Práctica integradora de Recorridos

ATENCIÓN: Se recomienda realizar esta práctica íntegramente en papel, y recurrir a la computadora solamente cuando así lo solicite el enunciado.

Gobs-Man es un clon argentino de un popular juego de arcade de principios de los 80, que está realizado en Gobstones.

En los siguientes ejercicios implementaremos diversos procedimientos y funciones que trabajan distintos aspectos del juego (no implementaremos el juego en sí, sino procedimientos y funciones que podrían servir para el mismo).

Gobs-Man es una criatura que vive en un tablero del cual desconocemos su ancho y su alto, el cual varía de un nivel a otro. Gobs-Man puede ser programado para moverse de una celda a otra del tablero, para lo cual podemos utilizar las siguientes primitivas:

```
procedure MoverGobsManAl_(dirección)
PROPÓSITO: Mueve a Gobs-Man a la celda vecina en la dirección "dirección". El cabezal queda sobre Gobs-Man.
PRECONDICIÓNES:
  * Existe una celda vecina en dirección "dirección".
  * El cabezal se encuentra sobre Gobs-Man.
PARÁMETROS:
  * dirección: Dirección - Indica hacia dónde se moverá Gobs-Man.
```

```
procedure LlevarGobsManAlBorde_(dirección)
PROPÓSITO: Mueve a Gobs-Man a la celda en el borde hacia la dirección "dirección". El cabezal queda sobre Gobs-Man.
PRECONDICIÓNES:
  * El cabezal se encuentra sobre Gobs-Man.
PARÁMETROS:
  * dirección: Dirección - Indica hacia dónde se moverá Gobs-Man.
```

A medida que avancemos en los distintos ejercicios iremos introduciendo las reglas puntuales del juego, y nuevas primitivas que podrán utilizarse junto con las anteriormente mencionadas para solucionar el problema.

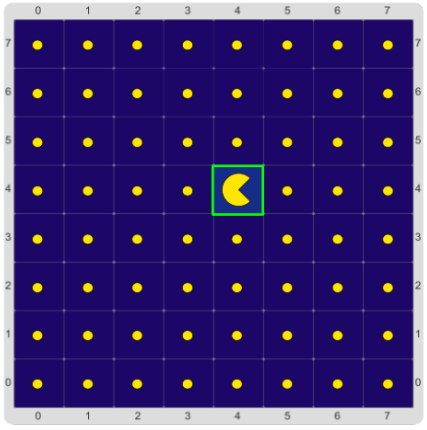
ATENCIÓN: No tiene que implementar las primitivas salvo que así lo indique el enunciado, pero si puede hacer uso de ellas para solucionar todos los problemas planteados.

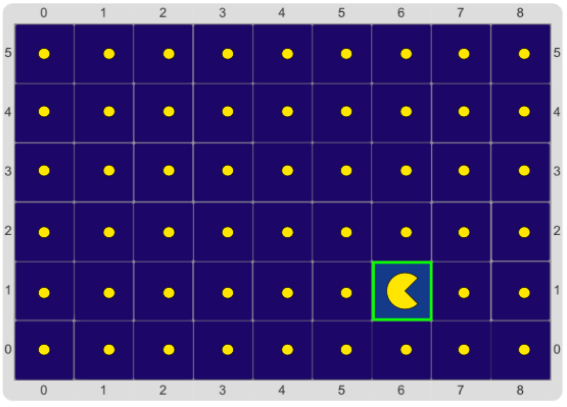
Ejercicio 1)
Al comenzar un nuevo "nivel" del juego, en cada celda del tablero hay un "coco" (pequeños puntos amarillos) que son el alimento natural de los seres como Gobs-Man. El objetivo de Gobs-Man es precisamente comerse

todos los cocos del nivel. Para poder hacer esto, contamos con la siguiente primitiva adicional a las anteriores:

procedure ComerCoco()
PROPÓSITO: Come el coco que haya en la celda actual.
PRECONDICIONES:
* Hay un coco en la celda actual.
* Gobs-Man está en la celda actual.

Se desea implementar el procedimiento **ComerTodosLosCocosDelNivel**(), que hace que Gobs-Man se coma absolutamente todos los cocos del nivel (tablero). Sabemos, *por precondición de dicho procedimiento, que hay un coco en cada celda del tablero* (incluida en la que inicia Gobs-Man). A continuación hay algunos posibles niveles de Gobs-Man (Note que son solo ejemplos, y que su solución tiene que funcionar en cualquiera de estos tableros, e incluso otros que cumplan las mismas características)



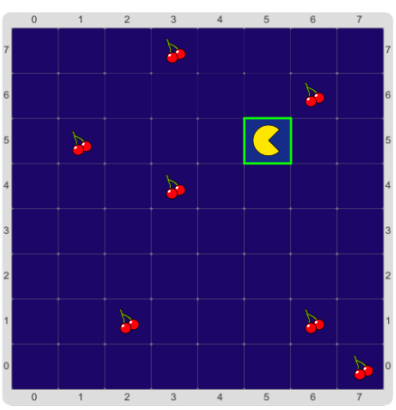
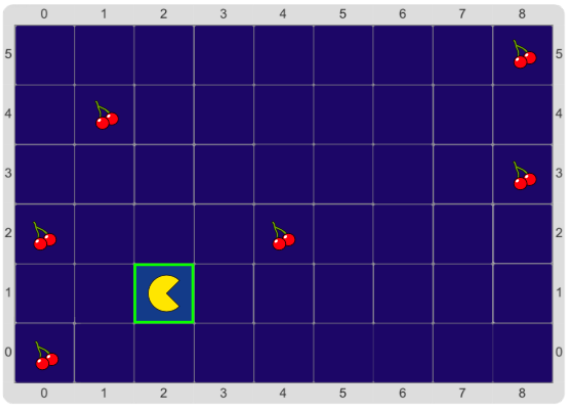


Ejercicio 2)
Gobs-Man también gusta de comer cerezas. En este caso queremos que Gobs-Man se coma absolutamente todas las cerezas del nivel. Ojo, a diferencia de los cocos, las cerezas no están en todas las celdas, sino que pueden aparecer en algunas celdas y en otras no, y nunca sabemos al arrancar un nivel en cuáles celdas estarán las cerezas. Para implementar esto necesitaremos unas nuevas primitivas:

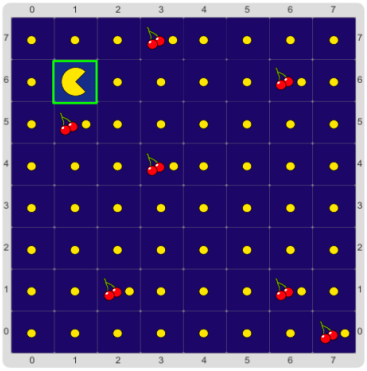
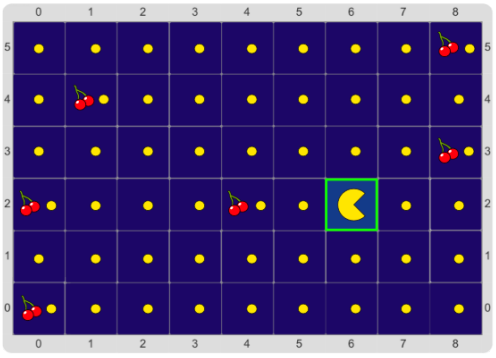
procedure ComerCereza()
PROPÓSITO: Come la cereza haya en la celda actual.
PRECONDICIONES:
* Hay una cereza en la celda actual.
* Gobs-Man está en la celda actual.

function hayCereza()
PROPÓSITO: Indica cuando hay una cereza en la celda actual.
PRECONDICIÓN: Ninguna.

Ahora si, implementemos el procedimiento **ComerTodasLasCerezasDelNivel**(), que hace que Gobs-Man se coma todas las cerezas del tablero. Note que los tableros iniciales posibles de este ejercicio no tienen cocos, sino que en cada celda puede haber una cereza, o no haber nada, como muestran los ejemplos de tableros iniciales a continuación:

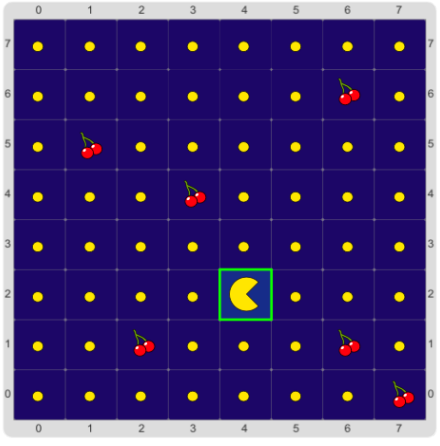
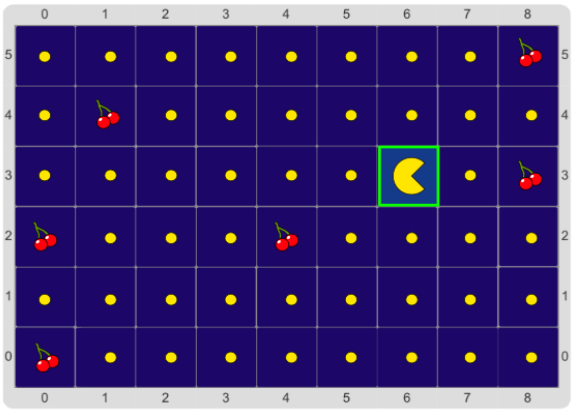



Ejercicio 3)
Para este ejercicio queremos trabajar sobre niveles que tienen tanto cerezas como cocos. En nuestros tableros iniciales habrá cocos en todas las celdas, y en algunas habrá adicionalmente una cereza. Gobs-Man quiere comerse absolutamente todo lo que encuentre en el nivel, y nosotros debemos indicarle como en el código de **ComerTodoLoQueSeEncuentreEnElNivel**().

Pista: Si su solución requiere más de dos líneas de código, plantee en otra estrategia (piense en qué cosas ya realizó anteriormente)

Ejercicio 4)
El programador del juego ha decidido hacer unos pequeños retoques a cómo inician los niveles. Ahora, en todas las celdas hay cocos, menos en aquellas donde hay una cereza. Es decir, en cada celda puede, o bien haber una cereza, o bien haber un coco (solo estará vacía la celda cuando Gobs-Man se haya comido todo lo de la celda, nunca al inicio del nivel). Debemos entonces volver a realizar **ComerTodoLoQueSeEncuentreEnElNivel**(), y en este caso, la solución no es tan sencilla como en el ejercicio anterior. A continuación, algunos posibles tableros iniciales de este caso:

ATENCIÓN: Note que no dispone de una primitiva "hayCoco" para solucionar el problema. Si su estrategia está realizada utilizando dicha primitiva, entonces es incorrecta.

ATENCIÓN: Si a esta altura sus soluciones le están demandando el planteo de más de un recorrido, probablemente esté planteando recorridos sobre filas o columnas. Le proponemos plantee la solución en términos de un recorrido único sobre todas las celdas del tablero, o los siguientes ejercicios se volverán sumamente complicados.

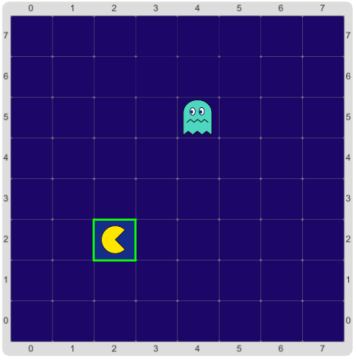
Ejercicio 5)
Gobs-Man puede toparse en algún momento con un fantasma. Si lo hace, Gobs-Man sufre un paro cardíaco que la hace morir en la celda en donde vió el espectro. Sabiendo que existen ahora las siguientes primitivas:

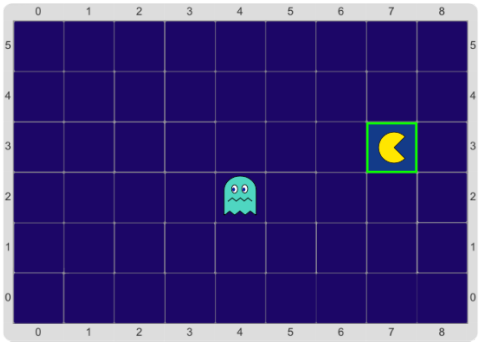
```
procedure MorirGobsMan()
PROPÓSITO: Hace que Gobs-Man muera, dejando su cuerpo en la celda actual.
PRECONDICIONES:
  * El cabezal se encuentra sobre Gobs-Man
```

```
function hayFantasma()
PROPÓSITO: Indica cuando hay un fantasma en la celda actual.
PRECONDICIÓN: Ninguna.
```

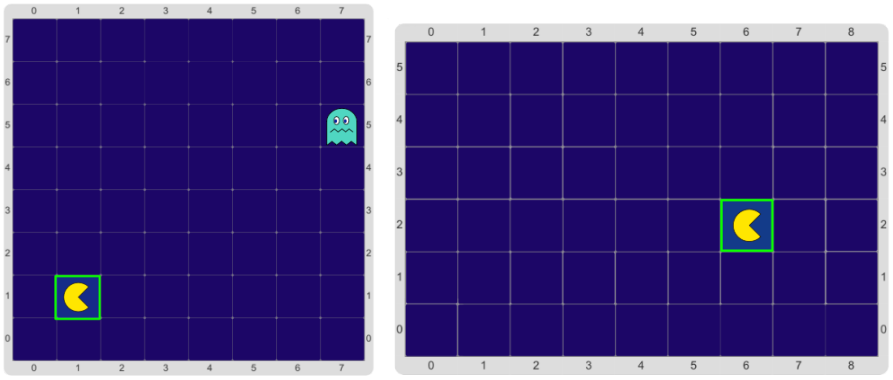
ATENCIÓN: Note que una vez muerto Gobs-Man no puede moverse. Es decir, los procedimientos que mueven a Gobs-Man tienen ahora una nueva precondición: Gobs-Man está vivo.

Se pide entonces hagamos una prueba sobre un nivel vacío (Es decir, en las celdas no hay cocos ni cerezas) donde Gobs-Man deberá moverse desde la celda más al Oeste y al Sur, hacia la celda más al Norte y al Este. Se garantiza que en algún lado del tablero habrá un fantasma, y Gob-Man debe morir en la celda en donde encuentre el mismo. Realice entonces el procedimiento **RecorrerNivelMuriendoEnElFantasma().** Como es costumbre, dejamos algunos tableros iniciales:

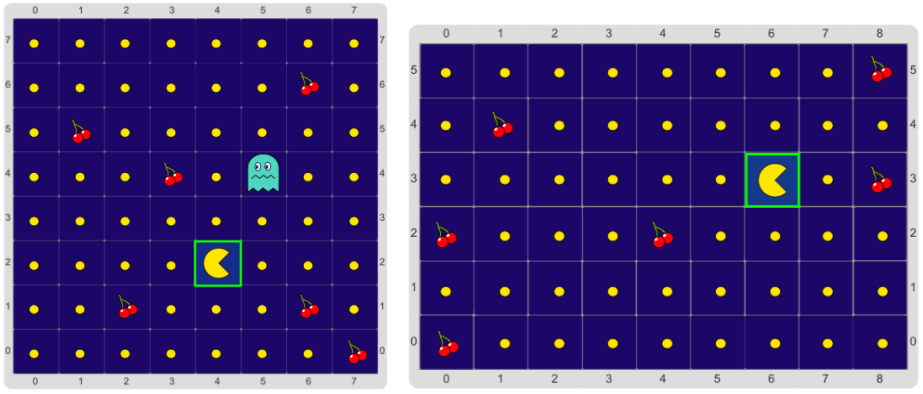




Ejercicio 6)
Si bien hemos logrado que Gobs-Man muera en el lugar correcto, también se desea contemplar los niveles en donde tal vez no haya un fantasma. Es decir, ahora queremos volver a recorrer el nivel, pero esta vez, no tenemos la certeza de que hay un fantasma en el nivel. Si hay uno, Gobs-Man deberá morir allí, sino, Gobs-Man deberá quedar vivo en la última celda del recorrido. Realice entonces el procedimiento **RecorrerNivelMuriendoSiHayFantasma()** que solucione dicho problema. Los tableros iniciales son idénticos a los anteriores, pero, el fantasma podría no estar, como muestra el segundo tablero de ejemplo:



Ejercicio 7)
En este ejercicio queremos integrar todas nuestras soluciones al momento. Aunque probablemente no podamos reutilizar el código, si podremos reutilizar las ideas de lo que venimos trabajando.
En este caso, el nivel comienza con un coco en cada celda, menos en las que hay cerezas, y tal vez, algún fantasma en alguna celda del tablero. Gobs-Man debe comer todos los cocos y cerezas que pueda, partiendo esta vez de la esquina Norte y Oeste, y yendo hacia el Sur y el Este. Al finalizar el nivel, Gobs-Man debe quedar en dicha esquina, si es que no se cruzó con ningún fantasma. Si por el contrario el nivel tiene un fantasma, Gobs-Man deberá comer todo lo que tenga en el camino, hasta que se tope con el espectro, donde morirá y terminará el juego. Implemente entonces **JugarNivel**() que realice lo mencionado.



Ejercicio 8)
Que pasa si quiero que Gobs-Man ahora recorra en sentido inverso, es decir, partiendo en la esquina Sur y Este, y yendo hacia el Norte y el Oeste. ¿Su solución permite cambiar fácilmente ese detalle? Si la respuesta es negativa, piense alguna forma de lograrlo. Pista, el truco está en pasar información a los procedimientos que realizan el recorrido celda a celda.

Ejercicio 9)
Queremos realizar el juego, y probar que funcionen nuestras soluciones, pero el diseñador gráfico ha renunciado y no tenemos vestimentas ni primitivas que nos abstraiga de la representación, debemos contentarnos con ver bolitas. Por suerte, todas nuestras soluciones anteriores asumen la existencia de ciertos procedimientos y funciones primitivas, por lo que bastará implementar las mismas para tener andando nuestro trabajo previo. Asumiremos la siguiente representación:

- Gobs-Man estará representado por una bolita de color Azul si está vivo, y dos, si está muerto.
- Un coco estará representado por una bolita de color Negro.
- Una cereza estará representada por dos bolitas de color Rojo.
- Un fantasma estará representado por cinco bolitas de color Verde.

Se pide entonces implemente cada uno de los procedimientos y funciones primitivas mencionados en esta guía utilizando esta representación, y luego pruebe las soluciones que hicimos en papel, en la máquina.