Proyecto 3: Calculadora IP — CIDR y VLSM

Objetivo:
Desarrollar dos herramientas: (1) una **Calculadora IP (CIDR)** y (2) una **Calculadora IP VLSM** que permitan calcular subredes, rangos, broadcast, máscaras y asignaciones según requisitos de hosts. Entregar documento, ejecutable y video explicativo con participación de todos los integrantes.

Integrantes por equipo: 4 alumnos (todos deben aparecer y participar en el video).

Entregables (obligatorios)

1. **Documento** (PDF) con:
   - Portada (nombre del proyecto, materia, módulo, integrantes con matrícula y correo, profesor, fecha).
   - Introducción (objetivo del proyecto y breve descripción).
   - Desarrollo (explicación de diseño, estructura, algoritmo VLSM, ejemplos de uso).
   - Código fuente comentado (estructura de archivos, dependencias).
   - Instrucciones de compilación/ejecución.
   - Pruebas de funcionamiento (casos de prueba y resultados).
   - Conclusiones y retroalimentación del equipo.
2. **Archivo ejecutable** (programa listo para correr):
   - Si es aplicación de escritorio: .exe (Windows) o instrucciones para crear ejecutable cross-platform.
   - Si es web app: carpeta lista para desplegar o URL si lo hospedan (opcional).
   - Si es script: incluir un wrapper que ejecute sin fallos (por ejemplo, .bat/.sh).
3. **Video** (máx. 6–8 min por equipo) que muestre:
   - Demo en funcionamiento (uso de ambas calculadoras)
   - Breve exposición de diseño/algoritmo (mín. 1 minuto cada integrante explicando distintas partes)
   - Cada integrante debe hablar — mostrar quién hizo qué
4. **Código fuente** en un repositorio (opcional: GitHub): se puede incluir link en el PDF y añadir ZIP con el código.
5. **Archivo README** con pasos claros para ejecutar y pruebas.

Requisitos funcionales mínimos

Calculadora IP (CIDR)

- Entrada: dirección IP + prefijo (ej.: `192.168.1.0/24`) **o** IP + máscara.
- Salida:
  - Red (network address)
  - Rango de hosts (primer host — último host)
  - Dirección broadcast
  - Máscara en notación decimal punteada y prefijo `/n`
  - Número de hosts disponibles
  - Representación binaria opcional
- Validación de entradas y mensajes de error claros.

Calculadora VLSM

- Entrada: red base (ej.: `10.0.0.0/16`) y lista de requerimientos por subred (p. ej. `DepartamentoA: 120 hosts, DepartamentoB: 50 hosts, ...`).
- Algoritmo VLSM que:
  - Ordena subredes por tamaño requerido (mayor → menor).
  - Calcula la máscara más pequeña que acomode los hosts (considerando 2 direcciones no asignables: network y broadcast).
  - Asigna bloques contiguos dentro de la red base sin solapamientos.
  - Devuelve para cada subred: Network, Mask (prefijo y dotted), Rango de hosts, Broadcast, número de hosts válidos y espacio desperdiciado.
- Manejo de errores cuando la red base no tiene suficiente espacio.

Recomendaciones técnicas (lenguajes / herramientas)

- **Lenguajes sugeridos:** Python (Tkinter/Flask), JavaScript (Node + Express + React/Vue), Java (Swing/JavaFX), C# (.NET/WPF).
- **Generar ejecutable:** PyInstaller para Python, pkg/webpack/electron para JS desktop, jar para Java, publish para .NET.
- **Pruebas unitarias:** incluir tests básicos para funciones críticas (ej.: cálculo de máscara, cálculo de broadcast).
- **UI:** consola funcional + GUI opcional. La prioridad es correcto cálculo y salida legible.

Pseudocódigo / algoritmo VLSM (detallado — copiar en el documento)

1. **Entrada:** `network_base`, `prefix_base`, lista `subnets = [(name, required_hosts), ...]`
2. **Preparar lista:** para cada `required_hosts` calcular `needed_hosts = required_hosts + 2` (network + broadcast).
3. **Ordenar** `subnets` por `needed_hosts` descendente.
4. **Puntero** `current_network = network_base` (como entero 32-bit).
5. Para cada `subnet` en `subnets`:
   - Calcular `mask_bits = 32 - ceil(log2(needed_hosts))`. (ej: si needed_hosts = 120 → ceil(log2(120)) = 7 → mask_bits = 25)
   - `block_size = 2^(32 - mask_bits)` (número de direcciones totales del bloque)
   - Asignar `network_address = current_network`
   - `first_host = network_address + 1`
   - `last_host = network_address + block_size - 2`
   - `broadcast = network_address + block_size - 1`
   - Guardar la asignación con máscara `/mask_bits`
   - Actualizar `current_network = current_network + block_size`
6. **Validar** que `current_network` no exceda `network_base + 2^(32 - prefix_base)`
7. Si excede → error: "espacio insuficiente".

Nota técnica: trabajar con IPs como enteros facilita sumas y alignments (usar funciones para convertir dotted↔integer).

Casos de prueba sugeridos (inclúyanlos en el documento)

- CIDR: `192.168.10.0/24` → network `192.168.10.0`, host range `192.168.10.1–192.168.10.254`, broadcast `192.168.10.255`.
- VLSM: Base `192.168.0.0/24`, requerimientos: A=100, B=50, C=25, D=10. Mostrar asignación de subredes y espacio restante.
- Caso límite: pedir más hosts de los disponibles → debe arrojar error claro.

Formato del código y buenas prácticas (evaluadas)

- Comentarios claros en funciones críticas.
- Estructura modular (separar lógica de cálculo, UI y utilerías).
- Manejo de errores y validación.
- Instrucciones de instalación y requisitos en README.
- Licencia/atribución si usan bibliotecas externas.

Video (requisitos)

- Duración: **3–8 minutos**.
- Contenido mínimo:
  1. Presentación del equipo y roles (30–45 s).
  2. Explicación breve del diseño y algoritmo VLSM (cada integrante debe explicar al menos 20–30 s de contenido técnico distinto).
  3. Demo en vivo: introducir datos y mostrar resultados de ambas calculadoras.
  4. Mostrar pruebas con los casos de prueba incluidos en el documento.
  5. Conclusiones y lecciones aprendidas.
- Formato: mp4 preferible. Archivo nombrado: `EquipoX_CalculadoraIP.mp4`.
- Subir video junto con el resto de los archivos (no enlaces privados; si usan YouTube, dejarlo en modo no listado y adjuntar enlace en el PDF).

Rubrica de evaluación (100 puntos)

- **Funcionalidad (35 pts)**
  - Calculadora CIDR correcta: 10 pts
  - Calculadora VLSM correcta y sin solapamientos: 15 pts
  - Manejo de errores y validaciones: 10 pts
- **Calidad del código y documentación (20 pts)**
  - Código comentado y estructurado: 8 pts
  - README + instrucciones de ejecución: 6 pts
  - Tests y casos de prueba incluidos: 6 pts
- **Entrega ejecutable y packaging (15 pts)**
  - Ejecutable funcional y bien empaquetado: 10 pts
  - Inclusión de dependencias o instalador claro: 5 pts
- **Documento (15 pts)**
  - Portada, Introducción, Desarrollo (incluye algoritmo): 6 pts
  - Código fuente en el documento (fragmentos importantes) y explicación: 5 pts
  - Conclusiones y pruebas documentadas: 4 pts

- **Video y participación (15 pts)**
  - Demo clara y comprensible: 6 pts
  - Cada integrante participa activamente (mín. 3 pts por integrante): 9 pts (si alguien no participa se restan hasta 9 pts en total de este apartado).

Penalizaciones: entregas tardías: −10% por día (decisión del profesor), código no ejecutable: −peso proporcional en funcionalidad.

Checklist de entrega (copiar y usar como portada interna)

- ☐ PDF del documento (Portada, Introducción, Desarrollo, código, pruebas, conclusiones).
- ☐ ZIP con el código fuente y/o link a repo.
- ☐ Ejecutable / instrucciones de ejecución (incluir .bat/.sh si aplica).
- ☐ Video (mp4) o enlace no listado.
- ☐ Archivo README (paso a paso para ejecutar).
- ☐ Lista de integrantes y roles (firma electrónica o texto).

Formato de nombres: `EquipoX_CalculadoraIP.zip`, `EquipoX_CalculadoraIP.pdf`, `EquipoX_CalculadoraIP.mp4`.

Ejemplo de calculadora IP

1. Ejemplo de Calculadora IP CIDR

- https://www.aprendaredes.com/calculadora-ip/
- https://aprendaredes.com/cgi-bin/ipcalc/ipcalc_cgi1

2. Ejemplo de Calculadora VLSM

- https://arcadio.gq/calculadora-subredes-vlsm.html