*C).pri != -1){
        *x = (*C).datos[(*C).pri];
        if ((*C).pri == (*C).ult){
            (*C).pri = -1;
            (*C).ult = -1;
        }
        else
            (*C).pri++;
    }
}
```

main.c

```
#include <stdio.h>  #include <stdlib.h>  #include <string.h>
#include "Colaest.h"
#define DOM 8  #define FEC 9  #define HOR 6  #define PUE 4
// Lista simple de turnos
typedef struct nodoT{
    Char dominio[DOM], Fecha[FEC], hora[HOR];
    struct nodoT *sig;} nodoT;
typedef nodoT *TlistaT;
// Sublista de automoviles asignados
typedef struct noditoA{
    Char dominio[DOM];
    int CantObs;
    struct noditoA *sig;} noditoA;
typedef noditoA *TSub;
// Lista doble de puestos
typedef struct nodoP{
    Char Puesto[PUE];
    TSub SubAutos;
    struct nodoP *sig, *ant;} nodoP;
typedef nodoP *PnodoP;
typedef struct{
    PnodoP pri, ult;} TlistaP;
```

Szapowalo Joan · 2 h 3

total -> 100%
depur -> porc

a) int Escorrecto (Char dominio[], Char fecha[]){ Bien modularizado

```
  Char mes[3], ultnum;
  mes[0] = fecha[4]; mes[1] = fecha[5]; mes es innecesario, podrías usar fecha directamente.
  int longpat;
  longpat = Strlen(dominio);
  if (longpat == 6)
    ultnum = domnio[5];
  else
    ultnum = dominio[4]; Condición redundante
  if ( mes[0] != 1 || (mes[0] == 1 && mes[1] == 0)
    if ( mes[0] == 0)
      if ( mes[1] == ultnum)
        return 1;
      else
        return 0;
    else
      if (ultnum == 0)
        return 1;
      else
        return 0;
  else
    return 1;
}
```

return cond

return cond

return mes[1] == ultnum;

```
void EliminaTurnos (TListaT *LT){
  TListaT ant, act, elim; int depurados=0, total=0;
  act = (*LT); float porcentaje;
  While (act != NULL){
    If (!Escorrecto(act->dominio, act->fecha)){
      elim = act;
      if (act == *LT)
        *LT = act->sig;
      else
        ant->sig = act;
      act = act->sig;
      free(elim);
      depurados += 1;
    }else{
      ant = act;
      act = act->sig;
    }
    total += 1;
  }
  porcentaje = (total)? (float) depurados.100/total : 0;
  if (Porcentaje)
    Printf ("\n Porcentaje de turnos depurados: %.2f", Porcentaje)
  else
    Printf ("\n No se han depurado turnos");
}
```

N no es dato del problema (en este caso abril)

b) void AtiendeC(TlistaT LT, int mpuesto, char messol[], TCola *C, TlistaP LP){
TelementoC reg, centinela; int tieneturno;
Char mes[3]; TlistaT aux;
Centinela.dominio = "zzzzzz"; Asegura colocarlo directamente
PoneC(C, Centinela);
SacaC(C, &reg);
while(strcmp(reg.dominio, centinela.dominio)){
aux = LT; tieneturno = 0;
while(aux!=NULL && !tieneturno && strcmp(reg.dominio, aux->dominio > 0))
if(strcmp(aux->dominio, reg.dominio) == 0){ → No puede darse jamás esta condición por la del ciclo
mes[0] = aux->fecha[4];
mes[1] = aux->fecha[5];
if(strcmp(mes, messol) == 0) // el coche tiene turno para el mes solicitado
tieneturno = 1;
else
aux = aux->sig; ¿Más de un turno para la VTV?
}
else
aux = aux->sig; //sigo buscando
if(tieneturno)
asignaPuesto(LP, mpuestos, reg);
else{
PoneC(C, reg); Solo vuelven a la cola dominios con turno para otro mes
Printf("Sin turno: %s \n", reg.dominio);
} Solo debería mostrarse dominios sin turno alguno
SacaC(C, &reg);
}
}

void AsignaPuesto(TlistaP LP, int mpuestos, TelementoC reg){

```
}
}
void AsignaPuesto(TlistaP LP, int npuestos, TelementoC reg){
    int PuestoAsig, nuevoIzq, PuestoMitad, numPuesto;
    TSub ant, act, nuevoA;
    PnodoP PuestoAct;
    PuestoAsig = rand()% npuestos + 1;
    PuestoMitad = npuestos/2;
    if (PuestoAsig <= PuestoMitad){
        PuestoAct = LP.pri;
        nuevoIzq = 1; numPuesto = 1;
    }
    else{
        PuestoAct = LP.ult;
        nuevoIzq = 0; numPuesto = npuestos;
    }
    if (nuevoIzq)  En vez de este if, podría haber metido los actos en las
    {  while (numPuesto != PuestoAsig){  menos anteriores
           numPuesto++
           PuestoAct = PuestoAct->sig;
       }
    }
    else
       while (numPuesto != PuestoAsig){
           numPuesto--;
           PuestoAct = PuestoAct->ant;
       }
    {  nuevoA = (TSub) malloc(sizeof(noditoA));
       strcpy (nuevoA->dominio, reg.dominio);
       nuevoA -> cantObs = 0;
       nuevo -> sig = NULL;
       acts = PuestoAct -> subAutos;
       while (acts!=NULL){          // inserción al final de la sublista
           ants = acts;
           acts = acts->sig;
       }
       if (PuestoAct -> sub == NULL)
           PuestoAct -> sub = nuevoA;
       else
           ants -> sig = nuevo;
    }
}
```

En la misma línea

HOJA Nº

FECHA

c) void ProcesaArchivo(TlistaP LP){
FILE *arch; TlistaP Puestoact; Tsub ants, acts;
Char Puesto[PUE], dominio[DOM], estado;
int CantObs;
arch = fopen("TESTEOS.TXT", "rt");
If (arch == NULL)
printf("No se pudo abrir el archivo");
else{
while(fscanf(arch, "%s %s %d %c", puesto, dominio, &CantObs, &estado)==4)
// descarto que el puesto no este → If (strcmp(LP.pri->puesto, puesto)<=0) && strcmp(LP.ult->puesto, puesto)>=0){
puestoact = LP.pri; (cabeza auxiliar NULL)
while (strcmp(puestoact->puesto, puesto)<0)
puestoact = puestoact->sig;
If (strcmp(puestoact->puesto, puesto)==0){ // encontre el puesto
acts = puestoact->subAutos;
while (acts!=NULL && strcmp(acts->dominio, dominio)!=0){
ants = act;
acts = acts->sig;
}
if (acts!=NULL) // encontre el dominio
if (estado=='R'){ // elimino
ants->sig = acts->sig; // No contempla eliminacion por cabeza
free(act);
}
else // completo observaciones
acts->CantObs = CantObs;
+=
}
} No cierra el archivo
}
}

No hice Switch! No esta ordenada x puesto

No pude Comentarios al pie de la linea

```
int main(){
   TListaP LP; // lista puestos      int npuestos,
   TListaT LT; // lista turnos       char mes[3];
   TCola Cautos; // cola de autos;
   LP.pri = NULL;
   LP.ult = NULL;
   LT = NULL;
   IniciaC(&Cautos);
    :
   CargaPuestos(&LP, &npuestos); // devuelve el num de puestos d carga
   CargaTurnos(&LT);
   CargaAutos(&Cautos);
    :
a) EliminaTurnos(&LT);
b) printf("Indique el mes a procesar");
   scanf("%s", mes); // abril(04)
   AtiendeC(LT, npuestos, mes, &C, LP);
c) ProcesaArchivo(LP);
   return 0;
}
```

NO

La simulación no tiene N como dato según enunciado