UTEC
UNIVERSIDAD DE INGENIERÍA Y TECNOLOGÍA

Programación II
Práctica Calificada 2
Pregrado
2021-II
Profesor: Ian Paul Brossard
Lab:2.07

Indicaciones específicas:

- Esta evaluación contiene 11 páginas (incluyendo esta página) con 3 preguntas. El total de puntos son 20.
- El tiempo límite para la evaluación es 100 minutos.
- Cada pregunta deberá ser respondida en un solo archivo con el número de la pregunta.
  - `p1.cpp`
  - `p2.cpp`
  - `p3.cpp`
- Deberás subir estos archivos directamente a www.gradescope.com, uno en cada ejercicio. También puedes crear un .zip

Competencias:

- Para los alumnos de la carrera de Ciencia de la Computación

  Aplicar conocimientos de computación apropiados para la solución de problemas definidos y sus requerimientos en la disciplina del programa. (nivel 2)

  Diseñar, implementar y evaluar soluciones a problemas complejos de computación.(nivel 2)

  Crear, seleccionar, adaptar y aplicar técnicas, recursos y herramientas modernas para la práctica de la computación y comprende sus limitaciones. (nivel 2)

- Para los alumnos de las carreras de Ingeniería

  Aplicar conocimientos de ingeniería en la solución de problemas complejos de ingeniería (nivel 2).

  Diseñar soluciones relacionados a problemas complejos de ingeniería (nivel 2)

  Crear, seleccionar y utilizar técnicas, habilidades, recursos y herramientas modernas de la ingeniería y las tecnologías de la información, incluyendo la predicción y el modelamiento, con la comprensión de sus limitaciones (nivel 2)

- Para los alumnos de Administración y Negocios Digitales

  Analizar información verbal y/o lógica proveniente de distintas fuentes, encontrando relaciones y presentándola de manera clara y concisa (nivel 2)

Analizar y evaluar el comportamiento del consumidor y el desarrollo de estrategias comerciales (nivel 2)

Trabajar de manera efectiva con equipos multidisciplinarios y diversos en género, nacionalidad, edad, etc. (nivel 2)

Calificación:

Tabla de puntos (sólo para uso del professor)

Question	Points	Score
1	7	
2	6	
3	7	
Total:	20	

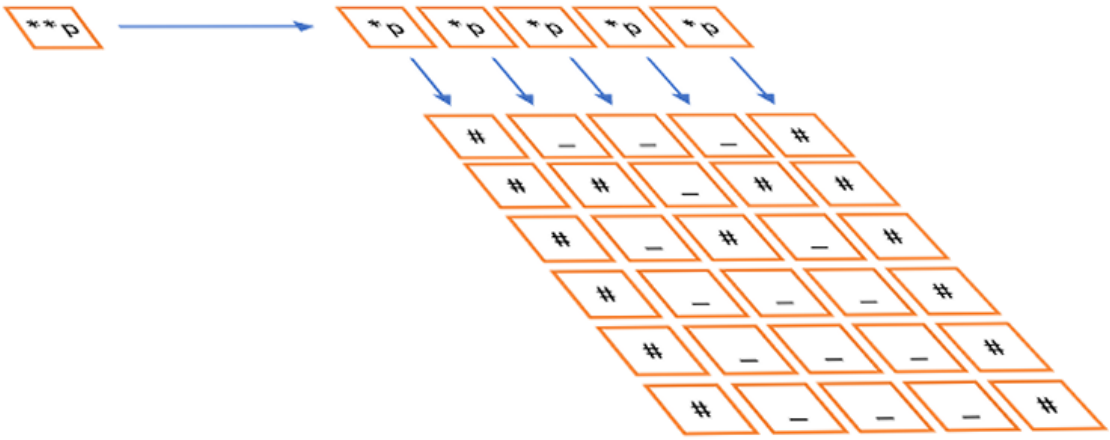
1. (7 points) **Letra M**
   La letra M es la decimotercera letra del alfabeto español. Su grafema proviene de la Mu mayúscula griega. Es un letra con mucha presencia en el mundo del diseño. Muchas marcas como McDonald's, Mazda, Movistar o Motorola, utilizan una M como su logotipo.

   Usted deberá implementar un programa en lenguaje C++ que dibuje una letra M. Su programa deberá solicitar al usuario el ancho y alto de su M, e imprimir una M con los caracteres # y _ (tal y como lo hacíamos en los ejercicios de bucles anidados).

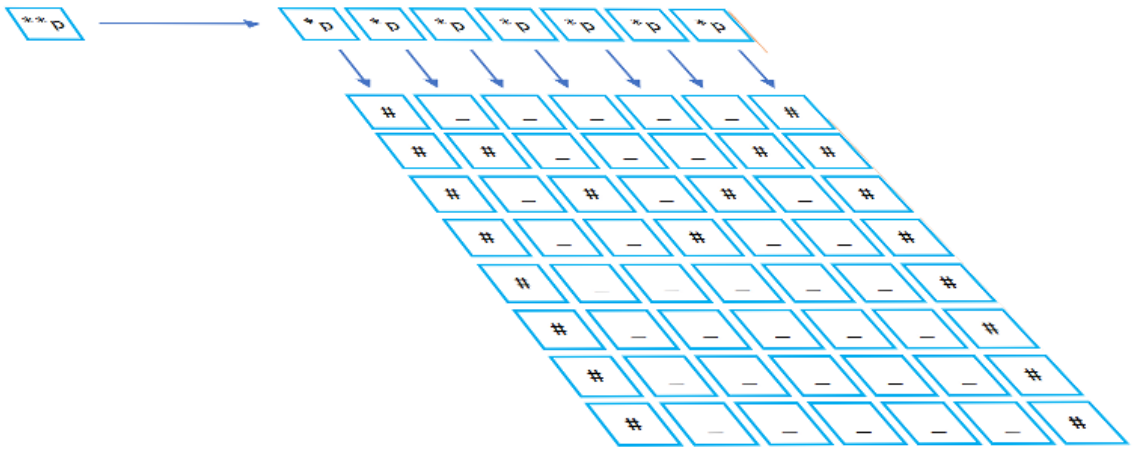
   El ancho de la M deberá ser un número impar mayor o igual a 5. El alto de la M deberá ser un número mayor o igual al ancho.

   Usted deberá usar el concepto de puntero a puntero para representar una matriz que contenga la información correspondiente a su M.

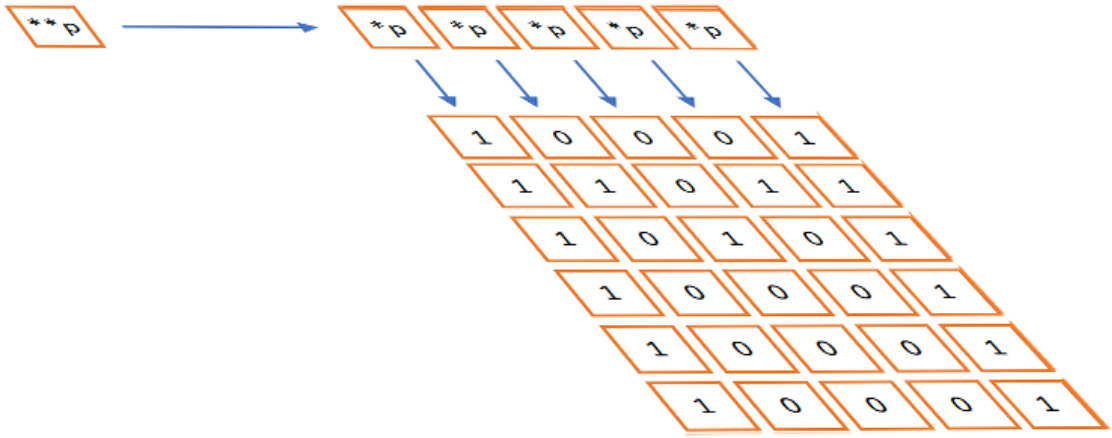
   Por ejemplo, si usted desea hacer una M de 5 de ancho y 6 de alto, usted podría implementar un puntero a puntero, que apunte a un arreglo de punteros a char, y cada puntero a char podría apuntar a un arreglo de chars, que contengan # o _ según se muestra a continuación:



   Y si usted desea hacer una M de 7 de ancho y 8 de alto, usted podría implementar un puntero a puntero, que apunte a un arreglo de punteros a char, y cada puntero a char podría apuntar a un arreglo de chars, que contengan # o _ según se muestra a continuación:



Asimismo, si usted desea hacer una M de 5 de ancho y 6 de alto, usted podría implementar un puntero a puntero, que apunte a un arreglo de punteros a int, y cada puntero a int podría apuntar a un arreglo de ints, que contengan 1 o 0 según se muestra a continuación:



Usted puede decidir utilizar int o char para la representación de su M, sin embargo, para la impresión de la M solo se deben usar # y _. Para la impresión, se recomienda dejar un espacio entre cada carácter impreso, para facilitar su visualización. Así, cada celda donde debe aparecer un # en realidad imprimirá un # seguido por un espacio, y cada celda donde debe aparecer un _ en realidad imprimirá un _ seguido por un espacio.

Sugerencia:

- **Se recomienda utilizar los índices i,j para el manejo de filas y columnas. i para las filas y j para las columnas.**

- **Se recomienda utilizar el concepto de diagonal principal (para esto, solo debe evaluar i == j) para la primera mitad de la cabecera de la M.**
- **Se recomienda utilizar el concepto de diagonal secundaria (para esto, solo debe evaluar i == ancho - j - 1) para la primera mitad de la cabecera de la M.**
- **Se recomienda hacer una pequeña prueba usando cout, paraque usted pueda imprimir una diagonal principal o una diagonal secundaria y comprenda la utilidad de las 2 sugerencias anteriores.**

Algunos ejemplos de diálogo de este programa serían:

Ejemplo de ejecución 1:

```
Ingrese ancho: 5
Ingrese alto: 6
# _ _ _ #
# # _ # #
# _ # _ #
# _ _ _ #
# _ _ _ #
# _ _ _ #
```

Ejemplo de ejecución 2:

```
Ingrese ancho: 5
Ingrese alto: 5
# _ _ _ #
# # _ # #
# _ # _ #
# _ _ _ #
# _ _ _ #
```

Ejemplo de ejecución 3:

```
Ingrese ancho: 7
Ingrese alto: 3
El alto debe ser mayor o igual al ancho
```

Ejemplo de ejecución 4:

```
Ingrese ancho: 8
El ancho debe ser impar
```

Ejemplo de ejecución 7:

```
Ingrese ancho: 7
Ingrese alto: 12
# _ _ _ _ _ #
# # _ _ _ # #
# _ # _ # _ #
# _ _ # _ _ #
# _ _ _ _ _ #
# _ _ _ _ _ #
# _ _ _ _ _ #
# _ _ _ _ _ #
# _ _ _ _ _ #
# _ _ _ _ _ #
# _ _ _ _ _ #
# _ _ _ _ _ #
```

Los criterios en la rúbrica (y el puntaje respectivo) se condicionan a que la solución presentada corresponda al problema planteado

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Ejecución	El diseño del algoritmo es ordenado y claro, siguiendo buenas prácticas en programación. La ejecución es correcta (`3pts`)	El diseño del algoritmo es ordenado y claro. La ejecución es correcta (`2pts`)	El diseño tiene algunas deficiencias pero la ejecución es correcta (`1pts`).	El diseño es deficiente y la ejecución no es correcta (`0.5pts`)
Sintaxis	No existen errores sintácticos o de compilación (`2pts`)	Existen algunos errores sintácticos de menor relevancia, que no afectan el resultado (`1.5pts`).	Existen errores sintácticos en la forma de ejecución, que no afectan el resultado (`1pts`).	El código tiene errores de sintáxis que afectan el resultado (`0.5pts`)
Optimizacion	El código es óptimo y eficiente. De buen performance e interacción con el usuario (`2pts`)	El código es de buen performance durante la ejecución (`1.5pts`)	El código no está optimizado pero la ejecución no es deficiente(`1pts`)	El código no está optimizado y la ejecución es deficiente (`0pts`)

2. (6 points) **Aproximación de Pi**
Una de las aproximaciones de Pi más fáciles de implementar es la división de 355/113.

Usted deberá implementar un programa en lenguaje C++ que solicite al usuario una cifra n, y almacene en un vector las n primeras cifras de 355/113

Importante:

- **Imprima todos los valores de su vector para poder visualizarlos.**
- **El último valor que se imprime es la cifra sin redondear. No debe redondear.**
- **La primera cifra que se imprime es el 3 de la parte entera. No se debe imprimir solo los decimales, sino las n cifras más significativas de su número.**
- **Asuma que el usuario siempre ingresa n positivo mayor o igual a 1.**
- **No es necesario que imprima el punto que marca el inicio de los decimales, solo imprima las cifras.**

Algunos ejemplos de diálogo de este programa serían:

Ejemplo de ejecución 1:

```
Ingrese n: 2
Las cifras de Pi son:
3
1
```

Ejemplo de ejecución 2:

```
Ingrese n: 6
Las cifras de Pi son:
3
1
4
1
5
9
```

Ejemplo de ejecución 7:

```
Ingrese n: 7
Las cifras de Pi son:
```

```
3
1
4
1
5
9
2
```

Los criterios en la rúbrica (y el puntaje respectivo) se condicionan a que la solución presentada corresponda al problema planteado

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Ejecución	El diseño del algoritmo es ordenado y claro, siguiendo buenas prácticas en programación. La ejecución es correcta (`3pts`)	El diseño del algoritmo es ordenado y claro. La ejecución es correcta (`2pts`)	El diseño tiene algunas deficiencias pero la ejecución es correcta (`1pts`).	El diseño es deficiente y la ejecución no es correcta (`0.5pts`)
Sintaxis	No existen errores sintácticos o de compilación (`2pts`)	Existen algunos errores sintácticos de menor relevancia, que no afectan el resultado (`1.5pts`).	Existen errores sintácticos en la forma de ejecución, que no afectan el resultado (`1pts`).	El código tiene errores de sintáxis que afectan el resultado (`0.5pts`)
Optimizacion	El código es óptimo y eficiente. De buen performance e interacción con el usuario (`2pts`)	El código es de buen performance durante la ejecución (`1.5pts`)	El código no está optimizado pero la ejecución no es deficiente(`1pts`)	El código no está optimizado y la ejecución es deficiente (`0pts`)

3. (7 points) **Banda de rock**
4 estudiantes de Programación 2 han decidido formar una banda de rock pero no tienen vocalista. Un(a) estudiante toca el bajo eléctrico, otro(a) estudiantes toca la guitarra rítmica, otro(a) estudiante toca la guitarra melódica o líder (lead guitar, en inglés), y otro(a)estudiante toca la batería.

Los estudiantes han decidido hacer un pequeño show en el auditorio de la universidad mostrando sus habilidades musicales, y así poder conseguir vocalista.

Se solicita:
Crear cuatro cuatro clases (una para cada instrumento), leer 1 objeto de cada una de las 4 clases creadas y tocar un pequeño concierto.

Los atributos de cada instrumento son:

- **Marca**
- **Modelo**
- **Año**
- **Sonido**

Cada clase deberá tener un constructor que cargue el contenido del atributo Sonido, según se detalla a continuación:

- **El bajo suena "bum bum bum bum"**
- **La guitarra rítmica suena "chan-chan chan-chan chan-chan chan-chan"**
- **La guitarra melódica suena "larala-larala larala-la larala-larala larala-la"**
- **La batería suena "chuku-pa chuku-pa chuku-pa chuku-pa"**

Lectura de datos:
Para poder ingresar los instrumentos al campus de la universidad, primero deberán ser registrados en la garita. Allí, el personal de seguridad solicitará los siguientes datos de registro:

- **Marca**
- **Modelo**
- **Año**

Para la elaboración de este programa, no es necesario que usted conozca de marcas y modelos de instrumentos, solo lea datos de tipo string para todas las marcas y modelos.

Concierto:
Para tocar el concierto, usted deberá implementar una función genérica que le permita imprimir el Sonido de un objeto de cualquiera de las 4 clases creadas. Para eso, usted deberá utilizar el concepto de Template.

Deberá llamar a esta función cuatro veces, una por cada instrumento. La función necesitará saber de qué clase se trata (por medio del template) para que dicha función pueda imprimir el Sonido correspondiente. Es muy importante que el método getSonido() se llame igual en las 4 clases.

Antes de cada llamado a la función, desde main, usted puede imprimir un mensaje indicando cuál es el instrumento que va a sonar. Usted debe decidir si la función genérica que va a implementar será de tipo void con un cout de getSonido() dentro de ella, o va a ser de tipo string con un return que tenga el valor de getSonido(). Esto queda a su criterio.

Al finalizar el concierto de rock, el público quedó satisfecho con el desempeño de los músicos, por lo que su programa deberá imprimir el mensaje "¡Qué buen concierto". No se preocupe si alguno de los caracteres no se imprime correctamente, solo agregue el cout correspondiente y asuma que se imprimirá bien.

Algunos ejemplos de diálogo de este programa serían:

Ejemplo de ejecución 1:

```
Datos del bajo:
Ingrese Marca: Yamaha
Ingrese modelo: BB435
Ingrese Año: 2022

Datos de la guitarra rítmica:
Ingrese Marca: Gibson
Ingrese modelo: SG
Ingrese Año: 1961

Datos de la guitarra melódica:
Ingrese Marca: Fender
Ingrese modelo: Stratocaster
Ingrese Año: 2001

Datos de la batería:
Ingrese Marca: Ludwig
Ingrese modelo: BSSS
```

```
Ingrese Año: 1999

Concierto de rock:

El bajo suena:
bum bum bum bum

La guitarra rítmica suena:
chan-chan chan-chan chan-chan chan-chan

La guitarra melódica suena:
larala-larala larala-la larala-larala larala-la

La batería suena:
chuku-pa chuku-pa chuku-pa chuku-pa

¡Qué buen concierto!
```

Los criterios en la rúbrica (y el puntaje respectivo) se condicionan a que la solución presentada corresponda al problema planteado

Criterio	Excelente	Adecuado	Mínimo	Insuficiente
Ejecución	El diseño del algoritmo es ordenado y claro, siguiendo buenas prácticas en programación. La ejecución es correcta (`3pts`)	El diseño del algoritmo es ordenado y claro. La ejecución es correcta (`2pts`)	El diseño tiene algunas deficiencias pero la ejecución es correcta (`1pts`).	El diseño es deficiente y la ejecución no es correcta (`0.5pts`)
Sintaxis	No existen errores sintácticos o de compilación (`2pts`)	Existen algunos errores sintácticos de menor relevancia, que no afectan el resultado (`1.5pts`).	Existen errores sintácticos en la forma de ejecución, que no afectan el resultado (`1pts`).	El código tiene errores de sintáxis que afectan el resultado (`0.5pts`)
Optimizacion	El código es óptimo y eficiente. De buen performance e interacción con el usuario (`2pts`)	El código es de buen performance durante la ejecución (`1.5pts`)	El código no está optimizado pero la ejecución no es deficiente(`1pts`)	El código no está optimizado y la ejecución es deficiente (`0pts`)