LABORATORIO 1

ALUMNO: GALVEZ TOBIAS

PROFESOR/A: SCARAFILO GERMAN

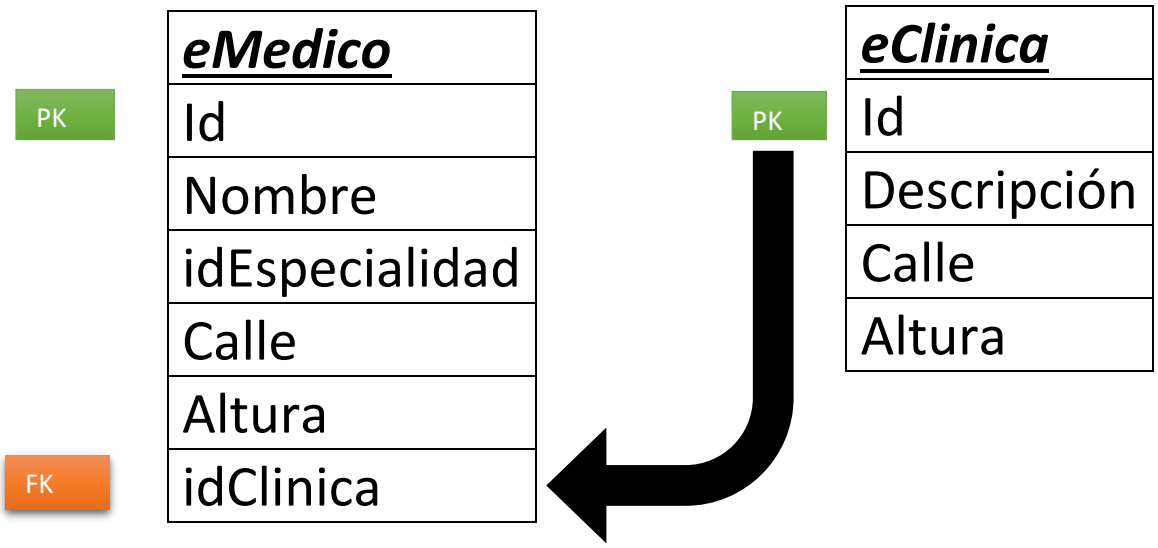
CURSO: 1° C

AÑO: 2022

B) La estructura agregada representa el lugar en el que el médico ejerce su trabajo:

Tengo a Franco González, Juan Paz y Andrés Méndez. Los Franco y Juan trabajan en la clínica “A” , mientras que Andrés en la clínica “B”

Esta nueva estructura “eClinica” la relaciono con la estructura de “eMedico” a través de un id.
El diagrama entidad relación (DER) sería el siguiente:



PK: Primary Key (Clave primaria)

FK: Foreign Key (Clave foránea)

C)

```
/**
 * Funcion que permite averiguar el medico mas cercano de una clinica determinada
 * @param lista
 * @param tam
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
 * @param especialidades
 * @param tamEspecialidades
 * @param clinicas
 * @param tamClinicas
 * @return 1 si tuvo exito, 0 si no
 */
```

```
int medicoMasCercanoAClinica(eConsulta lista[],int tam,eMedico
medicos[],int tamMedicos, eDiagnostico diagnosticos[], int
tamDiagnosticos, eEspecialidad especialidades[], int
tamEspecialidades, eClinica clinicas[], int tamClinicas)
{
}
```

D)

```
#ifndef CONSULTA_H_
#define CONSULTA_H_

#include "fecha.h"
#include "especialidad.h"
#include "clinica.h"



typedef struct
{
     int idConsulta;
     char nombrePaciente[50];
     eFecha fecha;
     int idDiagnostico;
     int idMedico;
     int estado;
}eConsulta;



/**
 * Funcion que inicializa todos los estados de consultas en
"VACIO"
 * @param lista
 * @param tam
 * @return 1 si tuvo exito, y 0 si no.
 */
int inicializarConsulta(eConsulta lista[], int tam);
/**
 * Funcion que permite buscar si hay espacio libre para dar de
alta una consulta
 * @param lista
 * @param tam
 * @return 1 si tuvo exito, y 0 si no.
```

```
 */
int buscarLibre(eConsulta lista[], int tam);
/**
 * Funcion que permite buscar una consulta por id
 * @param lista
 * @param tam
 * @param id
 * @return 1 si tuvo exito, y 0 si no.
 */
int buscarConsultaId(eConsulta lista[], int tam, int id);
/**
 * Funcion que permite mostrar una consulta
 * @param unaConsulta
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
 * @param especialidades
 * @param tamEspecialidades
 */
void mostrarConsulta(eConsulta unaConsulta, eMedico unMedico,
eMedico medicos[], int tamMedicos, eDiagnostico diagnosticos[],
int tamDiagnosticos, eEspecialidad especialidades[], int
tamEspecialidades);
/**
 * Funcion que permite mostrar una lista de consultas
 * @param lista
 * @param tam
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
 * @param especialidades
 * @param tamEspecialidades
 * @return 1 si tuvo exito, 0 si no
 */
int mostrarConsultas(eConsulta lista[], int tam, eMedico
medicos[], int tamMedicos, eDiagnostico diagnosticos[], int
tamDiagnosticos, eEspecialidad especialidades[], int
tamEspecialidades);

/**
 * Funcion que permite dar de alta una consulta
 * @param lista
 * @param tam
 * @param pId
 * @param medicos
 * @param tamMedicos
 * @return 1 si tuvo exito, 0 si no
```

```
 */
int altaConsulta(eConsulta lista[], int tam, int* pId, eMedico
medicos[], int tamMedicos);

/**
 * Funcion que permite modificar datos de consulta
 * @param lista
 * @param tam
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
 * @param especialidades
 * @param tamEspecialidades
 * @return 1 si tuvo exito, 0 si no
 */
int modificarConsulta(eConsulta lista[], int tam, eMedico
medicos[], int tamMedicos, eDiagnostico diagnosticos[], int
tamDiagnosticos, eEspecialidad especialidades[], int
tamEspecialidades);

/**
 * Funcion que permite dar de baja una consulta
 * @param lista
 * @param tam
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
 * @param especialidades
 * @param tamEspecialidades
 * @return 1 si tuvo exito, 0 si no
 */
int bajaConsulta(eConsulta lista[], int tam, eMedico medicos[],
int tamMedicos, eDiagnostico diagnosticos[], int
tamDiagnosticos, eEspecialidad especialidades[], int
tamEspecialidades);
/**
 * Funcion que permite diagnosticar la consulta de un paciente
 * @param lista
 * @param tam
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
 * @param especialidades
 * @param tamEspecialidades
 * @return 1 si tuvo exito, 0 si no
 */
```

```
int diagnosticarConsulta(eConsulta lista[], int tam, eMedico
medicos[], int tamMedicos, eDiagnostico diagnosticos[], int
tamDiagnosticos, eEspecialidad especialidades[], int
tamEspecialidades);

#endif /* CONSULTA_H_ */
```

```
#ifndef MEDICO_H_
#define MEDICO_H_


typedef struct
{
     int idMedico;
     char nombre[50];
     int idEspecialidad;
     char calle[50];
     int altura;
     int idClinica;
}eMedico;

/**
 * Funcion que permite listar todos los medicos
 * @param lista
 * @param tam
 * @return 1 si tuvo exito, 0 si no
 */
int listarMedicos(eMedico lista[], int tam);
/**
 * Funcion que permite cargar el nombre de medico
 * @param medicos
 * @param tam
 * @param idMedico
 * @param nombre
 * @return 1 si tuvo exito, 0 si no
 */
int cargarNombreMedico(eMedico medicos[], int tam, int idMedico
, char nombre[]);
/**
 * Funcion que permite buscar un medico por id
 * @param lista
 * @param tam
 * @param id
 * @return si encontró el id, el id ingresado, si no -1
 */
```

```
int buscarMedicoId(eMedico lista[], int tam, int id);


#endif /* MEDICO_H_ */
```

```
#ifndef DIAGNOSTICO_H_
#define DIAGNOSTICO_H_

typedef struct
{
     int id;
     char descripcion[20];
}eDiagnostico;

/**
 * Funcion que permite listar todos los diagnosticos
 * @param lista
 * @param tam
 * @return 1 si tuvo exito, 0 si no
 */
int listarDiagnosticos(eDiagnostico lista[], int tam);
/**
 * Funcion que permite cargar la descripcion del diagnostico
 * @param diagnosticos
 * @param tam
 * @param idDiagnostico
 * @param descripcion
 * @return 1 si tuvo exito, 0 si no
 */
int cargarDescripcionDiagnostico(eDiagnostico diagnosticos[],
int tam, int idDiagnostico, char descripcion[]);

#endif /* DIAGNOSTICO_H_ */
```

```c
#ifndef ESPECIALIDAD_H_
#define ESPECIALIDAD_H_

typedef struct
{
     int id;
     char descripcion[20];
}eEspecialidad;


/**
 * Funcion que permite listar todas las especialidades
 * @param lista
 * @param tam
 * @return 1 si tuvo exito, 0 si no
 */
int listarEspecialidades(eEspecialidad lista[], int tam);
/**
 * Funcion que permite cargar el nombre de especialidad
 * @param especialidades
 * @param tam
 * @param idEspecialidad
 * @param especialidad
 * @return 1 si tuvo exito, 0 si no
 */
int cargarEspecialidad(eEspecialidad especialidades[], int tam,
int idEspecialidad , char especialidad[]);

#endif /* ESPECIALIDAD_H_ */
```

```c
#ifndef CLINICA_H_
#define CLINICA_H_

typedef struct
{
     int id;
     char descripcion[50];
     char calle[50];
     int altura;
}eClinica;

/**
 * Funcion que permite listar todas las clinicas
 * @param lista
```

```
 * @param tam
 * @return 1 si tuvo exito, 0 si no
 */
int listarClinicas(eClinica lista[], int tam);
/**
 * Funcion que permite buscar una clinica por id
 * @param lista
 * @param tam
 * @param id
 * @return si encontró el id, el id ingresado, si no -1
 */
int buscarClinicaId(eClinica lista[], int tam, int id);
#endif /* CLINICA_H_ */



#ifndef MENUS_H_
#define MENUS_H_

/**
 * Funcion que permite mostrar el men<U+FFFD> de consultas
 * @return opcion ingresada
 */
int menu();
/**
 * Funcion que permite mostrar el submenu de consultas,
perteneciente al menu de las listas
 * @return opcion ingresada
 */
int subMenu();
/**
 * Funcion que permite acceder al menu de modificacion
 * @return opcion ingresada
 */
int menuModificacion();

#endif /* MENUS_H_ */
```

```
#ifndef LISTAR_H_
#define LISTAR_H_

/**
 * Funcion que permite mostrar consultas diagnosticadas por id
de medico
 * @param lista
 * @param tam
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
 * @param idMedico
 * @param especialidades
 * @param tamEspecialidades
 * @return 1 si tuvo exito, 0 si no
 */
int mostrarConsultasIdMedicoDiagnosticado(eConsulta lista[],int
tam,eMedico medicos[],int tamMedicos,eDiagnostico
diagnosticos[], int tamDiagnosticos, int idMedico, eEspecialidad
especialidades[], int tamEspecialidades);

/**
 * Funcion que permite listar todos los m□dicos con consultas
diagnosticadas
 * @param lista
 * @param tam
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
 * @param especialidades
 * @param tamEspecialidades
 * @return 1 si tuvo exito, 0 si no
 */
int listaMedicosConsultasDiagnosticadas(eConsulta lista[], int
tam, eMedico medicos[], int tamMedicos, eDiagnostico
diagnosticos[], int tamDiagnosticos, eEspecialidad
especialidades[], int tamEspecialidades);
/**
 * Funcion que permite un listado de consultas ordenadas por
fecha, desde la m□s reciente
 * hasta la m□s antigua
 * @param lista
 * @param tam
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
```

```
 * @param especialidades
 * @param tamEspecialidades
 * @return 1 si tuvo exito, 0 si no
 */
int listaConsultasOrdenadasFecha(eConsulta lista[], int tam,
eMedico medicos[], int tamMedicos, eDiagnostico diagnosticos[],
int tamDiagnosticos, eEspecialidad especialidades[], int
tamEspecialidades);
/**
 * Funcion que permite un listado de consultas que ya fueron
diagnosticadas.
 * @param lista
 * @param tam
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
 * @param especialidades
 * @param tamEspecialidades
 * @return 1 si tuvo exito, 0 si no
 */
int listaConsultasDiagnosticadas(eConsulta lista[], int tam,
eMedico medicos[], int tamMedicos, eDiagnostico diagnosticos[],
int tamDiagnosticos, eEspecialidad especialidades[], int
tamEspecialidades);
/**
 * Funcion que permite un Listado de las consultas
diagnosticadas con covid-19 desde el
 * inicio de la pandemia hasta la actualidad
 * @param lista
 * @param tam
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
 * @param especialidades
 * @param tamEspecialidades
 * @return 1 si tuvo exito, 0 si no
 */
int listaConsultasDiagnosticadasCovid19(eConsulta lista[], int
tam, eMedico medicos[], int tamMedicos, eDiagnostico
diagnosticos[], int tamDiagnosticos, eEspecialidad
especialidades[], int tamEspecialidades);

/**
 * Funcion que permite mostrar un listado de consultas ordenadas
por especialidad alfabéticamente
 * @param lista
 * @param tam
```

```
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
 * @param especialidades
 * @param tamEspecialidades
 * @return 1 si tuvo exito, 0 si no
 */
int listaConsultasPorEspecialidadOrdenadas(eConsulta lista[],int
tam,eMedico medicos[],int tamMedicos, eDiagnostico
diagnosticos[], int tamDiagnosticos, eEspecialidad
especialidades[], int tamEspecialidades );



/**
 * Funcion que permite mostrar consultar por id de medico con
consultas entre abril y julio
 * @param lista
 * @param tam
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
 * @param idMedico
 * @param especialidades
 * @param tamEspecialidades
 * @return 1 si tuvo exito, 0 si no
 */
int mostrarConsultasIdPorEspecialidadAbrilJulio(eConsulta
lista[],int tam,eMedico medicos[],int tamMedicos,eDiagnostico
diagnosticos[], int tamDiagnosticos, int idEspecialidad, int
idMedico, eEspecialidad especialidades[], int
tamEspecialidades);



/**
 * Funcion que permite Listado de todas las consultas entre el
mes de abril y julio para una especialidad determinada.
 * @param lista
 * @param tam
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
 * @param especialidades
 * @param tamEspecialidades
 * @return 1 si tuvo exito, 0 si no
```

```
 */
int listaConsultasPorEspecialidad(eConsulta lista[],int
tam,eMedico medicos[],int tamMedicos, eDiagnostico
diagnosticos[], int tamDiagnosticos, eEspecialidad
especialidades[], int tamEspecialidades);


/**
 * Funcion que permite calcular el porcentaje de consultas
 * diagnosticadas que atiende cada m�dico en funci�n del total
de diagn�sticos
 * @param lista
 * @param tam
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
 * @return 1 si tuvo exito, 0 si no
 */
int porcentajeConsultasDiagnosticadas(eConsulta lista[],int
tam,eMedico medicos[],int tamMedicos, eDiagnostico
diagnosticos[], int tamDiagnosticos);

/**
 * Funcion que permite mostrar la/las especialidad/es mas
estudiada/s
 * @param lista
 * @param tam
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
 * @param especialidades
 * @param tamEspecialidades
 * @return 1 si tuvo exito, 0 si no
 */
int especialidadMasEstudiada(eConsulta lista[],int tam,eMedico
medicos[],int tamMedicos, eDiagnostico diagnosticos[], int
tamDiagnosticos, eEspecialidad especialidades[], int
tamEspecialidades );

/**
 * Funcion que permite mostrar la/las enfermedad/es menos
diagnosticada/s
 * @param lista
 * @param tam
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
```

```
 * @param tamDiagnosticos
 * @param especialidades
 * @param tamEspecialidades
 * @return 1 si tuvo exito, 0 si no
 */
int enfermedadMenosDiagnosticada(eConsulta lista[],int
tam,eMedico medicos[],int tamMedicos, eDiagnostico
diagnosticos[], int tamDiagnosticos, eEspecialidad
especialidades[], int tamEspecialidades );


/**
 * Funcion que permite averiguar el medico mas cercano de una
clinica determinada
 * @param lista
 * @param tam
 * @param medicos
 * @param tamMedicos
 * @param diagnosticos
 * @param tamDiagnosticos
 * @param especialidades
 * @param tamEspecialidades
 * @param clinicas
 * @param tamClinicas
 * @return 1 si tuvo exito, 0 si no
 */
int medicoMasCercanoAClinica(eConsulta lista[],int tam,eMedico
medicos[],int tamMedicos, eDiagnostico diagnosticos[], int
tamDiagnosticos, eEspecialidad especialidades[], int
tamEspecialidades, eClinica clinicas[], int tamClinicas );
#endif /* LISTAR_H_ */
```

E)

https://drive.google.com/drive/folders/1c9cNGb0HQu73RXcif5NYYV7o3Ul5S0vo?usp=sharing