Realizar un programa en MIPS con las siguientes especificaciones:

Se dispone de un laberinto representado mediante una matriz cuadrada de caracteres (tiene el mismo número de filas que de columnas). El programa debe recorrer este laberinto desde el inicio (esquina superior izquierda) hasta el final (cualquier posición de la columna derecha) de forma automática.

```
0######
00##000
#0##0##
#0##0##
#0000##
#######
#######
```

El carácter ASCII correspondiente al decimal 35, # representa un muro
El carácter ASCII correspondiente al decimal 48, 0 representa el camino libre

El camino debe ser único, sin tener alternativas, y debe comenzar en la esquina superior izquierda y terminar en cualquier posición de la última columna.
Únicamente podremos movernos en horizontal o en vertical, nunca en diagonal.
Cuando se vaya recorriendo el camino las casillas visitadas deberán marcarse con el carácter ASCII correspondiente al decimal 88 que es una X.

Se deben realizar las siguientes funciones, siendo imprescindible comentar en las mismas los parámetros de entrada que se reciben y cuáles son sus salidas, detallando los registros utilizados:

1) Realizar una función **print_matriz** que reciba los siguientes argumentos:
    a. Dirección de una matriz de caracteres.
    b. Número de filas de la matriz (recuerda que el número de columnas es el mismo porque el laberinto es una matriz cuadrada)

    Esta función deberá imprimir por consola el laberinto que hayamos definido en la zona de datos estática y que te suministramos en el esqueleto de este programa. Para imprimir la matriz te recomendamos imprimirla carácter a carácter, utilizando la llamada al sistema 11. Por ejemplo, para imprimir el carácter # que hemos comentado anteriormente que corresponden con el número 35 en decimal debemos usar la siguiente secuencia de instrucciones:

    ```
    li $a0,35
    li $v0,11
    syscall
    ```

    Si por ejemplo queremos imprimir una salto de línea, que corresponde con el número 10 en decimal deberíamos hacer:

    ```
    li $a0,10
    li $v0,11
    syscall
    ```

2) Realiza una función llamada **paso** que reciba como argumento:
    a. Dirección de una matriz de caracteres.
    b. Número de filas de la matriz (recuerda que el número de columnas es el mismo porque el laberinto es una matriz cuadrada).
    c. Fila actual en donde está el laberinto (este número irá desde 1 al número total de filas de la matriz, lo que significa que la posición inicial que es la superior izquierda es la fila 1, columna 1).

d. Columna actual en donde está el laberinto (este número irá desde 1 al número total de columnas de la matriz, lo que significa que la posición inicial que es la superior izquierda es la fila 1, columna 1).

Y debe realizar las siguientes acciones:

1. Decidir qué movimiento debe realizar para seguir avanzando en el camino: Buscando la única casilla adyacente con valor 0 (recuerda que el número ASCII correspondiente a este carácter es el 48). Ten en cuenta, que si estás en una posición del borde no debes desplazarte fuera de él.
2. Modificar la matriz del laberinto, marcando con una X (el número ASCII correspondiente es el 88) la nueva posición recorrida.
3. Devolver dos números enteros que representen la fila y la columna de la nueva posición que se ha marcado (ambos enteros, la fila y la columna serán obviamente enteros que van desde uno al número de filas o columnas, y que corresponden a la nueva posición marcada)

NOTA: esta función no debe imprimir nada por la consola, ni solicitar ningún valor por consola, únicamente decidir el nuevo paso y modificar adecuadamente el laberinto.

3) Realizar una función llamada **recorre_camino**, encargada de ir recorriendo el camino del laberinto y calculando el coste de los desplazamientos. Recibirá los siguientes argumentos:

a. Dirección de una matriz de caracteres.
b. Número de filas de la matriz (recuerda que el número de columnas es el mismo porque el laberinto es una matriz cuadrada).
c. Coste de los movimientos en horizontal como flotante doble precisión.
d. Coste de los movimientos en vertical como flotante doble precisión.

Lo que debe realizar esta función es (comenzando en la posición inicial 1,1 esquina superior izquierda):

1. Marcar esta posición inicial con una X (el número ASCII correspondiente es el 88).
2. Dar un paso utilizando la función **paso** suministrándole de forma adecuada sus argumentos.
3. En función de la nueva posición (que es devuelta por la función **paso**) debes deducir si se ha tratado de un movimiento horizontal o vertical, añadiendo al coste total del camino el coste del movimiento

4. Repetir hasta que haya llegado al final del camino (que será una posición de la última columna).
5. Devolver el coste del recorrido

NOTA: esta función no debe imprimir nada por la consola, ni solicitar ningún valor por consola, únicamente iterar de forma adecuada usando la función paso para llegar desde el punto inicial hasta el final.

Debes además realizar el cuerpo principal del programa que hará lo siguiente:

- Preguntar por consola el coste que tiene cada tipo de movimiento (horizontal y vertical), expresado como un número flotante en doble precisión.
- Imprimir el laberinto por consola haciendo uso de la función **print_matriz**.
- Recorrer el laberinto hasta el final usando la función **recorre_camino**
- Mostrar el laberinto recorrido haciendo uso de la función **print_matriz**.
- Imprimir el coste total del recorrido como la suma de los costes de cada movimiento (recuerda que es el valor que devolverá recorre_camino).

Con este enunciado te suministramos además un esqueleto de programa “examen.s” que ya tiene algunas definiciones, como la matriz del laberinto y algunas constantes con nombres que te pueden realizar útiles. Además de un pantallazo con un posible ejemplo de salida de consola.

RÚBRICA DE CORRECCIÓN.

- El programa tiene como comentarios el pseudocódigo de cada función y cuerpo principal, especificando los registros que representan cada variable del mismo. **1 punto.**
- Se usan los registros estándar para las funciones y se respeta el convenido de buenas prácticas en el uso de los mismos. **1 punto**
- Se realiza de forma correcta las llamadas a las funciones, y se realizan las llamadas desde donde se indica en el enunciado. **1 punto**
- Se hace un uso correcto de la pila, almacenando y recuperando los valores necesarios para que las funciones ejecuten de forma autónoma sin tener que conocer qué registros usan las demás. **1 puntos**
- El programa cumple con las especificaciones:
    - Se realiza la función print_matriz. **1 punto**
    - Se realiza la función paso. **2 punto**
    - Se realiza la función recorre_camino. **2 punto**
    - Se realiza la función main con sus especificaciones. **1 puntos**

Todas las puntuaciones dependen de que el alumno asista a la defensa y sea capaz de explicar lo realizado en cada una de las valoraciones, pudiendo obtener fracciones de la puntuación en su caso. No asistir a la defensa, no saber defender lo realizado o no pasar el test antiplagio conllevará una valoración de cero en el examen.

CONSEJOS ÚTILES:

- Lee bien el enunciado, y pregunta las dudas sobre el enunciado. Debes hacer lo que se especifica y tener en cuenta los ítems de valoración, priorizando aquellos que te sumen puntos.
- Puedes empezar realizando la función **print_matriz** y la fución **paso**, comprobando en un main sencillo que se imprime el laberinto, y que tras realizar paso se ha encontrado el siguiente punto del camino.
- Las funciones puntúan por separado, por lo que te recomendados empezar por aquellas que tengas más claras y termines por las más complejas.
- Desarrolla el programa principal main para poder probar al menos las funciones que vayas desarrollando.