Programación II. Parcial 1. 12 de Marzo 2019

Apellidos: _____Nombre:_____

Ejercicio 1. (3 puntos) Una empresa de telefonía desea reformar su aplicación (App) que gestiona los contactos que se almacenan en sus terminales móviles. Nuestro cometido es diseñar los TADs que serán utilizados en la nueva propuesta, la cual tiene los siguientes requisitos:

- La nueva App podrá gestionar, para el usuario de la misma, sus diferentes agendas (se permitirá un máximo de 100 agendas). Por ejemplo: agenda con los contactos personales, agenda con los contactos del trabajo, agenda con los contactos del club de ajedrez, etc. Por lo tanto, la App tendrá los datos del propietario del móvil y el conjunto de sus agendas.
- Cada agenda tiene un nombre ("mi trabajo", "personal", "club de ajedrez", etc.) y está compuesta por contactos (con un máximo de 10.000 contactos por agenda). De cada contacto se desea almacenar su nombre, sus apellidos y los diferentes medios disponibles para que sea contactado.
- Los posibles medios para contactar a un contacto son: direcciones postales, números de teléfonos y direcciones de correo electrónico. Para cada uno de estos medios, además de almacenar su valor ("911234567", "nombre.apellido@uam.es", etc.), se indicará si este medio es personal o de trabajo. Se permitirá guardar un máximo de 20 medios por cada contacto.

a) (0,2 p) ¿Cuántos Tipos Abstractos de Datos necesarios son necesarios para representar lo descrito anteriormente y qué nombre darías a cada uno de ellos? Los tipos Status y Bool se encuentran ya definidos (no se consideran aquí)

b) (0,8 p) Escribe el código en C de las Estructuras de Datos (EdD) necesarias para implementar los TADs que has respondido en la pregunta anterior, y di en qué ficheros se escribirían esos códigos.

Se cuenta con las siguientes definiciones:

```
typedef enum {ERROR=0, OK=1} Status;

typedef enum {FALSE=0, TRUE=1} Bool;

typedef enum {PERSONAL=1, TRABAJO=2} Tipo;

#define MAX_AGENDAS 100

#define MAX_CONTACTOS 10000

#define MAX_MEDIOS 20

#define MAX_TXT 124
```

c) (0,6 p) Dados los siguientes prototipos, escribe en C el código, sin control de errores, de estas funciones:

- Medio *medio_ini(char *dato, Tipo tipo); /* crea un medio para contactar a un contacto a partir de un dato y su tipología/tipo */
- Status contacto_add_medio (Contacto *pContacto, Medio *pMedio); /*agrega un medio de contactar a un contacto dado*/
- void app_free (App *pApp); /*libera toda la memoria ocupada por una App, liberando todos sus componentes */

d) (1,4 p) Implementa en C y sin control de errores una función principal main que realice las siguientes acciones:

- Crea un contacto con estos datos:
    i. Ana Perez Lopez
    ii. Teléfono trabajo: 916788761
    iii. Email personal: ana.pl@person.es
- Introduce dicho contacto en una agenda personal del propietario de la App del móvil que tiene estos datos:
    i. Juan Gil Mayor
    ii. Telefono personal: 676767676
    iii. Email trabajo: juan.gm@empresa.es
- Imprime por pantalla los contactos de las agendas del propietario de la App del móvil antes indicado.

Después de imprimir, ya no se va a trabajar más con las agendas de la App del móvil.
Gestiona adecuadamente la memoria utilizada.

Para la realización de este código se cuenta con los prototipos de las siguientes funciones:

```
Medio *medio_ini(char *dato, Tipo tipo);

Contacto *contacto_ini(char *nombre, char *apellidos);

Status contacto_add_medio (Contacto *pContacto, Medio *pMedio);

Agenda *agenda_ini (char *tipo);

Status agenda_add_contacto (Agenda *pAgenda, Contacto *pContacto);

App *app_ini (Contacto *pContacto);

Status app_add_agenda (App *pApp, Agenda *pAgenda);

int medio_print (FILE *pf, Medio *pMedio);

int contacto_print (FILE *pf, Contacto *pContacto);

int agenda_print (FILE *pf, Agenda *pAgenda);

int app_print (FILE *pf, App *pApp);

void medio_free (Medio *pMedio);

void contacto_free (Contacto *pContacto);

void agenda_free (Agenda *pAgenda);

void app_free (App *pApp);
```

SOLUCION

a)

App

Agenda

Contacto

Medio

b)

```
//medio.c
struct _Medio
{
   char dato[MAX_TXT];
   Tipo tipo;
};
//contacto.c
struct _Contacto
{
   char nombre[MAX_TXT];
   char apellidos[MAX_TXT];
   Medio *medios[MAX_MEDIOS];
   int numMedios;
};
//agenda.c
struct _Agenda
{
   char tipo[MAX_TXT];
   Contacto *contactos[MAX_CONTACTOS];
   int numContactos;
};
//app.c
struct _App
{
```

```
    Contacto *propietario;
    Agenda *agendas[MAX_AGENDAS];
    int numAgendas;
};
```

c)

```
Medio *medio_ini(char *dato, Tipo tipo)
{
            Medio *pMedio;

        pMedio = (Medio *) malloc (sizeof(Medio));
        pMedio->tipo = tipo;
        strcpy (pMedio->dato, dato);
        return pMedio;
}
Status contacto_add_medio (Contacto *pContacto, Medio *pMedio)
{
    pContacto->medios[pContacto->numMedios] = pMedio;
    pContacto->numMedios++;
    return OK;
}
void app_free (App *pApp)
{
    int i;
    for (i=0; i < pApp->numAgendas;i++)
        agenda_free (pApp->agendas[i]);
    contacto_free (pApp->propietario);
    free (pApp);
}
```

d)

```
int main(int argc, char** argv) {
    App *pApp;
    Agenda *pAgenda;
    Contacto *pContacto1, *pContacto2;
    Medio *pMedio1,*pMedio2,*pMedio3, *pMedio4;
    Status temp;

    pMedio1 = medio_ini ("916788761", TRABAJO);
    pMedio2 = medio_ini ("ana.pl@person.es", PERSONAL);
    pContacto1 = contacto_ini("Ana", "Perez Lopez");
    contacto_add_medio (pContacto1, pMedio1);
    contacto_add_medio (pContacto1, pMedio2);

    pMedio3 = medio_ini ("676767676", PERSONAL);
    pMedio4 = medio_ini ("juan.gm@empresa.es ", TRABAJO);
    pContacto2 = contacto_ini("Juan", "Gil Mayor");
    contacto_add_medio (pContacto2, pMedio3);
    contacto_add_medio (pContacto2, pMedio4);

    pAgenda = agenda_ini ("agenda_personal");
    temp =  agenda_add_contacto (pAgenda, pContacto1)    ;

    pApp = app_ini (pContacto2);
    app_add_agenda (pApp, pAgenda);

    app_print (stdout,pApp);

    app_free (pApp);
    return 1;
}
```