# Prueba de Caja Blanca

## *"Sistema computarizado de registro de ingreso y salida vehicular y peatonal para el conjunto habitacional Armenia Etapa II"*

**Integrantes:**
Kerly Andreina Chuqui Aguinda
Denis Alexander Ullcu Ullco

**Fecha 2024-09-02**

**Prueba caja blanca de Requisito 1 :** Registro de ingreso para usuarios para el conjunto habitacional Armenia II.

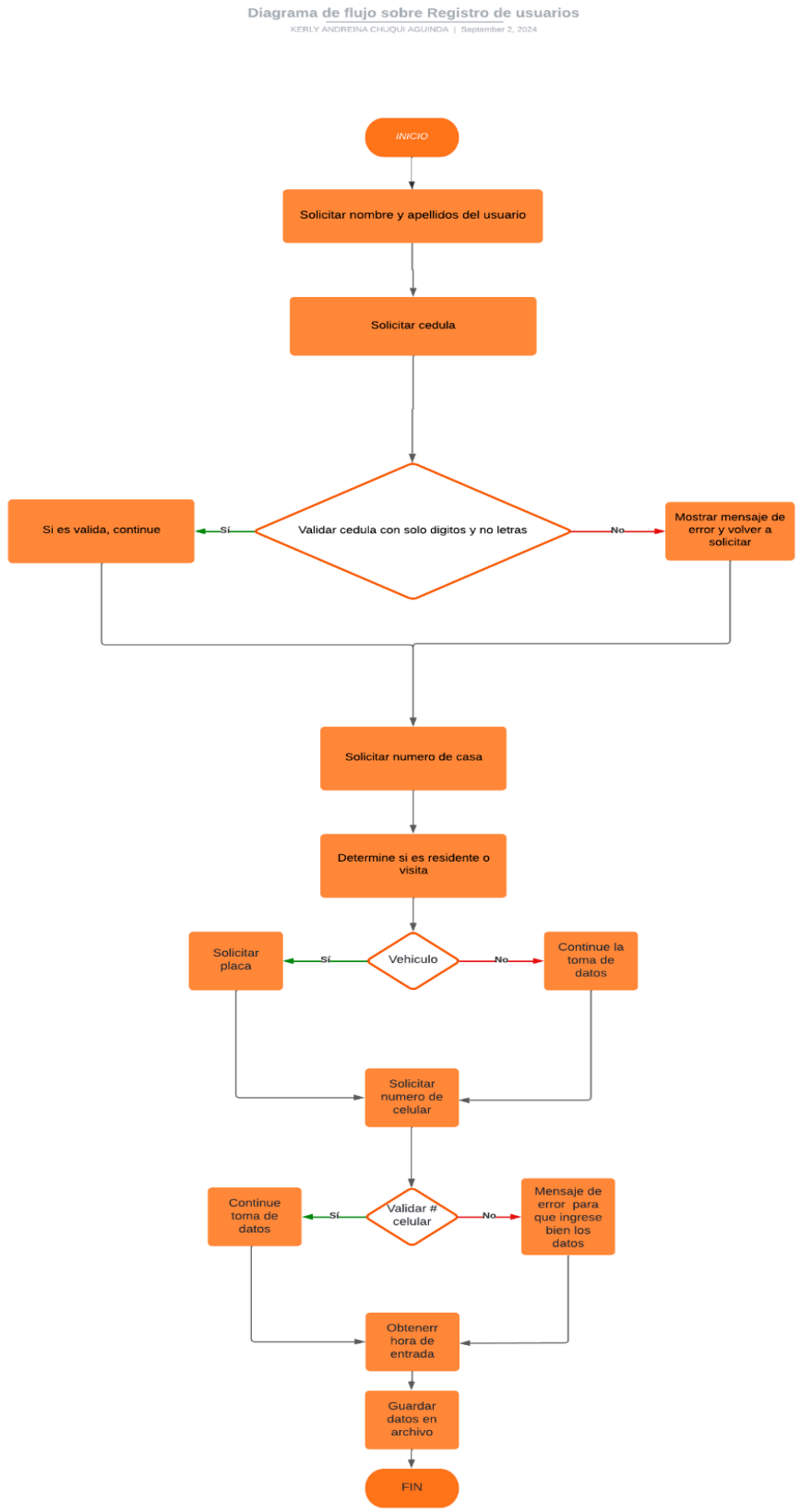
1. **CÓDIGO FUENTE**

```
#include <iostream>
#include <string>

using namespace std;

void clearScreen() {
    system("cls");
}

// Funcion para validar cedula
bool ValidarCedula(const string& cedula) {
    if (cedula.length() != 10) return false;
    for (char c : cedula) {
        if (!isdigit(c)) return false;
    }
    return true;
}

// Funcion para validar numero de celular
bool Validarnumero(const string& cellNumber) {
    if (cellNumber.length() != 10) return false;
    for (char c : cellNumber) {
        if (!isdigit(c)) return false;
    }
    return true;
}

// Funcion para ingresar datos del usuario
void registrarusuario() {
    clearScreen();
    string nombreApellido, cedula, tipoUsuario, motivoVisita, placaVehiculo, celular, numeroCasa;

    cout << "Ingrese nombre y apellido: ";
    getline(cin, nombreApellido);

    while (true) {
        cout << "Ingrese numero de cedula (10 digitos): ";
        getline(cin, cedula);
        if (ValidarCedula(cedula)) break;
        cout << "Cedula invalida. Debe tener 10 digitos y solo numeros.\n";
    }

    cout << "Numero de casa a la que se dirige: ";
    getline(cin, numeroCasa);

    cout << "Es residente o visita? (residente/visita): ";
    getline(cin, tipoUsuario);

    if (tipoUsuario == "visita") {
        cout << "Motivo de la visita: ";
        getline(cin, motivoVisita);
    }

    cout << "Tiene vehiculo? (si/no): ";
    string tieneVehiculo;
    getline(cin, tieneVehiculo);
```

```
    if (tieneVehiculo == "si") {
        cout << "Ingrese placa vehicular: ";
        getline(cin, placaVehiculo);
    }

    while (true) {
        cout << "Ingrese numero de celular (10 digitos): ";
        getline(cin, celular);
        if (Validarnumero(celular)) break;
        cout << "Numero de celular invalido. Debe tener 10 digitos y solo numeros.\n";
    }
}
```

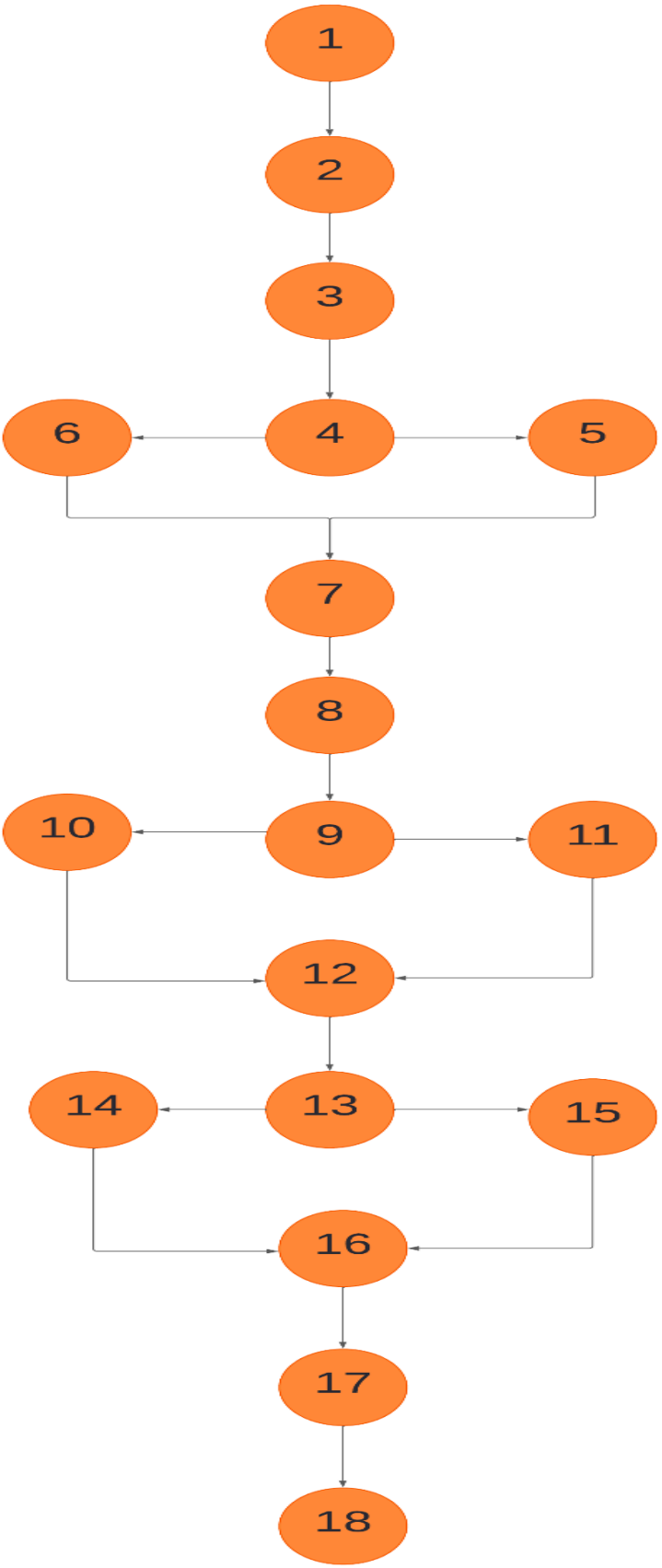
## 2. DIAGRAMA DE FLUJO (DF)

Realizar un DF del código fuente del numeral 1



### 3. GRAFO DE FLUJO (GF)

Realizar un GF en base al DF del numeral 2



## 4. IDENTIFICACIÒN DE LAS RUTAS (Camino básico)

Determinar en base al GF del numeral 3
**RUTAS**

**R1:** 1,2,3,4,6,7,8,9,10,12,13,14,16,17,18
**R2:** 1,2,3,4,5,7,8,9,11,12,13,15,16,17,18

## 5. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- V(G) = número de nodos predicados(decisiones)+1
  V(G)=3+1=4

- V(G) = A – N + 2
  V(G)= 20-18+2=4

DONDE:
**P:** Número de nodos predicado
**A:** Número de aristas
**N:** Número de nodos

**Prueba caja blanca de Requisito 2 :** Registro de salida para usuarios del conjunto habitacional Armenia II.

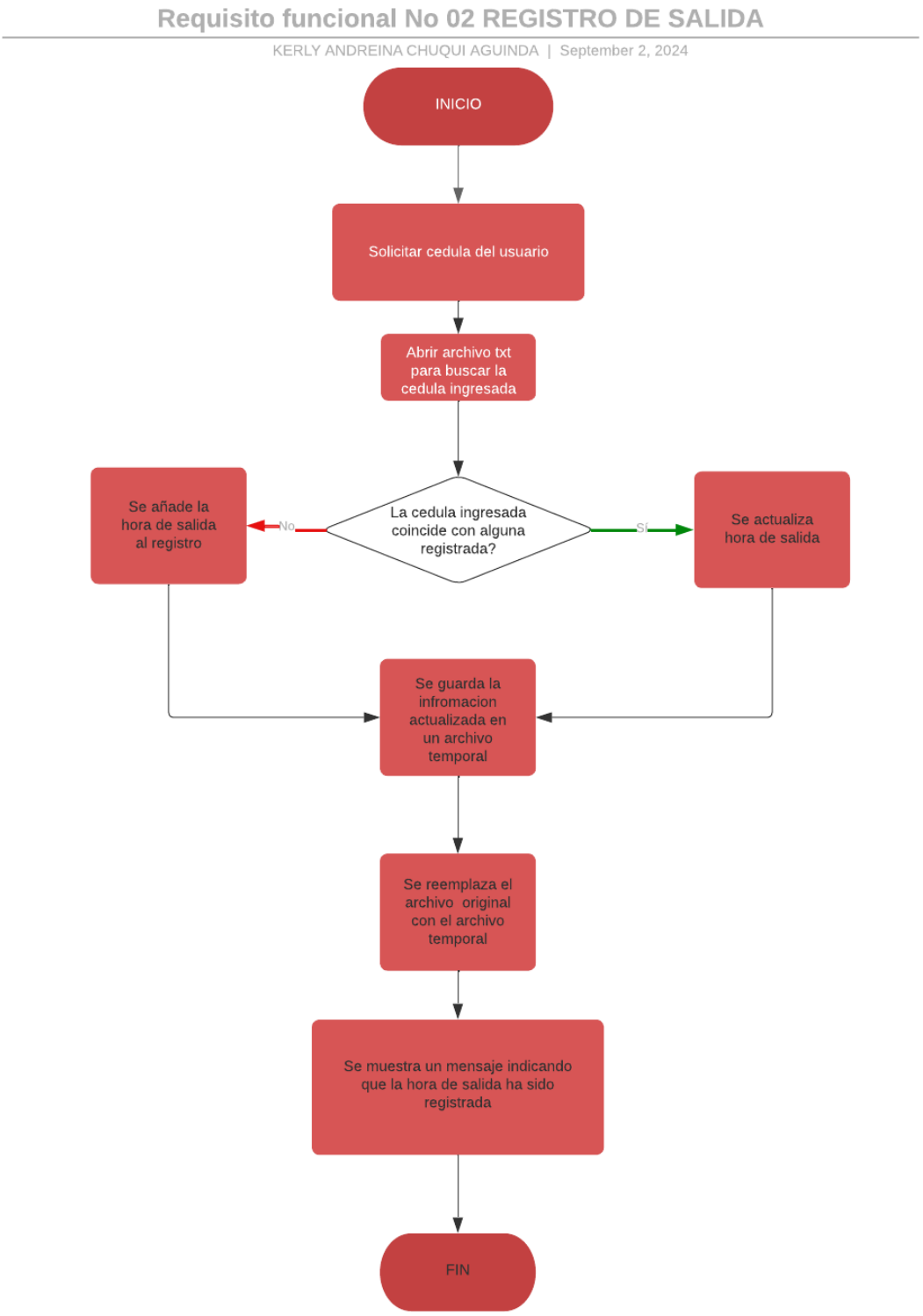
## 6. CÓDIGO FUENTE

```
#include <iostream>
#include <fstream>
#include <string>
#include <ctime>

using namespace std;

void clearScreen() {
   system("cls");
}

// Funcion para mostrar la hora actual
string obtenerhora() {
   time_t now = time(0);
   tm* localtm = localtime(&now);
   char buffer[80];
   strftime(buffer, sizeof(buffer), "%Y-%m-%d %H:%M:%S", localtm);
   return string(buffer);
}

// Funcion para registrar salida
void registrarsalida() {
   clearScreen();
   string cedula, line;
   string tempFileName = "temp.txt";
   string outputFileName = "usuarios.txt";

   cout << "Ingrese numero de cedula para registrar salida: ";
   getline(cin, cedula);

   ifstream inFile(outputFileName);
   ofstream outFile(tempFileName);

   bool found = false;
   if (inFile.is_open() && outFile.is_open()) {
      while (getline(inFile, line)) {
         size_t pos = line.find("Numero de Cedula: " + cedula);

         if (pos != string::npos) {
            string salidaHora = obtenerhora();
            size_t horaEntradaPos = line.find("Hora de Entrada:");

            if (horaEntradaPos != string::npos) {
               // Asegurate de agregar o actualizar la hora de salida
               size_t horaSalidaPos = line.find("Hora de Salida:");

               if (horaSalidaPos != string::npos) {
                  // Actualiza solo la hora de salida si ya existe
                  line = line.substr(0, horaSalidaPos) + "Hora de Salida: " + salidaHora;
               } else {
                  // Añade la hora de salida si no existe
                  line += "\nHora de Salida: " + salidaHora;
               }
```

```
                } else {
                    // La hora de entrada no se encontro, lo que indica un problema
                    line += "\nHora de Salida: " + salidaHora;
                }
                outFile << line << endl;
                found = true;
            } else {
                outFile << line << endl;
            }
        }
        inFile.close();
        outFile.close();

        if (found) {
            remove(outputFileName.c_str());
            rename(tempFileName.c_str(), outputFileName.c_str());
            cout << "Hora de salida registrada exitosamente.\n";
        } else {
            remove(tempFileName.c_str());
            cout << "Cedula no encontrada.\n";
        }
    } else {
        cout << "No se pudo abrir el archivo.\n";
    }
}
```

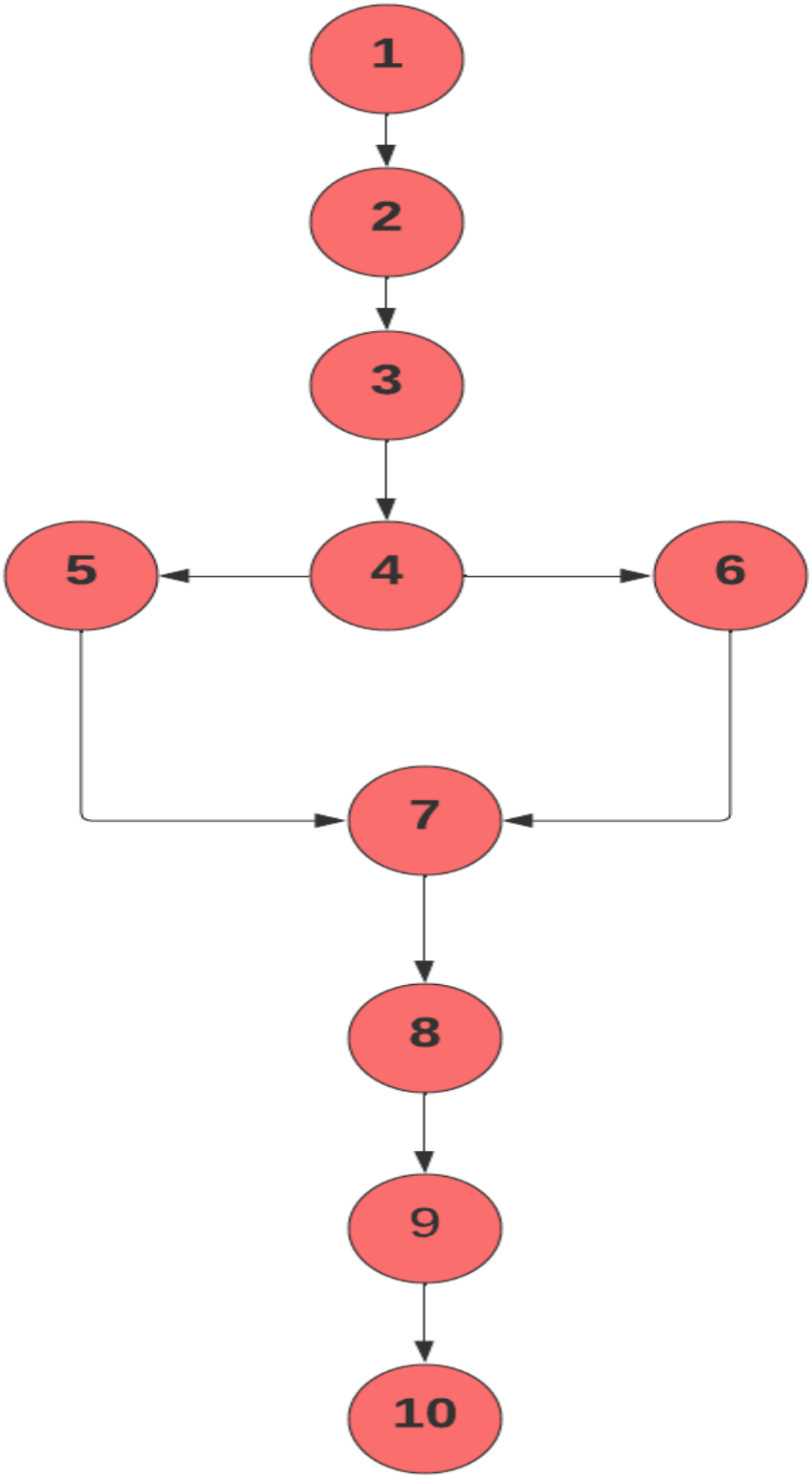
## 7. DIAGRAMA DE FLUJO (DF)

Realizar un DF del código fuente del numeral anterior



**8. GRAFO DE FLUJO (GF)**

Realizar un GF en base al DF del numeral anterior



## 9. IDENTIFICACIÒN DE LAS RUTAS (Camino básico)

Determinar en base al GF del numeral 3
**RUTAS**

**R1:** 1,2,3,4,5,7,8,9,10
**R2:** 1,2,3,4,6,7,8,9,10

## 10. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- V(G) = número de nodos predicados(decisiones)+1
  V(G)=1+1=2

- V(G) = A – N + 2
  V(G)= 10-10+2=2

DONDE:
**P:** Número de nodos predicado
**A:** Número de aristas
**N:** Número de nodos

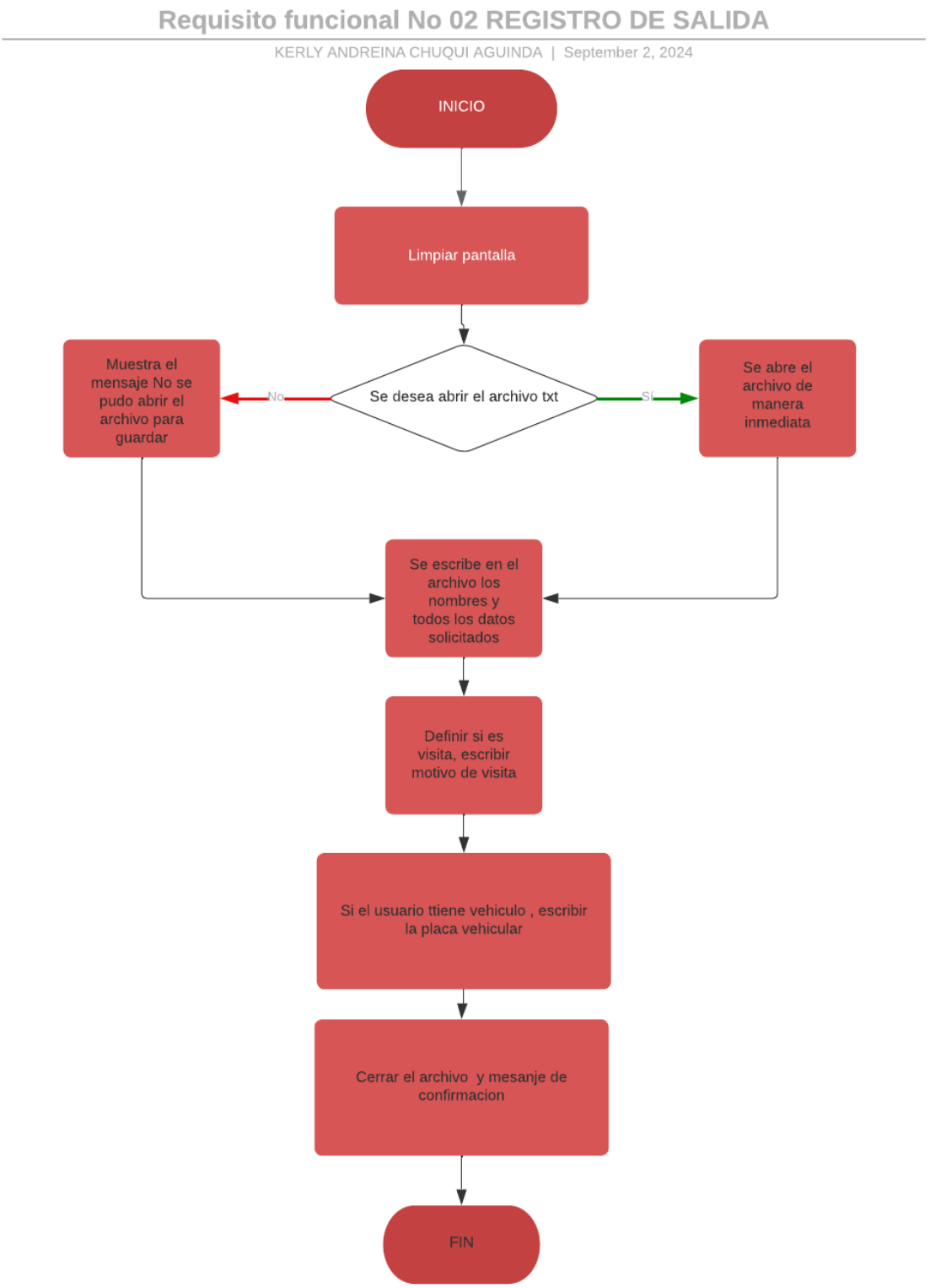
**Prueba caja blanca de Requisito 3 :** Guardar usuarios en archivo.

**11. CÓDIGO FUENTE**

```
// Guardar en archivo
ofstream outFile("usuarios.txt", ios::app);
if (outFile.is_open()) {
    outFile << "Nombre y Apellido: " << nombreApellido << endl;
    outFile << "Numero de Cedula: " << cedula << endl;
    outFile << "Numero de Casa: " << numeroCasa << endl;
    outFile << "Tipo de Usuario: " << tipoUsuario << endl;
    if (tipoUsuario == "visita") {
        outFile << "Motivo de la Visita: " << motivoVisita << endl;
    }
    if (tieneVehiculo == "si") {
        outFile << "Placa Vehicular: " << placaVehiculo << endl;
    }
    outFile << "Numero de Celular: " << celular << endl;
    outFile << "Hora de Entrada: " << entradaHora << endl;
    outFile << "----------" << endl;
    outFile.close();
} else {
    cout << "No se pudo abrir el archivo para guardar.\n";
}

 cout << "Usuario registrado exitosamente. Hora de entrada: " << entradaHora <<
endl;
}
```

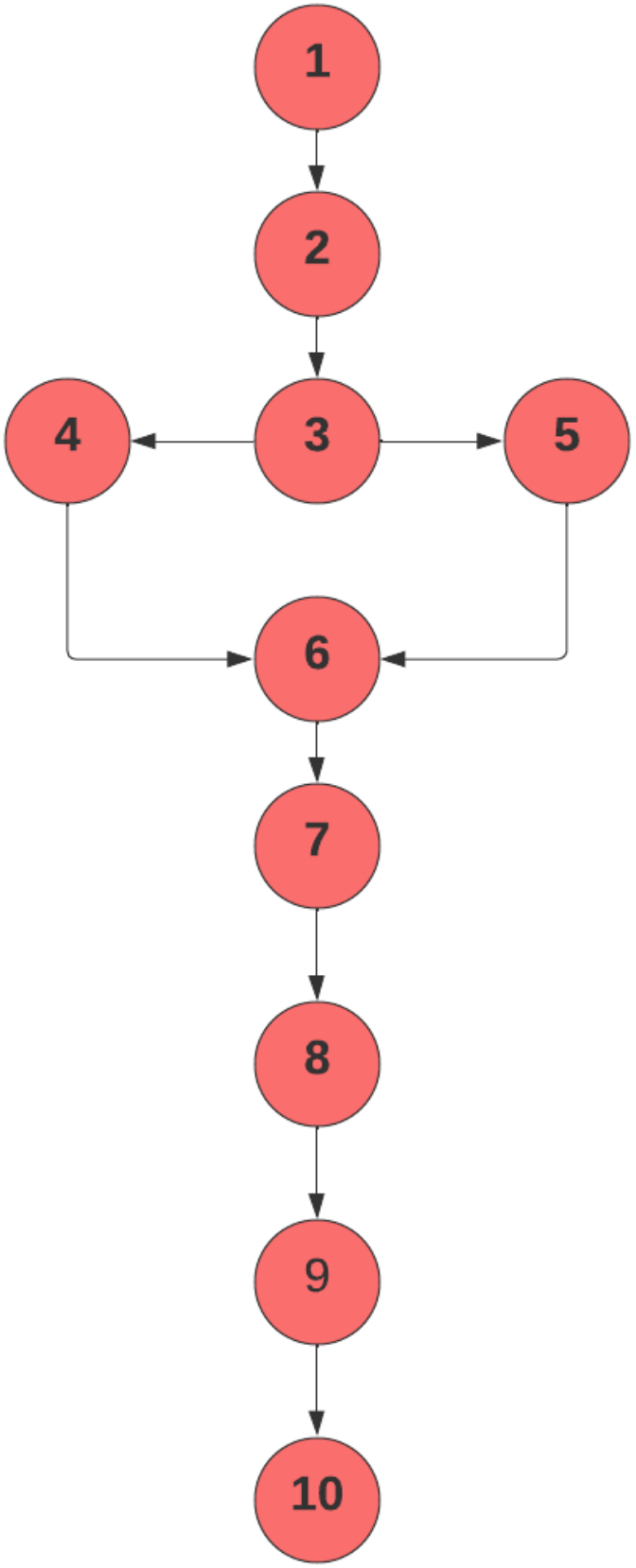
## 12. DIAGRAMA DE FLUJO (DF)

Realizar un DF del código fuente del numeral anterior



**13. GRAFO DE FLUJO (GF)**

Realizar un GF en base al DF del numeral anterior



## 14. IDENTIFICACIÒN DE LAS RUTAS (Camino básico)

Determinar en base al GF del numeral 3
**RUTAS**

**R1:** 1,2,3,4,6,7,8,9,10
**R2:** 1,2,3,5,6,7,8,9,10

## 15. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- V(G) = número de nodos predicados(decisiones)+1
  V(G)=1+1=2

- V(G) = A – N + 2
  V(G)= 10-10+2=2

DONDE:
**P:** Número de nodos predicado
**A:** Número de aristas
**N:** Número de nodos

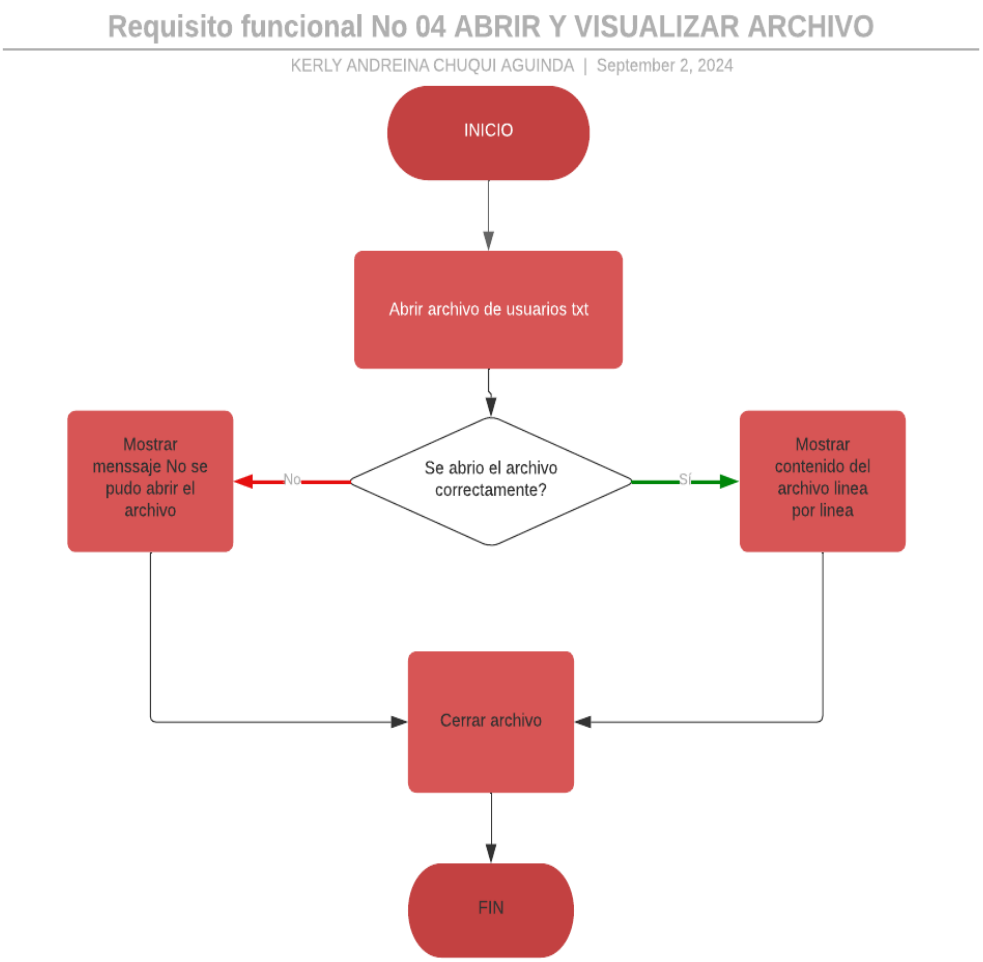
**Prueba caja blanca de Requisito 4 :** Abrir y visualizar usuarios en archivo.

**16. CÓDIGO FUENTE**

```
// Funcion para abrir y visualizar el archivo de usuarios
void abrirarchivo() {
    clearScreen();
    ifstream inFile("usuarios.txt");
    string line;

    if (inFile.is_open()) {
        cout << "Contenido del archivo usuarios.txt:\n";
        cout << "-------------------------------------\n";
        while (getline(inFile, line)) {
            cout << line << endl;
        }
        inFile.close();
    } else {
        cout << "No se pudo abrir el archivo para leer.\n";
    }
}
```

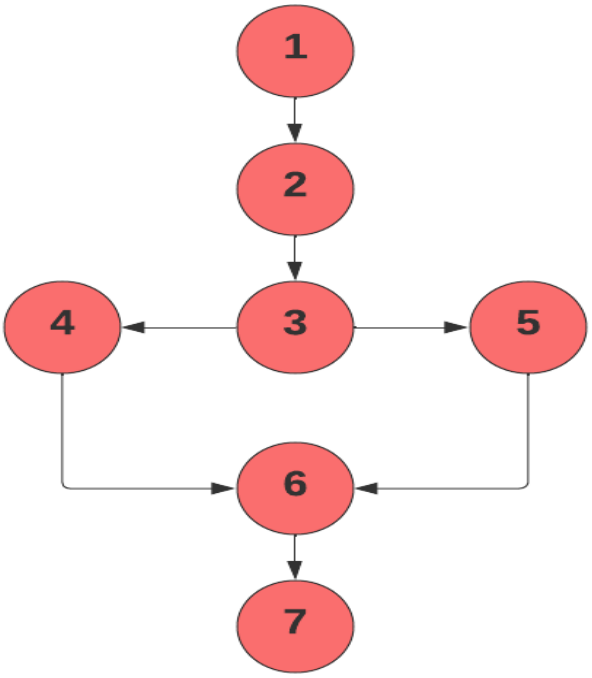
**17. DIAGRAMA DE FLUJO (DF)**

Realizar un DF del código fuente del numeral anterior



## 18. GRAFO DE FLUJO (GF)

Realizar un GF en base al DF del numeral anterior




## 19. IDENTIFICACIÒN DE LAS RUTAS (Camino básico)

Determinar en base al GF del numeral 3
**RUTAS**

**R1:** 1,2,3,4,6,7
**R2:** 1,2,3,5,6,7

## 20. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- V(G) = número de nodos predicados(decisiones)+1
  V(G)=1+1=2

- V(G) = A – N + 2
  V(G)= 7-7+2=2

DONDE:
**P:** Número de nodos predicado
**A:** Número de aristas
**N:** Número de nodos

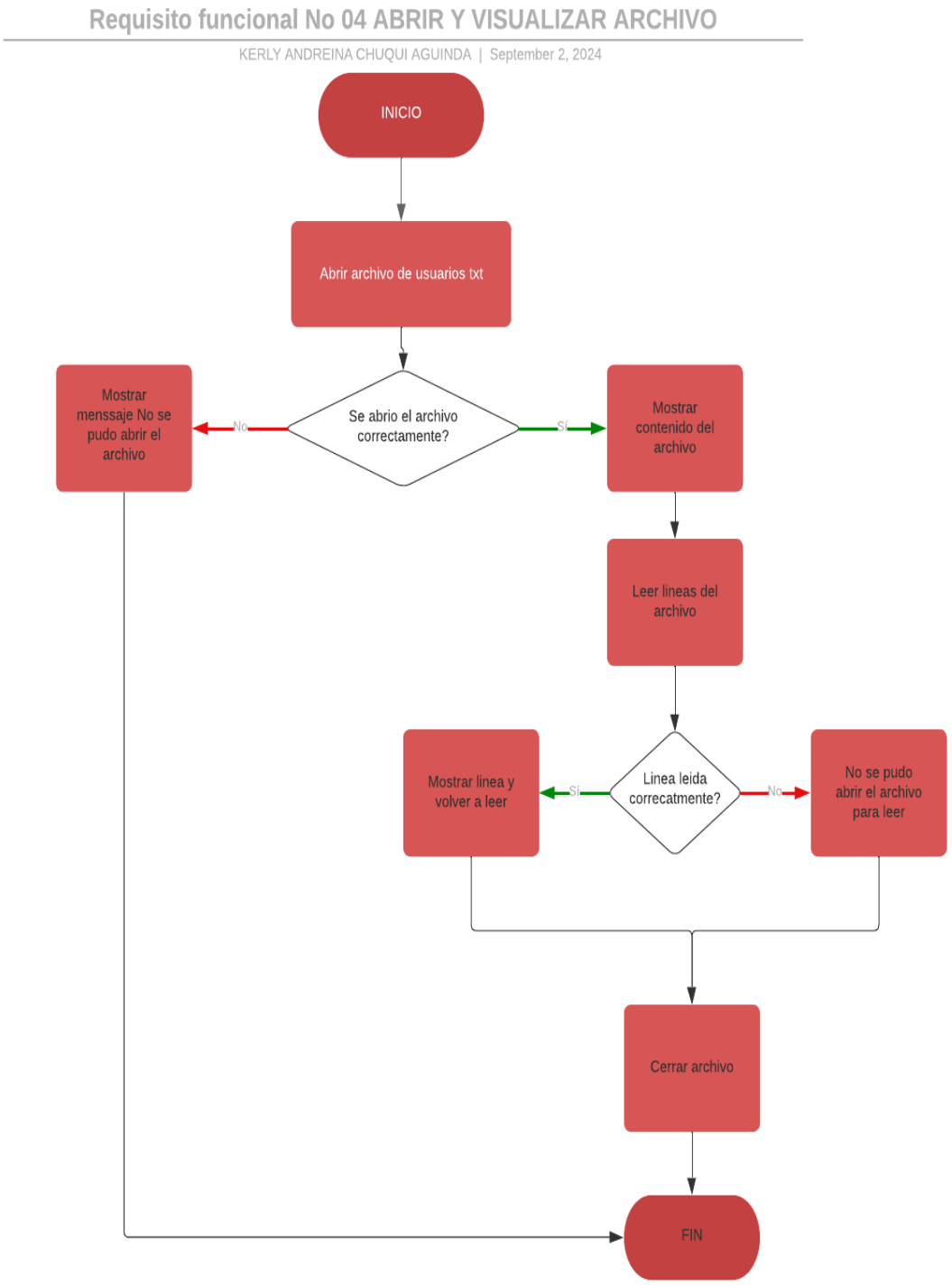
**Prueba caja blanca de Requisito 5 :** Mostrar lista de usuarios registrados.

21. **CÓDIGO FUENTE**

```
// Funcion para mostrar lista de usuarios
void mostrarusuario() {
    clearScreen();
    ifstream inFile("usuarios.txt");
    string line;

    if (inFile.is_open()) {
        cout << "Contenido del archivo usuarios.txt:\n";
        cout << "-------------------------------------\n";
        while (getline(inFile, line)) {
            cout << line << endl;
        }
        inFile.close();
    } else {
        cout << "No se pudo abrir el archivo para leer.\n";
    }
}
```

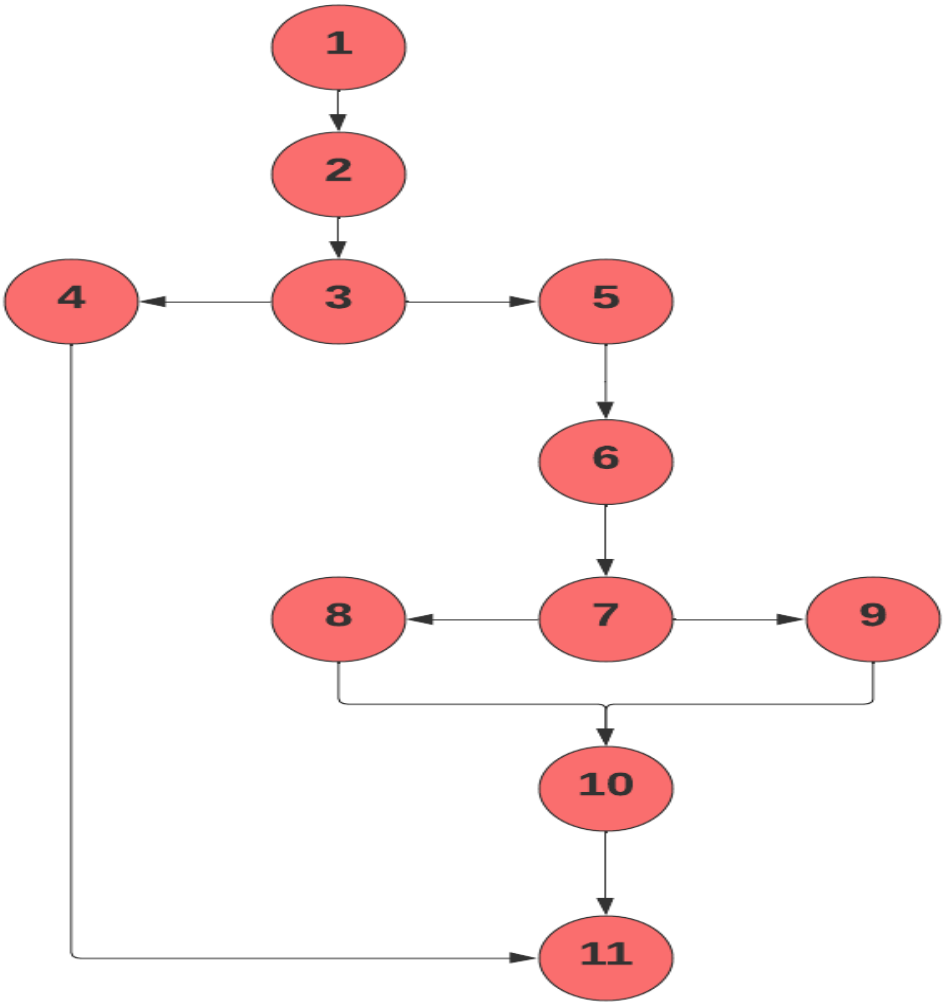
## 22. DIAGRAMA DE FLUJO (DF)

Realizar un DF del código fuente del numeral anterior



### 23. GRAFO DE FLUJO (GF)

Realizar un GF en base al DF del numeral anterior




### 24. IDENTIFICACIÒN DE LAS RUTAS (Camino básico)

Determinar en base al GF del numeral 3
**RUTAS**

**R1:** 1,2,3,4,11
**R2:** 1,2,3,5,6,7,8,10,11
**R3**: 1,2,3,5,6,7,9,10,11

### 25. COMPLEJIDAD CICLOMÁTICA

Se puede calcular de las siguientes formas:

- V(G) = número de nodos predicados(decisiones)+1
  V(G)=2+1=3

- V(G) = A – N + 2
  V(G)= 12-11+2=3

DONDE:
**P:** Número de nodos predicado
**A:** Número de aristas
**N:** Número de nodos