Proyecto C

Este laboratorio se desarrollará utilizando una instalación (hecha por el estudiante) de Ubuntu 18.04 LTS, 20.04 LTS, o 22.04 LTS. **Si el estudiante usa otra versión, debe indicarlo en el documento.**

Instrucciones generales:
- El trabajo es grupal, 2 o 3 estudiantes (altamente recomendado que sean 3).

- Se deben entregar cuatro archivos “proyecto1.c”, “proyecto2.c”, “proyecto3.c”, y el “Makefile”, y se deben subir a Mediación Virtual. Puede añadir un archivo README si la compilación necesitara instrucciones especiales.

- También se debe entregar el enlace al repositorio de GitHub donde se encuentren estos tres archivos. Nota: entregue los tres archivos en Mediación Virtual, además del enlace.

- No debe entregar el ejecutable compilado (el profesor lo compilará manualmente).

- El código debe venir bien documentado.

- Cada estudiante debe entregar una autoevaluación del grupo (se asigna nota a cada integrante), la cual tendrá un peso del 40% de la nota. Esta autoevaluación debe ser añadida como comentario en la entrega de Mediación Virtual.

- Use un encabezado al inicio del programa para dar una pequeña descripción del programa, incluyendo su nombre.

- Se debe crear un Makefile para compilar automáticamente cada proyecto. Use tres “targets”: p1, p2, y p3.

- Se penalizará con un 40% de la nota si no se respetan todos estos lineamientos.

Parte 1: Camino más corto (45%)

Cree un programa que encuentre el camino más corto entre dos ciudades. Cada ciudad está ubicada en un plano bidimensional. Una persona desea moverse desde la ciudad de inicio hasta la ciudad destino, y necesita encontrar el camino más corto.

El programa debe preguntar la cantidad de ciudades a utilizar, y los caminos que las conectan a otras ciudades (cada camino tiene una distancia asociada). Puede utilizar una representación tipo matriz para representar las distancias entre cada ciudad. Las distancias recíprocas son iguales. Por ejemplo:

Origen/Destino	A	B	C	D	E	F
A	-	-1	-1	5	-1	-1
B	-1	-	4	6	1	4
C	-1	4	-	2	2	10
D	5	6	2	-	1	-1
E	-1	1	2	1	-	-1
F	-1	4	10	-1	-1	-

En este ejemplo, “-1” significa que NO hay camino de una ciudad a otra, y cada número representa la distancia del camino entra una ciudad a otra (unidad arbitraria).

El usuario, además de proporcionar la cantidad de ciudades y las distancias entre ellas, debe definir su punto de partida y su punto de destino. El programa debe encontrar la ruta más corta entre ellas. Por ejemplo: si el usuario quiere ir desde A hasta B se podrían calcular varios caminos:

A -> D -> B: 5 +6 = 11

A -> D -> C -> E -> B: 5 + 2 + 2 + 1 = 10

A -> D -> C -> B: 5 + 2 + 4 = 11

En este caso el camino A, D, C, E, B es el camino más corto ya que el total de la distancia de cada camino es el menor posible.

Aspectos a evaluar (los puntos suman 100, que equivalen al 45%):

1. (20 pts) El programa debe recibir la información del usuario correctamente.

2. (75 pt) El programa calcula la distancia más corta posible y muestra la ruta utilizada.

3. (5 pt) Añada al Makefile un target para compilar este proyecto. El target debe ser “p1”, y debe crear un ejecutable llamado “exe”.

$ make p1

Puede asumir que solamente se usarán números enteros como entrada.

Parte 2: Ángulos (25%)

Cree un programa que reciba las coordenadas de un polígono bidimensional (de cualquier cantidad de lados). El programa debe calcular todos los ángulos internos. Solamente serán válidos los polígonos convexos.

Aspectos a evaluar (los puntos suman 100, que equivalen al 25%):

1. (10 pts) El programa recibe la información del usuario correctamente.

2. (65 pts) El programa imprime exitosamente los ángulos, con la cantidad correcta de ellos.

3. (20 pts) El programa reconoce si el polígono es válido o no (solamente convexos, de tres o más lados).

4. (5 pt) Añada al Makefile un target para compilar este proyecto. El target debe ser “p2”, y debe crear un ejecutable llamado “exe”.

$ make p2

Puede asumir que solamente se usarán números flotantes como entrada.

Parte 3: Memoria Dinámica (30%)

El reto de este ejercicio es poder crear una función a partir de una descripción de texto, que cumpla con cada requerimiento. Los aspectos de implementación y formato se dejan a la libre. **El estudiante debe asumir cosas**, y recibirá poca ayuda del profesor adrede.

Cree una función que reciba del usuario una cantidad arbitraria de números **enteros**, que representen las posiciones dentro de una matriz NxM.

La función debe reservar memoria dinámicamente para colocar cada uno de los números ingresados por el usuario.

La función debe devolver al usuario las direcciones de memoria de los números que inician cada FILA.

La función retorna un número entero: 0 si no hay errores, -1 si hubo algún error.

Cree un programa principal de prueba (que usará el profesor) que llame a dicha función, donde se demuestre el correcto funcionamiento de la misma.

No olvide liberar la memoria utilizada.

Añada al Makefile un target para compilar este proyecto. El target debe ser “p3”, y debe crear un ejecutable llamado “exe”.

$ make p3