Una Relojería busca sistematizar los pasos que sigue en la atención de reparaciones. En resumen, dichos pasos son los siguientes:

- El cliente solicita una reparación. El personal de atención toma sus datos y los del problema a reparar.
- El técnico revisa el reloj, registra el arreglo a realizar y su precio.
- El personal de atención comunica por teléfono al cliente lo que informó el técnico. El cliente decide que se haga o no el arreglo, y se registra esta decisión.
- Si el cliente decidió que no se haga el arreglo, queda listo para entregar. Caso contrario, el técnico lo repara y al terminarlo lo registra.
- El personal de atención comunica por teléfono al cliente que la reparación está lista.
- El cliente pasa por la relojería, retira el reloj y si pidió que se repare paga el precio correspondiente.

A continuación se describen los datos que debe manejar el sistema, y su funcionalidad:

✓ Hasta 500 clientes. De cada una de ellos se almacenan los siguientes datos:
  - Número de Documento (numérico sin decimales, entre 1 y 99.999.999)
  - Nombre (texto)
  - Teléfono (texto)

✓ Hasta 3.000 pedidos de reparación almacenando para cada una:
  - Número de Reparación (numérico entre 1 y 40.000, porque periódicamente se quitan del sistema las reparaciones entregadas y se archivan)
  - Número de Documento del Cliente que la solicitó
  - Descripción del Problema (texto)
  - Descripción de la Solución (texto)
  - Precio (numérico con decimales)
  - Estado: indica si la reparación se encuentra iniciada (I), presupuestada (P), para reparar (R), terminada, avisada para entregar (A) o entregada (E)

Se pide construir parte del módulo de atención al cliente, de acuerdo a las siguientes especificaciones:

1. Al iniciarse, el sistema cargará los datos necesarios para su funcionamiento diario, mediante el procedimiento CargaInicial (NO DESARROLLAR – COMPLETAR PARÁMETROS).
2. Luego presentará un menú con las siguientes opciones: “Ingresar nuevo cliente”, “Ingresar Nueva Reparación”, “Comunicar Precio Presupuestado”, “Entregar”, y “Salir”. Las funcionalidades a construir se describen a continuación:
3. **INGRESAR NUEVO CLIENTE:**
   a. Al ingresar en esta opción, el sistema verificará que exista espacio para cargar un nuevo cliente; en caso contrario presentará un cartel informando del error y volverá al Menú.
   b. Si hay lugar, el sistema pedirá el documento del nuevo cliente. Luego verificará que el número de documento no haya sido utilizado en un cliente anterior; en caso contrario presentará un mensaje informando el error y volverá al menú.
   c. De satisfacer las validaciones anteriores, el sistema pedirá nombre y teléfono, almacenando los 3 datos.
4. **INGRESAR NUEVA REPARACION:** (NO DESARROLLAR – COMPLETAR PARÁMETROS).
5. **COMUNICAR PRECIO PRESUPUESTADO:** (NO DESARROLLAR – COMPLETAR PARÁMETROS).
6. **ENTREGAR REPARACION:**
   a. El sistema solicitará el número de documento del cliente. Validará que exista y volverá a solicitarlo en caso contrario.
   b. Luego preparará una lista con las reparaciones que están avisadas para entregar, que correspondan al cliente. Si ninguna cumple con estas condiciones, informa al usuario y vuelve al menú principal.
   c. Si existen reparaciones, las muestra en pantalla solicitando el número de reparación a entregar:
   d. Si el número de reparación no existe, el sistema informa la situación al usuario y vuelve al menú principal.
   e. Si el número de reparación no pertenece al cliente o no está avisada para entregar, el sistema informa la situación al usuario y vuelve al menú principal.
   f. Cumplidas las validaciones mencionadas, el sistema pedirá confirmación para registrar la entrega (“S” o “N”), mientras muestra el nombre del cliente, número de reparación y precio. Si la confirma, cambiará el estado de la reparación a entregada e informará que se registró la entrega. Si el usuario decide no confirmarla, informará que no se registró la entrega.
7. **SALIR:** Al ingresar en esta opción, la aplicación guardará los conjuntos de datos que pudieron cambiar durante la ejecución del programa mediante el procedimiento GuardadoFinal (NO DESARROLLAR – COMPLETAR PARAMETROS) y se cerrará.

Se pide:

a) **Completar** los parámetros de las llamadas a los procedimientos comprendidos en CommandButton1_Click()

b) **Desarrollar** los procedimientos **IngresarCli y Entregar**, y todas las funciones y procedimientos que considere necesarias para los mismos.

Aclaraciones:

- ✓ Todos los ingresos de datos deben de ser validados (de acuerdo a los valores máximos y mínimos definidos en cada caso, y en ningún caso se admiten valores vacíos).
- ✓ Para todos los valores inválidos, se debe pedir al usuario que vuelva a ingresar el mismo, salvo en los casos donde se especifique otra acción (Ejemplo: volver al Menú…)
- ✓ La no utilización de las funcionalidades previamente desarrolladas será considerada como error.

CÓDIGO DESARROLLADO HASTA EL MOMENTO:

```
using System;

public class Program
{

      public static void Main (string[] args) {

            int[] ClientesDocumento = new int[500];
            string[] ClientesNombre = new string[500];
            string[] ClientesTelefono = new string[500];

          int[] ReparacionesNumero = new int[3000];
      int[] ReparacionesDocCliente = new int[3000];
      string[] ReparacionesProblema = new string[3000];
      string[] ReparacionesSolucion = new string[3000];
      int[] ReparacionesPrecio = new int[3000];
      string[] ReparacionesEstado = new string[3000];

            string opcion = "";

      do {
                  opcion = strMenu();
                  switch (opcion) {
                        case kstrOpcNuevoCli:

      IngresarCli(_____________________________________________________);
                              break;
                        case kstrOpcIngresar:

      IngresarRep(_____________________________________________________);
                              break;
                        case kstrOpcComunPre:

      ComunicarPrecio(_____________________________________________________);
                              break;
                        case kstrOpcAvisarTe:

      AvisarTerminacion(_____________________________________________________);
                              break;
                        case kstrOpcEntregar:

      Entregar(_____________________________________________________);
                              break;
                  }
            } while (opcion != "S");
            GuardadoFinal(_____________________________________________________);

      }
```

```
      private static int iBuscarRepar(int[] ReparacionesNumero, int numeroRep)
{
      //Busca una reparación por su número; devuelve 0 si no la encuentra.
      }
      private static string strMenu() {
      //Pide el ingreso de una opción por parte del usuario, mientras muestra
las opciones
      }
      private static string strPedirString(string strCartel) {
      //Pide un string, validando que no sea vacio
      }
      private static void strPedirSoN(ByVal strCartel {) {
      //Pide un string, validando que sea S o N
      }
      private static int iPedirNro(string strCartel, int lMin, int lMax {
      //Pide un integer, validando que sea mayor o igual a lMin y menor o igual
a lMax
      }



}
```

```
Option Explicit
Option Base 1

Const kiMaxReps As Integer = 3000
Const kiMaxClis As Integer = 500
Const klMaxNroDoc As Long = 99999999
Const klMaxNroRep As Long = 40000

Const kstrEstIniciado As String = "I"
Const kstrEstPresupue As String = "P"
Const kstrEstReparar As String = "R"
Const kstrEstTerminad As String = "T"
Const kstrEstAvisado As String = "A"
Const kstrEstEntregad As String = "E"

Const kstrOpcNuevoCli As String = "N"
Const kstrOpcIngresar As String = "I"
Const kstrOpcComunPre As String = "C"
Const kstrOpcAvisarTe As String = "A"
Const kstrOpcEntregar As String = "E"
Const kstrOpcSalir As String = "S"
```

```
Private Sub CommandButton1_Click()
```