



# LOGICA TAREAS

## TAREA 1 DIETA A BASE DE CHURRASCOS DESAFIO 1 (AYUDA A LA PUMA A COMER SU CHURRASCO EVITA LOS OBSTACULOS)

### AL EMPEZAR A EJECUTAR

| ALIMENTAR A LA PUMA

### DEFINIR ALIMENTAR A LA PUMA

| MOVER A LA DERECHA

| COMER CHURRASCO

### DEFINIR MOVER 3 A LA DERECHA

| REPETIR 3 VECES

| MOVER A LA DERECHA

## TAREA 1 DIETA A BASE DE CHURRASCOS DESAFIO 2 (DUBA QUIERE COMER SU CHURRASCO. ¡AYUDALA!)

### AL EMPEZAR A EJECUTAR

| ALIMENTAR A LA PUMA

### DEFINIR ALIMENTAR A LA PUMA

| MOVER A LA DERECHA

| MOVER ARRIBA

| MOVER A LA DERECHA

| COMER CHURRASCO

## TAREA 1 DIETA A BASE DE CHURRASCOS DESAFIO 3 (¿AYUDÁS A LA PUMA A COMER SU CHURRASCO? EVITÁ LOS OBSTACULOS)

### AL EMPEZAR A EJECUTAR

| ALIMENTAR A LA PUMA

### DEFINIR ALIMENTAR CHURRASCO

| MOVER ABAJO

| MOVER 3 A LA DERECHA

| COMER CHURRASCO

## TAREA 1 DIETA A BASE DE CHURRASCOS DESAFIO 4 (DUBA QUIERE COMER CHURRASCO ¿Cómo LO PUEDE HAER SIN CHICASRE CON LOS OBSTACULOS?)

### AL EMPEZAR A EJECUTAR

| ALIMENTAR A LA PUMA

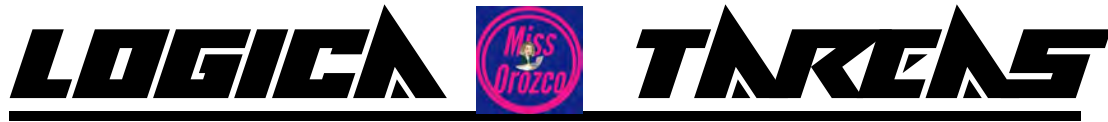
### DEFINIR ALIMENTAR A LA PUMA

| BAJAR 2 CASILLEROS

| MOVER A LA DERECHA

| MOVER ABAJO

| COMER CHURRASCO



$\wedge = Y$   $\neg = \text{NO}$   $V = O$

**DEFINIR BAJAR 2 CASILLEROS**

| REPETIR 2 VECES

| MOVER ABAJO

**TAREA 1 DIETA A BASE DE CHURRASCOS DESAFIO 5 (AYUDAR A LA PUMA DUBA A SACIAR SU HAMBRE EVITANDO LOS OBSTACULOS)**

**AL EMPEZAR A EJECUTAR**

| SACIAR HAMBRE

**DEFINIR SACIAR HAMBRE**

| MOVER 2 A LA DERECHA

| BAJAR 2

| COMER CHURRASCO

**DEFINIR MOVER 2 A LA DERECHA**

| REPETIR 2 VECES

| MOVER ABAJO

**DEFINIR BAJAR 2**

| REPETIR 2 VECES

| MOVER ABAJO

**TAREA 1 DIETA A BASE DE CHURRASCOS DESAFIO 6 (DUBA QUIERE DEVORAR SU CHURRASCO)**

**AL EMPEZAR A EJECUTAR**

| SUBIR 2 CASILLEROS

| MOVER 2 A LA DERECHA

| MOVER ABAJO

| COMER CHURRASCO

**DEFINIR SUBIR 2 CASILLEROS**

| REPETIR 2 VECES

| MOVER ARRIBA

**TAREA 2 EL ALIEN TOCA EL BOTON**

**AL EMPEZAR A EJECUTAR**

| TOCAR EL BOTON

**DEFINIR TOCAR EL BOTON**

| MOVER 3 A LA DERECHA

| APRETAR BOTON

**DEFINIR MOVER 3 A LA DERECHA**

| REPETIR 3 VECES

| MOVER A LA DERECHA

## TAREA 3 Ejercicio 1 LOGO LA TORTUGA

### PRIMITIVAS

#### Dar un paso

Hace que Logo se mueva un paso hacia la dirección en la que está mirando. Sí el pincel está en el piso, entonces dibuja en esa línea, si no, entonces la línea no queda pintada.

#### Girar a derecha

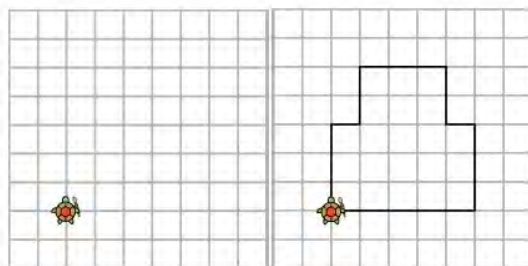
Hace que Logo gire en sentido horario exactamente 90°, cambiando la dirección en la que está mirando.

#### Bajar pincel

Hace que Logo baje el pincel, apoyándolo en el suelo. Sí el pincel estaba ya en el suelo, entonces no hace nada (pero no falla).

#### Levantar pincel

Hace que Logo levante el pincel, sacándolo del suelo. Sí ya estaba levantado, entonces no hace nada (pero no falla).



### PROCEDIMIENTOS

a) Escribir el procedimiento **Girar media vuelta** que gire a Logo 180° (es decir, quede mirando en el sentido opuesto al que se encontraba).

#### DEFINIR GIRAR MEDIA VUELTA

| GIRAR A DERECHA

| GIRAR A DERECHA

b) Escribir el procedimiento **Girar a la izquierda** que gire a Logo en sentido antihorario exactamente 90°, cambiando la dirección a la que está mirando.

#### DEFINIR GIRAR A LA IZQUIERDA

| GIRAR A DERECHA

| GIRAR A DERECHA

| GIRAR A DERECHA

c) Escribir el procedimiento **Dar un paso sin pintar** que da un paso sin pintar la línea por la que camina Logo.

#### DEFINIR DAR UN PASO SIN PINTAR

| LEVANTAR PINCEL

| DAR UN PASO

d) Escribir el procedimiento **Dar un paso pintando** que da un paso pintando la línea por la que camina Logo.

#### DEFINIR DAR UN PASO PINTADO

| BAJAR PINCEL

| DAR UN PASO

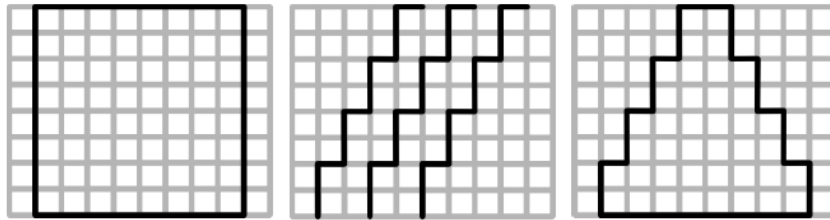
## TAREA 3 Ejercicio 2. Logo empieza a pintar

$\wedge = Y$   $\neg = \text{NO}$   $V = O$

## DIBUJO 1

### AL EMPEZAR

| F



### DEFINIR F

| REPETIR 3 VECES

| PINTAR LADO DEL CUADRADO

| GIRAR A DERECHA

| PINTAR UN LADO DEL CUADRADO

### DEFINIR PINTAR LADO DEL CUADRADO

| REPETIR 8 VECES

| DAR UN PASO PINTADO

## DIBUJO 2 ESCALERA

### AL EMPEZAR A EJECUTAR

| DIBUJAR 3 ESCALERAS

### DEFINIR DIBUJAR 3 ESCALERAS

| REPETIR 3 VECES

| SUBIR 1 ESCALERA

### DEFINIR SUBIR 1 ESCALERA

| SUBIR 4 ESCALONES

| IR AL PUNTO DE INICIO

### DEFINIR SUBIR 4 ESCALONES

| REPETIR 4 VECES

| SUBIR UN ESCALON

### DEFINIR SUBIR UN ESCALON

| DAR UN PASO PINTANDO

| DAR UN PASO PINTANDO

| GIRAR A DERECHA

| DAR UN PASO

### DEFINIR IR AL PUNTO DE INICIO

| GIRAR A DERECHA

| BAJAR 8 ESPACIOS SIN PINTAR

| GIRAR DERECHA

| CAMINAR 2 ESPACIOS SIN PINTAR

| GIRAR A DERECHA

$\wedge = Y$   $\neg = \text{NO}$   $V = O$

**DEFINIR CAMINAR 2 ESPACIOS SIN PINTAR**

| REPETIR 2 VECES

| DAR UN PASO SIN PINTAR

**DIBUJO 2 TORRE****AL EMPEZAR A EJECUTAR**

| DIBUJAR TORRE

**DEFINIR DIBUJAR TORRE**

| SUBIR 1 ESCALERA

| BAJAR 1 ESCALERA

| GIRAR A DERECHA

| PINTAR LADO DEL CUADRADO

**DEFINIR BAJAR 1 ESCALERA**

| REPETIR 4 VECES

BAJAR UN ESCALON

**DEFINIR BAJAR UN ESCALON**

| DAR UN PASO PINTADO

| GIRAR A DERECHA

| DAR 2 PASOS PINTANDO

| GIRAR IZQUIERDA

**DEFINIR DAR 2 PASOS PINTANDO**

| REPETIR 2 VECES

| DAR UN PASO

**TAREA 3 Ejercicio 3. COCINERO DE HIERRO****PRIMITIVAS**

**Tomar ingrediente 1**

**Tomar ingrediente 2**

**Tomar ingrediente 3**

**Tomar ingrediente 4**

**Tomar ingrediente 5**

Coloca el elemento en la posición mencionada de la lista de ingredientes y lo coloca en la tabla. Falla sí ya hay un ingrediente en la tabla.

**Trocear**

Troza el ingrediente en la tabla. Falla sí el ingrediente fue picado, rallado o troceado previamente.

**Picar**

Pica el ingrediente en la tabla. Falla sí el ingrediente fue troceado, picado o rallado previamente.

**Rallar**

$\wedge = Y$   $\neg = \text{NO}$   $V = O$

Ralla el ingrediente en la tabla. Falla sí el ingrediente fue troceado, picado o rallado previamente.

**Colocar en wok**

Coloca el ingrediente en la tabla en el wok. Falla sí no hay ingredientes en la tabla.

**Revolver wok**

Revuelve los ingredientes del wok.

**Sacudir wok**

Sacude el wok.

**Servir**

Sirve los contenidos del wok a los comensales

**a) Chop Suey**

Ingredientes:

1. Cebolla
2. Zanahoria
3. Bife de lomo
4. Pimiento rojo
5. Arroz

Pasos:

1. Picar la cebolla y el pimiento rojo y colocar en el wok. Revolver dos veces el wok para mezclar bien los ingredientes.
2. Rallar la zanahoria y colocar en el wok.
3. Trocear el bife de lomo y colocar en el wok. Sacudir el wok dos veces.
4. Colocar el arroz y sacudir el wok dos veces más.
5. Servir el contenido del wok.

**AL EMPEZAR EJECUTAR**

|SERVIR RECETA DE CHOP SUEY

**DEFINIR SERVIR RECETA DE CHOP SUEY**

|AGREGAR CEBOLLA Y PIMIENTO

|AGREGAR ZANAHORIA

|AGREGAR BIFE

|AGREGAR ARROZ

|SERVIR

**DEFINIR AGREGAR CEBOLLA Y PIMIENTO**

|COLOCAR CEBOLLA PICADA

|COLOCAR PIMIENTO PICADO

|REVOLVER WOK 2 VECES

**DEFINIR AGREGAR ZANAHORIA**

|TOMAR INGREDIENTE 2

|RALLAR

$\wedge = Y$   $\neg = \text{NO}$   $V = O$

| COLOCAR EN WOK

#### DEFINIR REVOLVER WOK 2 VECES

| REPETIR 2 VECES

| REVOLVER WOK

#### DEFINIR AGREGAR BIFE

| TOMAR INGREDIENTE 5

| COLOCAR EN WOK

| REPETIR 2 VECES

| SACUDIR WOK

#### DEFINIR AGREGAR ARROZ

| COLOCAR ARROZ EN WOK

| SACUDIR WOK 2 VECES

#### DEFINIR CEBOLLA PICADA

| TOMAR INGREDIENTE 1

| PICAR

| COLOCAR EN WOK

#### DEFINIR COLOCAR PIMIENTO PICADO

| TOMAR INGREDIENTE 4

| PICAR

| COLOCAR EN WOK

#### DEFINIR SACUDIR WOK 2 VECES

| REPETIR 2 VECES

| SACUDIR WOK

#### DEFINIR COLOCAR ARROZ EN WOK

| TOMAR INGREDIENTE 5

| PICAR

| COLOCAR EN WOK

#### TAREA 4 Chow mien

Ingredientes:

1. Cebolla
2. Zanahoria
3. Pollo
4. Pimiento rojo
5. Fideos

Pasos:

1. Picar la cebolla y el pimiento rojo y colocar en el wok. Revolver dos veces el wok para mezclar

bien los ingredientes.

2. Rallar la zanahoria y colocar en el wok.
3. Trocear el pollo. Sacudir el wok dos veces.
4. Colocar los fideos y revolver el wok.
5. Sacudir el wok dos veces.
6. Servir el contenido del wok

#### AL EMPEZAR A EJECUTAR

| SERVIR RECETA DE CHOW MIEN

#### DEFINIR SERVIR RECETA DE CHOW MIEN

| AGREGAR CEBOLLA Y PIMIENTO

| AGREGAR ZANAHORIA

| AGREGAR POLLO PICADO

| AGREGAR FIDEOS

| SACUDIR WOK 2 VECES

| SERVIR

#### DEFINIR AGREGAR POLLO PICADO

| TOMAR INGREDIENTE 3

| TROCEAR

| SACUDIR WOK 2 VECES

#### DEFINIR AGREGAR FIDEOS

| TOMAR INGREDIENTE 5

| COLOCAR EN WOK

| REVOLVER WOK

### TAREA 5 REPETICIONES SIMPLES Y ESTRATEGIA DE SOLUCIÓN

#### EJERCICIO 1) LA ESTACIÓN FORESTAL

##### ***Dar un paso a la derecha***

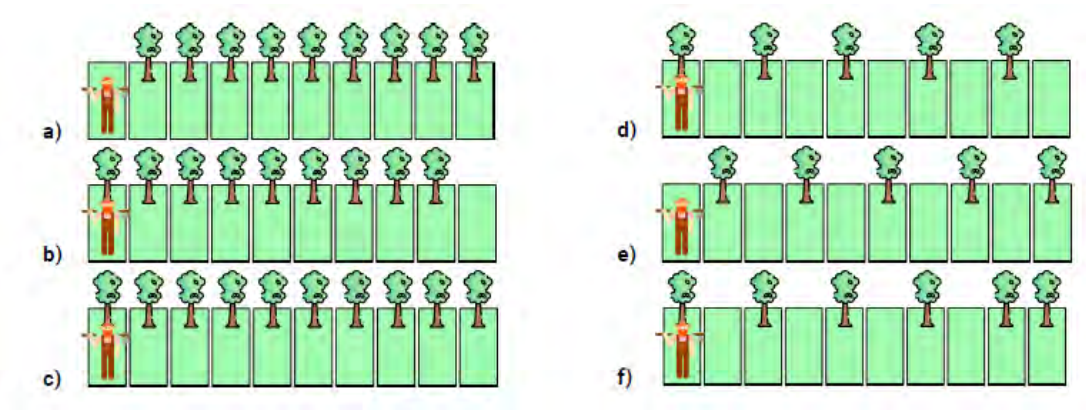
Mueve al leñador una ubicación a la derecha. Falla sí no hay más ubicaciones a la derecha.

##### ***Talar eucalipto***

Tala el eucalipto de la ubicación donde está el leñador. Falla sí el leñador no está sobre un eucalipto.



$\wedge=Y \rightarrow \text{NO } V=O$



## A) AL EMPEZAR A EJECUTAR

| TALAR TODOS LOS EUCALIPTOS

## DEFINIR TALAR TODOS LOS EUCALIPTOS DEL ESCENARIO A

| REPETIR 9 VECES

| CAMINAR Y TALAR SI HAY EUCALIPTO

## DEFINIR CAMINAR Y TALAR SI HAY EUCALIPTO

| DAR UN PASO A LA DERECHA

| TALAR SI HAY EUCALIPTO

## DEFINIR TALAR SI HAY EUCALIPTO

| SI ¿HAY ARBOLES? ENTONCES

| TALAR EUCALIPTO

## A) AL EMPEZAR A EJECUTAR

| TALAR TODOS LOS EUCALIPTOS DEL ESCENARIO B

## DEFINIR TALAR TODOS LOS EUCALIPTOS DEL ESCENARIO B

| TALAR

| REPETIR 8 VECES

| CAMINAR Y TALAR SI HAY EUCALIPTO

## B) AL EMPEZAR A EJECUTAR

| TALAR TODOS LOS EUCALIPTOS DEL ESCENARIO C

## DEFINIR TALAR TODOS LOS EUCALIPTOS DEL ESCENARIO C

| TALAR

| REPETIR 9 VECES

| CAMINAR Y TALAR SI HAY EUCALIPTO

## C) AL EMPEZAR A EJECUTAR

| TALAR TODOS LOS EUCALIPTOS DEL ESCENARIO D

## DEFINIR TALAR TODOS LOS EUCALIPTOS DEL ESCENARIO D

$\wedge = Y$   $\neg = \text{NO}$   $\vee = \text{O}$

- | TALAR
  - | REPETIR 9 VECES
  - | CAMINAR Y TALAR SI HAY EUCALIPTO
- D) AL EMPEZAR A EJECUTAR**
  - | TALAR TODOS LOS EUCALIPTOS DEL ESCENARIO E
  - | REPETIR 9 VECES
  - | CAMINAR Y TALAR SI HAY EUCALIPTO
- E) AL EMPEZAR A EJECUTAR**
  - | TALAR TODOS LOS EUCALIPTOS DEL ESCENARIO F
- F) DEFINIR TALAR TODOS LOS EUCALIPTOS DEL ESCENARIO F**
  - | TALAR
  - | REPETIR 9 VECES
  - | CAMINAR Y TALAR SI HAY EUCALIPTO

## EJERCICIO 2 BURGUERBOT

### Al empezar a ejecutar

- | Servir todos los combos

### Definir Servir todos los combos

- | Repetir 4 veces
  - | Servir combo de la fila actual
  - | Ir al inicio de la siguiente fila
- | Servir combo de la fila actual

### Definir Servir combo de la fila actual

- | Armar combo actual
- | Servir combo actual

### Definir Armar combo actual

- | Acercarse un paso al mostrador
- | Recoger gaseosa
- | Acercarse un paso al mostrador
- | Recoger papas
- | Acercarse un paso al mostrador
- | Recoger hamburguesa

### Definir Servir combo actual

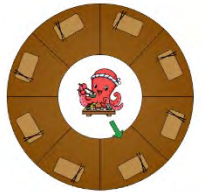
- | Acercarse un paso al mostrador
- | Servir combo

## Definir Ir al inicio de la siguiente fila

- | Mover un paso a la derecha
- | Volver al fondo de la cocina

### Ejercicio 3) El sushi de Jiro

Jiro es un muy reconocido sushiman, que prepara uno de los mejores sushis del planeta. Pero no es tanto el sushi y su saber, sino el espectáculo de la preparación en vivo que Jiro realiza lo que atrae a sus comensales. Jiro es tan famoso que recibe por cena solamente a ocho comensales a la vez, uno por cada tentáculo que tiene, y estos se sientan en una mesa circular, con un espacio en el medio, donde Jiro se pone en el medio, preparando y sirviendo el sushi a los diversos comensales. Jiro sirve el sushi en rondas (así como en otros restaurantes se sirve entrada, primer plato, segundo plato, etc.). En cada ronda Jiro sirve una o varias piezas de un tipo de sushi particular, y sirve las rondas a todos los comensales al mismo tiempo. En la primera ronda sirve a cada uno de los comensales dos niguiris de salmón. En la segunda ronda sirve a cada comensal seis rolls, tres de cangrejo y tres de atún. Como tercer ronda se sirven los sashimis, tres de salmón y cinco de atún rojo, a cada uno. Por último se sirve el tamago a cada comensal, una unidad. A pesar de que haya un comensal por tentáculo, Jiro solo puede atender y mirar a uno a la vez. Por lo tanto, va girando y mirando a cada comensal a medida que le va sirviendo las piezas de sushi que le corresponde. El dibujo a continuación muestra un ejemplo de Jiro atendiendo a sus comensales, donde la flecha verde indica a quién se encuentra atendiendo en ese momento.



Lo que se pide es que realice un programa que haga que Jiro sirva a los comensales de la mesa las diversas piezas de sushi, según las rondas antes descritas. Para esto, se brindan las siguientes primitivas:

#### **Girar en sentido horario**

Hace que Jiro rote en sentido horario y quede mirando al siguiente comensal.

#### **Girar en sentido antihorario**

Hace que Jiro rote en sentido antihorario y quede mirando al siguiente comensal.

#### **Servir roll de cangrejo**

Sirve una porción de roll de cangrejo al comensal al que actualmente está mirando Jiro.

#### **Servir roll de atún**

Sirve una porción de roll de atún al comensal al que actualmente está mirando Jiro.

#### **Servir sashimi de salmón**

Sirve una porción de sashimi de salmón al comensal al que actualmente está mirando Jiro.

#### **Servir sashimi de atún rojo**

Sirve una porción de sashimi de atún rojo al comensal al que actualmente está mirando Jiro.

#### **Servir nigui de salmón**

Sirve una porción de nigui de salmón al comensal al

$\wedge=Y \neg=NO \vee=O$

que actualmente está mirando Jiro.

## Servir tamago

Sirve una porción de tamago al comensal al que actualmente está mirando Jiro.

Una vez realizada la solución, se pide que conteste.

- ¿De qué forma representa las rondas de comida?
- ¿Fue necesario usar la totalidad de los comandos primitivos disponibles?
- Sí la respuesta es no, Se le ocurre otra solución equivalente a la que realizó, pero usando dicha primitiva

**FALTA TERMINAR**

## Ejercicio 4) Don Barredora

Los inviernos en Ushuaia son largos y difíciles para los vecinos, pues la ciudad se llena de nieve y es imposible salir de casa. Es también una temporada muy lucrativa para Don Barredora, el dueño de la única máquina quitanieve de la ciudad, que se encarga de sacar la nieve de las entradas de los vecinos. Por supuesto que no es fácil, porque la ciudad no es uniforme. Algunas calles acumulan más nieve que otras. Otras calles son más largas, y otras más cortas. Puede apreciarse el escenario a continuación.

Para manipular a Don Barredora están las siguientes primitivas:

**Mover barredora arriba** Mueve la barredora una ubicación hacia arriba. Falla sí no hay más ubicaciones hacia arriba.

**Mover barredora abajo** Mueve la barredora una ubicación hacia abajo. Falla sí no hay más ubicaciones hacia abajo.

**Mover barredora izquierda** Mueve la barredora una ubicación hacia la izquierda.

Falla sí no hay más ubicaciones hacia la izquierda.

**Mover barredora derecha** Mueve la barredora una ubicación hacia la derecha. Falla sí no hay más ubicaciones hacia la derecha.

**Quitar nieve** Quita la nieve de la ubicación actual. Falla sí no hay nieve en la ubicación actual.

## AL EMPEZAR A EJECUTAR

| QUITAR TODA LA NIEVE DE LA CIUDAD

## DEFINIR QUITAR TODA LA NIEVE DE LA CIUDAD

| QUITAR NIEVE DE 3 CALLES

| IR AL INICIO DE LA CUADRA SIGUIENTE

| QUITAR NIEVE DE 3 CALLES

## DEFINIR QUITAR NIEVE DE 3 CALLES

| REPETIR 3 VECES

| QUITAR NIEVE DE 3 CASAS

| IR AL INICIO DE LA CUADRA SIGUIENTE

## DEFINIR QUITAR NIEVE DE 3 CASAS

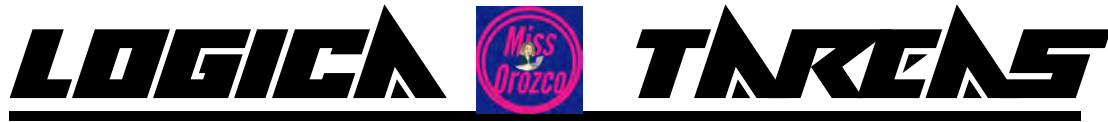
| REPETIR 3 VECES



¿DÓNDE ESTA LA BARREDORA HAY NIEVE?

XQ SI NO HAY FALLA!

Y DEBERIA SER



$\wedge=Y \neg= \text{NO } V=0$

| MOVER BARREDORA DERECHA  
| QUITAR NIEVE



**DEFINIR IR AL INICIO DE LA CUADRA SIGUIENTE**

| MOVER BARREDORA ABAJO  
| MOVER BARREDORA 3 CASAS A LA IZQUIERDA

**DEFINIR MOVER BARREDORA 3 CASA A LA IZQUIERDA**

| REPETIR 3 VECES  
| MOVER BARREDORA IZQUIERDA

## TAREA 6

### Ejercicio 1) Mi primer ensalada de frutas

En este ejercicio vamos a intentar entender mejor el tema de los diversos escenarios. El problema es el siguiente, tenemos una única ubicación en el escenario, en la que pueden haber frutas (manzanas y naranjas únicamente, por ahora). Queremos analizar cuáles son los escenarios posibles si se cumplen las siguientes condiciones:

- a) Siempre hay fruta, pero no pueden haber ambas..
- b) Puede o no haber fruta, pero sí hay de una, no hay de la otra.
- c) Puede o no haber fruta e incluso pueden haber ambas.

Dibuje todos los escenarios posibles para cada uno de los puntos mencionados.

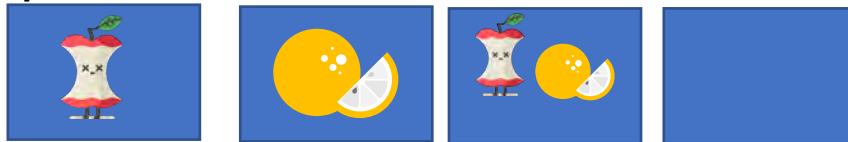
A)



B)



C)



### Ejercicio 2) La ensalada se complica

Sí tenemos 2 frutas posibles, y no tenemos restricciones sobre que puedan o no estar juntas, o que puede estar vacía la ubicación, entonces tendremos 4 estados distintos en el escenario.

Esto responde a lo siguiente:

cantidad de estados = estados posibles de una única fruta 'cantidad de frutas posibles en la ubicación'

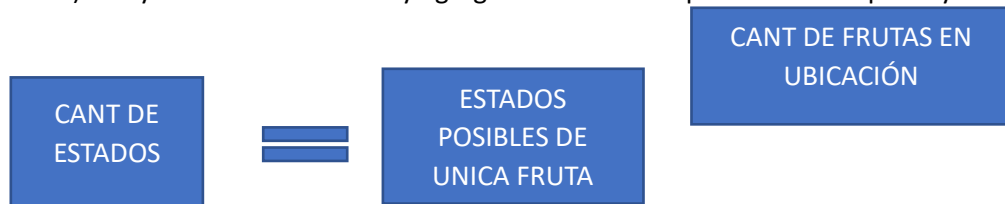
Como vimos hay dos estados para una única fruta (puede o no estar) y teníamos dos posibles frutas (manzana y naranja). Entonces:

cantidad de estados = 2 A LA 2

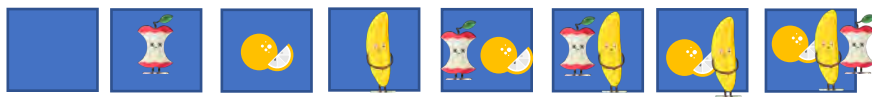
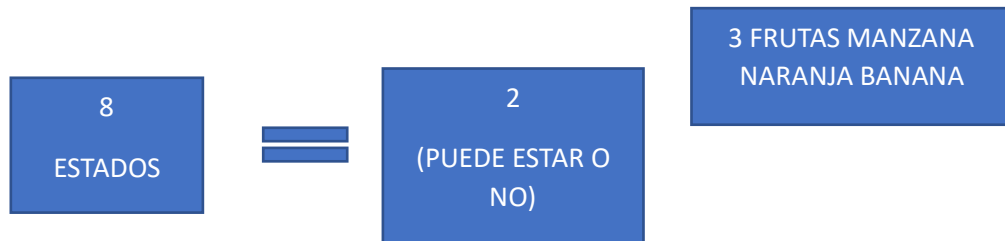
Lo cual como dijimos, da 4.

Se pide entonces analice cuantos estados iniciales posibles se tendría sí:

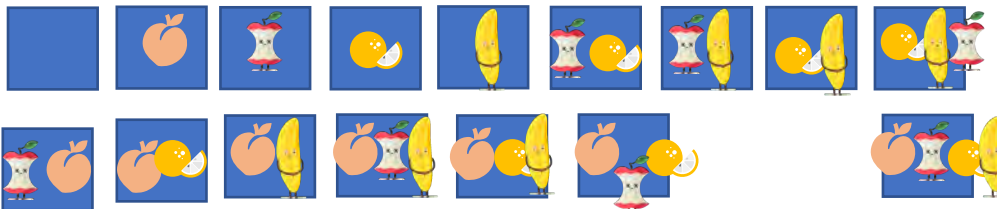
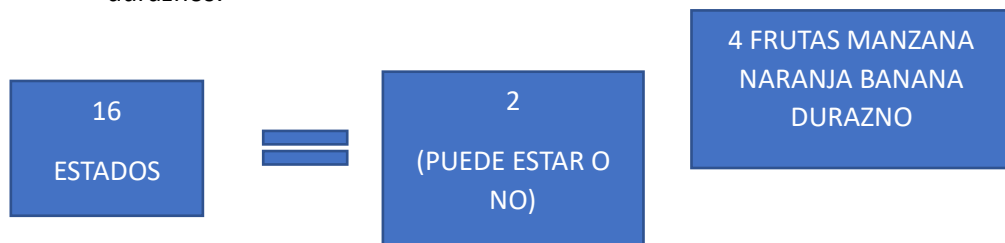
- a) Hay una única ubicación y agregamos además la posibilidad de que haya bananas.



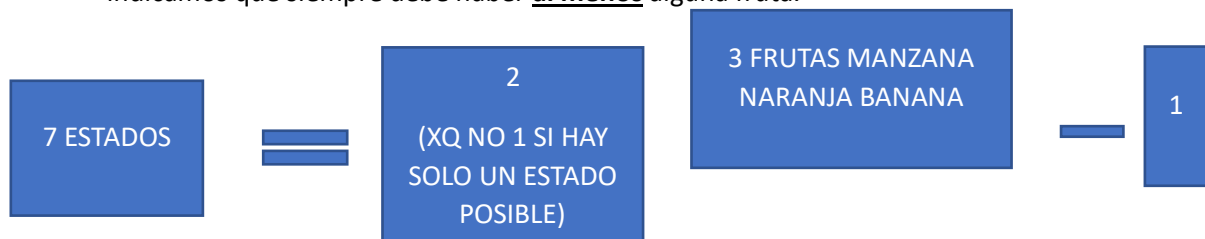
$\wedge=Y \rightarrow \text{NO } V=0$



- b) Hay una única ubicación y agregamos además la posibilidad de que haya bananas y duraznos.



- c) Hay una única ubicación y agregamos además la posibilidad de que haya bananas, pero indicamos que siempre debe haber al menos alguna fruta.



$\wedge=Y \rightarrow \text{NO } V=O$



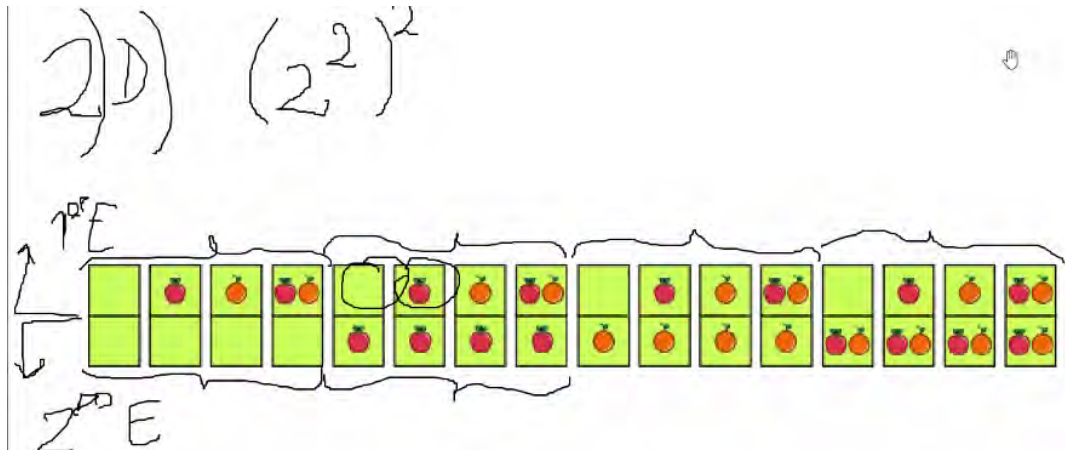
- d) Tenemos solo manzanas y naranjas, pero ahora hay dos ubicaciones.  
 $(2^2)^2=16$



UBICACIÓN 1



UBICACIÓN 2



También podemos usar con una tabla de verdad

- e) Tenemos solo manzanas y naranjas, pero ahora hay tres ubicaciones.  
 $(2^2)^3=64$

- f) Tenemos 100 ubicaciones y 5 frutas distintas.  
 $(2^5)^{100}=3.273390607E150$

Ahora conteste lo siguiente:

- g) ¿Cómo influye la cantidad de ubicaciones en la ecuación? es exponencial son mas ubicaciones mas posibilidades  
 h) ¿Le parece razonable analizar todos los estados iniciales posibles de un ejercicio donde hay muchas ubicaciones y muchas frutas? no, porque son muchos casos  
 i) ¿Qué herramienta vimos que permite simplificar nuestra forma de pensar el problema para no necesitar entender la totalidad de estados posibles? los procedimientos (supuestamente según la profe. cómo es



$$\wedge=Y \rightarrow \text{NO } V=O$$

j) ¿De qué manera podríamos usar esa herramienta para que nuestro cerebro deba pensar solo en 2 casos posibles? sub tareas

**PARCIAL (DIVIDO EN 2 PARTES... TEORICO (4preguntas) Y PRACTICO 1 problema con el autómatas!!!!!!!!!!!!!!!!!!!!!!**

**EJEMPLOS DE EJERCICIOS:**

- 1) Explique la siguiente frase.... "la repetición simplifica "... ejemplifique con un código
- 2) beneficios de usar procedimientos y ejemplifique
- 3) Fragmento de código. Es adecuado o no? Contempla todos los escenarios?

### Ejercicio 3) No todo da lo mismo en el huerto

Analice los siguientes programas y determine si todos son equivalentes o no. En caso de que no

**A y B son equivalentes (a la más adecuada?)**

**~~C y d son equivalentes~~ (c la más adecuada?) NO SON EQUIVALENTES.... AL PPIO LA DUDE...**

**Escenario1: no haya frutas**

**Escenario2: solo berenjena**

**Escenario 3: solo tomate**

**Escenario 4: berenjena y tomate**

Si **no** hay berenjena este falla!!

Sólo resuelven los escenarios 2 y 3!

El **A** es + **optimo** por estar el movimient o afuera de la condición

POSIBLES PROCEDIMIENTOS:  
COSECHAR TOMATE SI HAY  
COSECHAR BERENJENA SI HAY

Cambia el escenario

EL C

Contempla solamente:

ESCENARIOS:

TODOS

Y el es el + **optimo**

a)  
Al empezar a ejecutar  
| Si ¿hay tomate? Entonces  
| | Cosechar tomate  
| Sino  
| | Cosechar berenjena  
| Mover a parcela a la derecha

b)  
Al empezar a ejecutar  
| Si ¿hay tomate? Entonces  
| | Cosechar tomate  
| | Mover a parcela a la derecha  
| Sino  
| | Cosechar berenjena  
| | Mover a parcela a la derecha

c)  
Al empezar a ejecutar  
| Si ¿hay tomate? Entonces  
| | Cosechar tomate  
| Sí ¿Hay berenjena? Entonces  
| | Cosechar berenjena  
| Mover a parcela a la derecha

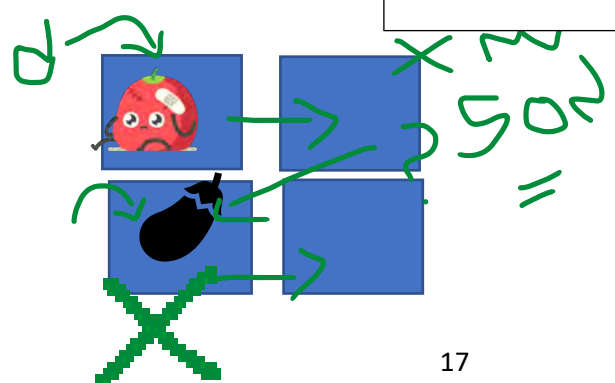
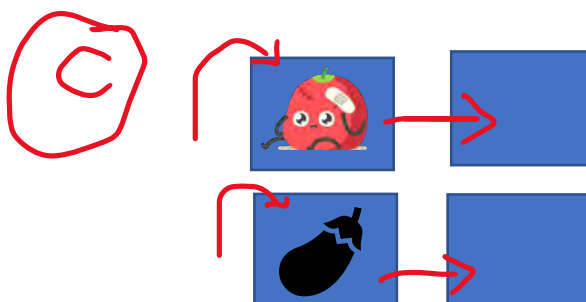
d)  
Al empezar a ejecutar  
| Si ¿hay tomate? Entonces  
| | Cosechar tomate  
| | Mover a parcela a la derecha  
| Sí ¿hay berenjena? Entonces  
| | Cosechar berenjena  
| | Mover a parcela a la derecha

EL D

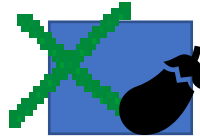
Contempla

ESCENARIOS:

1 2 3 (NO EL 4)

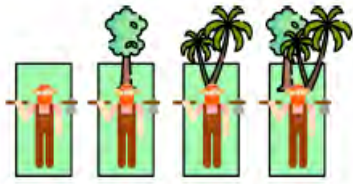


$\wedge=Y \neg=NO \vee=O$



## Ejercicio 4) El leñador y la palmera.

Cuando los leñadores de Ubajay trabajan en zonas protegidas, como La Aurora del Palmar, pueden talar los eucaliptos, pero deben tener mucho cuidado de **no talar las palmeras de yatay**, que son patrimonio y especie protegida. Nuestro problema incluirá **una única ubicación**, en la que se encuentra el leñador, y donde **puede o no haber un árbol**, el cual **puede ser, o bien eucalipto, o bien palmera**. También puede darse el caso en el que haya **tanto eucalipto y palmeras a la misma vez**. A la derecha se muestran los posibles estados iniciales:



Lo que vamos a hacer es solucionar problemas diferentes, contando en cada uno con distintas primitivas.

Para cada solución debe indicar si la misma efectivamente funciona en todos los escenarios, y justificar en caso de que no lo haga, por qué.

- a) En primer lugar, queremos centrarnos pura y exclusivamente en talar los eucaliptos. Para ello contamos con el comando primitivo **"Talar eucalipto"** y el sensor **"¿hay eucalipto?"**.

### AL EMPEZAR A EJECUTAR

| TALAR TODOS LOS EUCALIPTOS DEL ESCENARIO

### DEFINIR TALAR TODOS LOS EUCALIPTOS DEL ESCENARIO

| SI ¿HAY EUCALIPTO? ENTONCES

| TALAR EUCALIPTO

### ESTADOS INICIALES:

A) NO FALLA

B) NO FALLA

C) NO FALLA

D) SI FALLA Fallaría si hay eucalipto y palmeras??

O no falla porque analiza si hay eucalipto y lo tala y listo con el otro no hace nada??

- b) Nuevamente queremos centrarnos en talar los eucaliptos. Para ello contamos con el comando primitivo **"Talar eucalipto"** y el sensor **"¿hay palmera yatay?"**.

### AL EMPEZAR A EJECUTAR

| TALAR TODOS LOS EUCALIPTOS DEL ESCENARIO

### DEFINIR TALAR TODOS LOS EUCALIPTOS DEL ESCENARIO

| SI ¿HAY PALMERA DE YATAY? ENTONCES

|

| SINO

ESTO ESTA BIEN?  
PUEDE QUEDAR UN SI  
VACIO??

$\wedge=Y \neg=NO \vee=O$

### |TALAR EUCALIPTO

#### ESTADOS INICIALES:

A)

c) Esta vez vamos a querer juntar el fruto de la palmera yatay, pues sirve para hacer jaleas y licores. Para ello contamos con el comando primitivo **"Juntar fruto de yatay"** y el sensor **"¿hay eucalipto?"**.

#### AL EMPEZAR A EJECUTAR

|JUNTAR FRUTO DE LA PALMERA YATAY

#### DEFINIR JUNTAR FRUTO DE LA PALMERA YATAY

|SI ¿HAY EUCALIPTO? ENTONCES

|

|SI NO

|JUNTAR FRUTA DE YATAY

d) Nuevamente vamos a querer juntar el fruto de la palmera yatay. Pero esta vez contamos con los comandos **"Juntar fruto de yatay"** y el sensor **"¿hay palmera yatay?"**.

#### AL EMPEZAR A EJECUTAR

|JUNTAR FRUTO DE YATAY

#### DEFINIR JUNTAR FRUTO DE YATAY

|SI ¿HAY PALMERA YATAY?

|JUNTAR FRUTA DE YATAY

e) Por último queremos talar los eucaliptos y juntar los frutos, todo al mismo tiempo. Esta vez contamos con los comandos **"Talar eucalipto"**, **"Juntar fruto de yatay"** y los sensores **"¿hay eucalipto?"** y **"¿hay palmera yatay?"**.

#### AL EMPEZAR A EJECUTAR

|TALAR LOS EUCALIPTOS Y JUNTAR LOS FRUTOS

#### DEFINIR TALAR LOS EUCALIPTOS Y JUNTAR LOS FRUTOS

|TALAR LOS EUCALIPTOS SI LOS HAY

|JUNTAR LOS FRUTOS SI LOS HAY

#### DEFINIR TALAR LOS EUCALIPTOS SI LOS HAY

|SI ¿HAY EUCALIPTO? ENTONCES

|TALAR EUCALIPTO

#### DEFINIR JUNTAR LOS FRUTOS SI LOS HAY

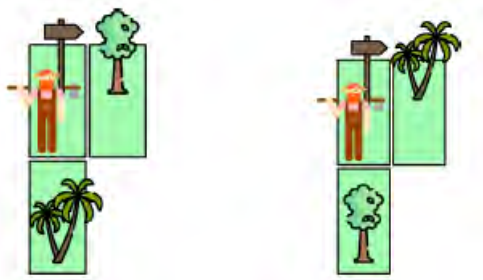
|SI ¿HAY PALMERA YATAY?

|JUNTAR LOS FRUTOS

#### Ejercicio 5) El leñador recargado

En la estación forestal hay indicaciones que ayudan a los leñadores a orientarse sobre dónde debe ir a talar. Una vez que llega a ese lugar debe talar o juntar los frutos de yatay que allí hubiera. Algunos escenarios de ejemplo se muestran a continuación:

$\wedge=Y \rightarrow \text{NO } V=0$



Observar que en estos ejemplos, justo la flecha apunta a la derecha, pero también podría apuntar abajo, pero siempre hay una flecha. Por otro lado, en la ubicación en donde apunta la flecha, puede haber eucaliptus, palmeras, o ambos. Las primitivas disponibles son: “**Mover al leñador a la derecha**”, “**Mover al leñador abajo**”, “**Talar eucalipto**”, “**Juntar fruto de yatay**” y los sensores “**¿hay eucalipto?**”, “**¿hay palmera yatay?**” y “**¿la flecha apunta abajo?**”.

## AL EMPEZAR A EJECUTAR

| TALAR O JUNTAR LOS FRUTOS QUE HUBIERA

## DEFINIR TALAR O JUNTAR LOS FRUTOS QUE HUBIERA

| MOVER LEÑADOR DE ACUERDO CON LA DIRECCION DE LA FLECHA

| JUNTAR FRUTO SI HAY PALMERA

| TALAR SI HAY EUCALIPTO

## DEFINIR MOVER LEÑADOR DE ACUERDO CON LA DIRECCIÓN DE LA FLECHA

| SI ¿LA FLECHA APUNTA ABAJO?

| MOVER AL LEÑADOR ABAJO

| SI NO

| MOVER AL LEÑADOR A LA DERECHA

## DEFINIR JUNTAR FRUTO SI HAY PALMERA

| SI ¿HAY PALMERA YATAY? ENTONCES

| JUNTAR FRUTO DE YATAY

## DEFINIR TALAR SI HAY EUCALIPTO

| SI ¿HAY EUCALIPTO? ENTONCES

| TALAR EUCALIPTO

## TAREA 7

### EJERCICIO 1. PILAS. LABERINTO CORTO

#### OPCIÓN 1

##### AL EMPEZAR A EJECUTAR

| GUIAR AL RATON A LLEGAR A LA META

##### DEFINIR GUIAR AL RATON A LLEGAR A LA META

| MOVER ABAJO SI SE PUEDE

| MOVER A LA DERECHA SI SE PUEDE

##### DEFINIR MOVER ABAJO SI SE PUEDE

| SI ¿PUEDO MOVER ABAJO? ENTONCES

PERO  
FALLA SI  
PUEDE  
MOVER A  
LA  
DERECHA Y  
YA  
TERMINO  
EL  
RECORRID  
O NO?  
NO..  
FALLARIA



$\wedge = Y \quad \neg = \text{NO} \quad \vee = \text{O}$

| MOVER ABAJO

**DEFINIR MOVER A LA DERECHA SI SE PUEDE**

| SI ¿PUEDO MOVER A LA DERECHA? ENTONCES

| MOVER A LA DERECHA

**OPCIÓN 2**

**AL EMPEZAR A EJECUTAR**

| GUIAR AL RATON A LA META

**DEFINIR GUIAR AL RATON A LA META<sup>1</sup>**

| SI ¿PUEDO MOVER A LA DERECHA? ENTONCES

MOVER A LA DERECHA

SI NO

| MOVER ABAJO

**ME FALTAN 3 DE PILAS BLOQUES**

## **TAREA 8**

### **Ejercicio 1) Babe, el chanchito comelón**

Babe, el cerdito comelón, vive en una granja en donde sus dueños disponen para él de manzanas y naranjas, todo un festín para un cerdito. Babe come todo lo encuentra a su paso, no deja absolutamente nada. Pero claro, sus dueños dejan la comida en cualquier lugar del corral, por lo que Babe debe aprender que no siempre las manzanas y las naranjas están en las mismas ubicaciones. En una ubicación puede haber solo una manzana, en otras solo una naranja, en otras pueden haber ambas frutas, e incluso en otros lugares no haber nada. El corral, desde ya, tiene siempre 6 lugares, y Babe siempre espera su almuerzo en la ubicación más a la izquierda, donde con seguridad no hay comida. Esto se muestra en el siguiente escenario:

Ayuda a Babe a comer todo lo que encuentre, escribiendo un programa que lo manipule y utilizando las primitivas y sensores a continuación:

#### **Mover a la derecha**

Mueve a Babe un lugar a la derecha. Debe haber lugar a la derecha.

#### **Comer manzana**

Hace que Babe coma la manzana en la ubicación donde se encuentra. Falla sí no hay una manzana en la ubicación.

#### **Comer naranja**

Hace que Babe coma la naranja en la ubicación donde se encuentra. Falla sí no hay una naranja en la ubicación.

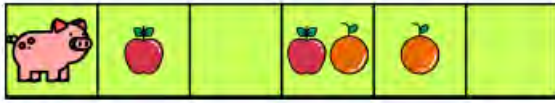
#### **¿hay una manzana?**

Indica sí hay una manzana en la ubicación donde se encuentra Babe.

#### **¿hay una naranja?**

Indica sí hay una naranja en la ubicación donde se encuentra Babe.

$\wedge=Y \rightarrow NO \vee=O$



**AL EMPEZAR A EJECUTAR**

|COMER TODA FRUTA DEL CORRAL

**DEFINIR COMER TODA FRUTA DEL CORRAL**

|MOVER A LA DERECHA Y COMER SI HAY

**DEFINIR COMER SI HAY**

|SI ¿HAY UNA MANZANA? ENTONCES

|COMER MANZANA

|SI ¿HAY NARANJA? ENTONCES

|COMER NARANJA

CREAR PROCEDIMIENTOS  
COMER MANZANA SI HAY  
|SI ¿HAY UNA MANZANA? ENTONCES  
|COMER MANZANA  
COMER NARANJA SI HAY  
|SI ¿HAY NARANJA? ENTONCES  
|COMER NARANJA

**DEFINIR MOVER A LA DERECHA Y COMER SI HAY**

|REPETIR 5 VECES

|MOVER A LA DERECHA

|COMER SI HAY

EL "SI NO" ES EXCLUYENTE!!  
POR ESO NO ES VÁLIDO

OPCION 2: SI NO...

DEFINIR COMER SI HAY

|SI ¿HAY MANZANA? ENTONCES

|COMER MANZANA

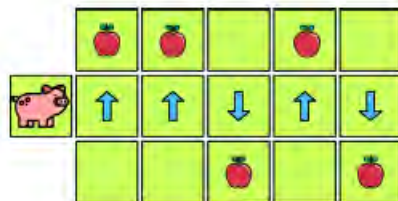
|SI NO

|COMER NARANJA

CREAR PROCEDIMIENTOS CUANDO HAYA CONDICIONALES  
SECUENCIALES

## Ejercicio 2) Babe, el chanchito comelón 2

Babe enfermó un poco y los granjeros lo han movido a otro corral, más ancho, para que tenga mayor movilidad, y han terminado su dieta de naranjas, para terminar con su acidez. Los granjeros ahora alimentan a Babe solo con manzanas que colocan en los bordes de arriba y abajo del corral. Nuevamente disponen manzanas a veces en un lado y a veces en otro (y se garantiza que siempre hay una manzana, o bien a uno o bien a otro lado). Para ayudar a babe, hay flechas en el centro del corral que indican hacia qué lado se encuentran las manzanas. Todo puede apreciarse en el escenario siguiente:



Nuevamente el objetivo es hacer un programa que ayude a Babe a comer todo lo que hay en el corral, pero esta vez tenemos diferentes primitivas:

### Mover abajo

Mueve a Babe un lugar abajo. Debe haber lugar abajo.

### Mover arriba

Mueve a Babe un lugar arriba. Debe haber lugar arriba.

### Mover a la derecha

Mueve a Babe un lugar a la derecha. Debe haber lugar a la derecha.

### Comer manzana

$\wedge=Y \neg= \text{NO } V=O$

Hace que Babe coma la manzana en la ubicación donde se encuentra. Falla sí no hay una manzana en la ubicación.

**¿la flecha apunta arriba?**

Indica sí hay una flecha en la ubicación donde se encuentra Babe que apunta hacia arriba.

**AL EMPEZAR A EJECUTAR**

|COMER TODAS LAS MANZANAS

**DEFINIR COMER TODAS LAS MANZANAS**

|REPETIR 5 VECES

|MOVER A LA DERECHA

|COMER MANZANA SEGÚN LA INDICACION DE FLECHA

**DEFINIR COMER MANZANA SEGÚN LA INDICACIÓN DE FLECHA**

|SI ¿la flecha apunta arriba? ENTONCES

|COMER MANZANA ARRIBA Y VOLVER AL CENTRO

|SINO

|COMER MANZANA ABAJO Y VOLVER AL CENTRO

**DEFINIR COMER MANZANA ARRIBA Y VOLVER AL CENTRO**

|MOVER ARRIBA

|COMER MANZANA

|MOVER ABAJO

**DEFINIR COMER MANZANA ABAJO Y VOLVER AL CENTRO**

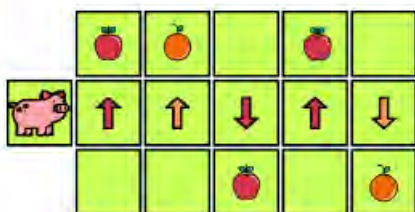
|MOVER ABAJO

|COMER MANZANA

|MOVER ARRIBA

### Ejercicio 3) Babe, el chanchito comelón 3

Ahora que Babe está curado, los granjeros decidieron devolverle su dieta normal, a base tanto de naranjas como de manzanas, pero manteniendo a Babe en el nuevo corral. Así, ahora Babe tendrá no solo flechas que indican hacia qué lado está la fruta a comer, sino que además, la flecha será de distinto color según la fruta que se debe comer. Si la flecha es naranja, se espera una naranja, mientras que si es roja, se espera una manzana.



Nuevamente el objetivo es hacer un programa que ayude a Babe a comer todo lo que hay en el corral, y justo a las primitivas del ejercicio anterior tenemos la primitiva **¿la flecha es roja?** que indica si la flecha en donde está Babe es de color rojo, además de poder **Comer naranja** como en el primer ejercicio.

#### AL EMPEZAR A EJECUTAR

|COMER TODAS LAS FRUTAS QUE HAY EN EL CORRAL

#### DEFINIR COMER TODAS LAS FRUTAS QUE HAY EN EL CORRAL

|REPETIR 5 VECES

|AVANZAR Y COMER LA FRUTA QUE HAYA

#### DEFINIR AVANZAR Y COMER LA FRUTA QUE HAYA

|MOVER A LA DERECHA

|COMER MANZANA O NARANJA DE ACUERDO CON EL COLOR DE LA FLECHA

|VOLVER AL CENTRO

#### DEFINIR COMER MANZANA O NARANJA DE ACUERDO CON EL COLOR DE LA FLECHA

|SI ¿la flecha es roja? ENTONCES

|COMER MANZANA

|SI NO

|COMER NARANJA

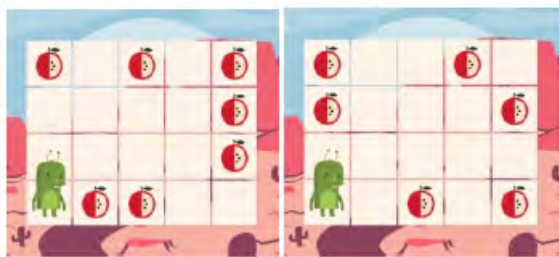
#### Ejercicio 4) El marciano vuelve a tener hambre

Nuevamente el marciano está en el desierto, y vuelve a tener hambre. Deberá comer todas las manzanas que haya en el desierto, las cuales ahora pueden estar en cualquier lugar del escenario. Esta vez el escenario es variable, y el lugar en donde se encuentran las manzanas no es fijo, sino que una ubicación puede o no haber una manzana, y no se sabe a priori. Incluso pueden haber manzanas en el lugar donde arranca el marciano, y también debe comerse dicha manzana. Se muestran a continuación dos posibles escenarios iniciales, pero el escenario inicial podría ser cualquier otro que cumpla la descripción arriba mencionada. Las primitivas son las ya utilizadas en el ejercicio “El marciano en el desierto” de PilasBloques (**Mover a la derecha, Mover a la izquierda, Mover arriba, Mover abajo y Comer manzana**). A la cual agregamos un sensor (**¿hay manzana?**) y la posibilidad de usar alternativa condicional en cualquiera de sus formas. Se pide entonces que escriba un programa que resuelva el problema del marciano.

**Pista:** Como el camino no es lineal, se vuelve difícil moverse. Se sugiere piense dos subtareas principales para solucionar el problema completo. Por un lado, piense como comer todas las manzanas de una única fila. Por otro lado, piense cómo ir desde el borde derecho de una fila al borde izquierdo de la fila siguiente (o sea, ir al comienzo de la próxima fila). Combine dichas subtareas para solucionar el problema principal.



$\wedge=Y \neg=NO \vee=O$



## AL EMPEZAR A EJECUTAR

| COMER TODAS LAS MANZANAS DEL DESIERTO

## DEFINIR COMER TODAS LAS MANZANAS DEL DESIERTO

| REPETIR 4 VECES

| | COMER MANZANAS DE LA COLUMNA ACTUAL

| | IR AL INICIO DE LA COLUMNA SIGUIENTE

| | COMER MANZANAS DE LA COLUMNA ACTUAL

ESTE REPETIR  
DEBERIA IR EN  
OTRO  
PROCEDIMIENTO?

## DEFINIR COMER MANZANAS DE LA COLUMNA ACTUAL

| REPETIR 4 VECES

| | COMER MANZANA SI HAY

| | MOVER ARRIBA

## DEFINIR COMER MANZANA SI HAY

| SI ¿HAY MANZANA? ENTONCES

| | COMER MANZANA

## DEFINIR IR AL INICIO DE LA COLUMNA SIGUIENTE

| MOVER A LA DERECHA

| BAJAR 3 ESPACIOS

## DEFINIR BAJAR 3 ESPACIOS

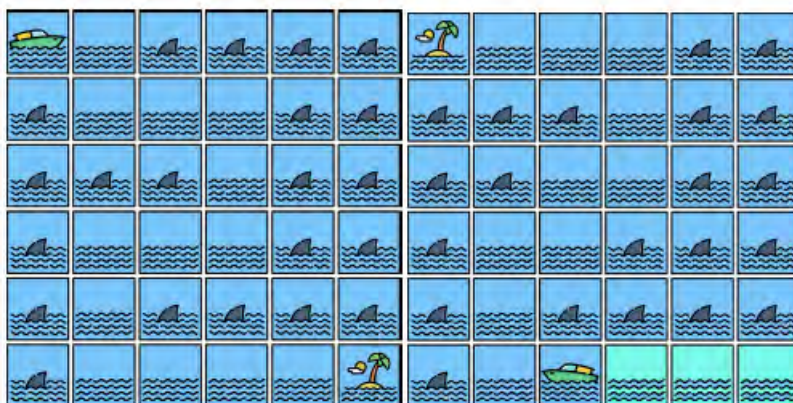
| REPETIR 3 VECES

| | MOVER ABAJO

## Ejercicio 5) La isla del sol

Todo está listo para llegar a la paradisíaca Isla del Sol, el bote tiene combustible, la tierra está a la vista, el mar está calmo y el día está soleado. Pero no será tan fácil, pues hay que encontrar el camino que no tiene tiburones para poder llegar de forma segura al destino. En esta actividad el escenario puede variar, tanto en el lugar de donde arranca el bote, como el lugar donde se encuentra la isla, como el camino a seguir para llegar. Lo bueno es que sabemos que la isla está a exactamente 14 ubicaciones de distancia, sin importar cual sea el camino, además de que siempre hay un camino. Pueden verse dos ejemplos de escenario a continuación, en el primero, el bote aún no ha arrancado el recorrido, mientras que en el segundo ya ha dado 3 pasos por el camino sin tiburones.

$\wedge=Y \rightarrow NO V=O$



El bote es nuestro autómeta, y entiende las siguientes primitivas:

### Mover arriba

Hace que el bote se mueva una ubicación hacia arriba. Falla sí no hay más lugar arriba o sí hay tiburones en dicha dirección.

### Mover abajo

Hace que el bote se mueva una ubicación hacia abajo. Falla sí no hay más lugar abajo o sí hay tiburones en dicha dirección.

### Mover a la izquierda

Hace que el bote se mueva una ubicación hacia la izquierda. Falla sí no hay más lugar a la izquierda o sí hay tiburones en dicha dirección.

### Mover a la derecha

Hace que el bote se mueva una ubicación hacia la derecha. Falla sí no hay más lugar a la derecha o sí hay tiburones en dicha dirección.

### ¿sigue el camino arriba?

Indica sí el camino del bote sigue hacia arriba (es decir, no hay tiburones o se acaba el camino hacia arriba, y el bote no viene de allí)

### ¿sigue el camino abajo?

Indica sí el camino del bote sigue hacia abajo (es decir, no hay tiburones o se acaba el camino hacia abajo, y el bote no viene de allí)

### ¿sigue el camino a la izquierda?

Indica sí el camino del bote sigue hacia la izquierda (es decir, no hay tiburones o se acaba el camino hacia la izquierda, y el bote no viene de allí)

### ¿sigue el camino a la derecha?

Indica sí el camino del bote sigue hacia la derecha (es decir, no hay tiburones o se acaba el camino hacia la derecha, y el bote no viene de allí)

### Desembarcar

Desembarca en la isla. El bote debe estar en la misma ubicación que la isla. Se busca un procedimiento "Ir a la isla del sol" que lleve el bote a la isla del sol y desembarque allí.

**Pista:** Acá la parte difícil del recorrido es la de moverse. ¿Qué sucede si pongo dos alternativas consecutivas? ¿Se mueve siempre de la forma esperada? ¿Cuántas veces avanza? ¿Y si pongo una alternativa dentro de otra (separando en subtareas adecuadamente, claro)?

$\wedge=Y \neg=NO V=O$

```

Al empezar a ejecutar
  Ir a la isla del sol

Definir Ir a la isla del sol
  Llevar el bote a la isla
  Desembarcar

Definir Llevar el bote a la isla
  Repetir 14 veces
    Mover a la posición posible sin tiburón

Definir Mover a la posición posible sin tiburón
  Si ¿sigue el camino a la derecha? Entonces
    Mover a la derecha
  Sino
    Mover a la posición posible sin derecha sin tiburón

Definir Mover a la posición posible sin derecha sin tiburón
  Si ¿sigue el camino a la abajo? Entonces
    Mover a la abajo
  Sino
    Mover a la posición posible sin derecha ni abajo sin tiburón

Definir Mover a la posición posible sin derecha ni abajo sin tiburón
  Si ¿sigue el camino a arriba? Entonces
    Mover a la arriba
  Sino
    Mover a la izquierda
  
```

## Ejercicio 6) María la fanática de las sandías

María quiere volver a comer todas las sandías del escenario, que ahora tiene una nueva complejidad. Las sandías pueden estar en cualquier lugar del escenario (incluso donde está María al arranque), o bien pueden haber lugares donde no haya sandías. Las primitivas a utilizar son las mismas del ejercicio “María y las sandías” de PilasBloques (**Mover arriba**, **Mover abajo**, **Mover a la izquierda**, **Mover a la derecha** y **Comer sandía**) a la cual se agrega el sensor **¿hay sandía acá?** que indica si hay una sandía en la ubicación donde está María. El escenario a continuación muestra un posible escenario inicial: Cree un programa con las primitivas mencionadas que ayude a María a comer todas las sandías del escenario.



**AL EMPEZAR A EJECUTAR**

| COMER TODAS LAS SANDIAS DEL ESCENARIO

**DEFINIR COMER TODAS LAS SANDIAS DEL ESCENARIO**

| REPETIR 5 VECES

ESTE REPETIR  
DEBERIA IR EN UN  
PROCEDIMIENTO  
NO?

$\wedge = Y$   $\neg = \text{NO}$   $V = O$

```
| COMER TODAS LAS SANDIAS DE LA COLUMNA ACTUAL
| IR AL PUNTO DE INICIO DE LA COLUMNA SIGUIENTE
| COMER TODAS LAS SANDIAS DE LA COLUMNA ACTUAL
```

## DEFINIR COMER TODAS LAS SANDIAS DE LA COLUMNA ACTUAL

```
| REPETIR 4 VECES
| COMER SI HAY SANDIA
| MOVER ARRIBA
|
```

## DEFINIR COMER SI HAY SANDIA

```
| SI ¿HAY SANDIA ACA? ENTONCES
| COMER SANDIA
```


## DEFINIR IR AL PUNTO DE INICIO DE LA COLUMNA SIGUIENTE

```
| MOVER A LA DERECHA
| REPETIR 5 VECES
| MOVER ARRIBA
```

## Ejercicio 7) María come otras frutas

María descubrió que además de las sandías existen otras frutas deliciosas. Estuvo probando las ciruelas y las bananas, y ahora quiere volver al ataque, comiendo todas las frutas que encuentre en el escenario. En este caso el escenario tiene 10 columnas por 6 filas. María comienza en la esquina inferior izquierda del mismo (donde también podría haber una fruta). En cada ubicación del escenario puede haber una o más frutas, o no haber nada. Sí hay fruta en una ubicación, solo hay una de ellas (es decir, una sandía, una banana o una ciruela). En este ejercicio se suman a las primitivas que ya teníamos los comandos primitivos **Comer banana** y **Comer ciruela**, que hacen lo que su nombre sugiere, así como los sensores **¿hay banana acá?** Y **¿hay ciruela acá?** Se pide escriba un programa que haga que María coma toda la fruta del escenario. Luego responda:

- ¿Su código funciona en el caso de que el escenario contenga ubicaciones con más de una fruta? SI
- ¿Y en el caso de que no haya nada? SI
- ¿Cambiaría el código si como condición inicial sabemos que nunca hay más de una fruta? MMM NO SE

## AL EMPEZAR A EJECUTAR

```
|COMER TODAS LAS FRUTAS DEL ESCENARIO
```

$\wedge = Y$   $\neg = \text{NO}$   $V = O$

## DEFINIR COMER TODAS LAS FRUTAS DEL ESCENARIO

| REPETIR 9 VECES

| | COMER FRUTAS DE LA COLUMNA ACTUAL SI HAY

| | IR AL INICIO DE LA COLUMNA SIGUIENTE

| | COMER FRUTAS DE LA COLUMNA ACTUAL

Este repetir debería ir en un

Procedimiento?

## DEFINIR COMER FRUTAS DE LA COLUMNA ACTUAL SI HAY

| REPETIR 6 VECES

| | REVISAR Y COMER LAS FRUTAS QUE HAYA

| | MOVER ARRIBA

## DEFINIR REVISAR Y COMER LAS FRUTAS QUE HAYA

| | COMER SANDIA SI HAY

| | COMER BANANA SI HAY

| | COMER CIRUELA SI HAY

ACA USE SÓLO CONDICIONAL  
SIMPLE...PORQUE SI HAY 2 FRUTAS ..  
LAS PUEDE COMER

## DEFINIR COMER CIRUELA SI HAY

| | SI ¿HAY CIRUELA? ENTONCES

| | | | COMER CIRUELA

## DEFINIR COMER SANDIA SI HAY

| | SI ¿HAY SANDIA? ENTONCES

| | | | COMER SANDIA

## DEFINIR COMER BANANA SI HAY

| | SI ¿HAY BANANA? ENTONCES

| | | | COMER BANANA

## DEFINIR IR AL INICIO DE LA SIGUIENTE COLUMNA

| | MOVER A LA DERECHA

| | BAJAR 5 ESPACIOS

## DEFINIR BAJAR 5 ESPACIOS

| | REPETIR 5 VECES

| | | | MOVER ABAJO

## Ejercicio 8) Beelly la abeja

Beelly la abeja, nuestro autómatas para esta actividad, **quiere llegar a su panal, recogiendo todo el polen de las flores que encuentre en el camino.** El escenario es sencillo, hay diversas ubicaciones, sectorizadas mediante una grilla cuadriculada. Hay un **camino que va desde donde arranca Beelly hasta el panal, el cual es único (no hay bifurcaciones) y tiene exactamente 21 ubicaciones (desde la ubicación donde está Beelly, incluyendo dicha ubicación, hasta la ubicación donde está el panal, inclusive).** La forma del camino es aleatoria, y no sabemos qué giros deberá realizar Beelly para llegar al panal. En las ubicaciones que no forman parte del camino, hay pasto, el cual Beelly no puede sobrevolar. Por otro lado, en las ubicaciones del camino, y de forma

$\wedge=Y \neg=NO \vee=O$

aleatoria, pueden haber flores (es decir, hay ubicaciones con flores y ubicaciones sin flores, sin estar claro en donde, ya que pueden variar). Incluso pueden haber flores en donde comienza Beelly, aunque seguro no hay flores sobre el panal. A continuación se muestra un posible escenario inicial, aunque otros escenarios iniciales son posibles, siempre que se ajusten a la descripción arriba mencionada. Lo que se pide es que realice un programa que lleve a Beelly de su ubicación actual, al panal, entrando al mismo, y recogiendo el polen de cada flor por la que pasa. Para ello se cuenta con las siguientes primitivas y sensores:

## **Volar arriba**

Hace que Beelly se mueva una ubicación para arriba. Falla sí hay pasto hacia arriba, o no hay más ubicaciones hacia allí.

## **Volar abajo**

Hace que Beelly se mueva una ubicación para abajo. Falla sí hay pasto hacia abajo, o no hay más ubicaciones hacia allí.

## **Volar a la derecha**

Hace que Beelly se mueva una ubicación para la derecha. Falla sí hay pasto hacia la derecha, o no hay más ubicaciones hacia allí.

## **Volar a la izquierda**

Hace que Beelly se mueva una ubicación para la izquierda. Falla sí hay pasto hacia la izquierda, o no hay más ubicaciones hacia allí.

## **Recoger polen**

Hace que Beelly recoja polen de la flor. Falla sí Beelly no está sobre una flor.

## **Entrar al panal**

Hace que Beelly entre al panal. Falla sí Beelly no está sobre el panal..

## **¿continúa el camino sin pasto arriba?**

Indica sí el camino, desde la ubicación actual de Beelly, continúa hacia arriba.

## **¿continúa el camino sin pasto abajo?**

Indica sí el camino, desde la ubicación actual de Beelly, continúa hacia abajo.

## **¿continúa el camino sin pasto a la izquierda?**

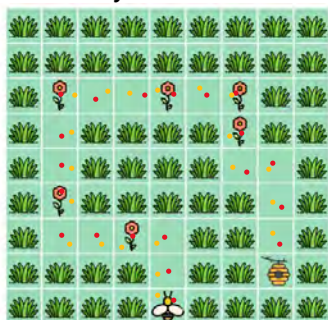
Indica sí el camino, desde la ubicación actual de Beelly, continúa hacia la izquierda.

## **¿continúa el camino sin pasto a la derecha?**

Indica sí el camino, desde la ubicación actual De Beelly, continúa hacia la derecha.

## **¿hay una flor acá?**

Indica sí hay una flor en la ubicación en donde se encuentra Beelly.



## **AL EMPEZAR A EJECUTAR**

| IR AL PANAL RECOGIENDO POLEN DE LAS FLORES QUE HAYA

**DEFINIR IR AL PANAL RECOGIENDO POLEN DE LAS FLORES QUE HAYA**

$\wedge = Y$   $\neg = \text{NO}$   $\vee = \text{O}$

| RECOGER TODO EL POLEN QUE HAYA  
| ENTRAR AL PANAL

## DEFINIR RECOGER TODO EL POLEN QUE HAYA

| REPETIR 20 VECES  
|       | RECOGER POLEN SI HAY FLOR  
|       | VOLAR A LA POSICION POSIBLE SIN PASTO

## DEFINIR RECOGER POLEN SI HAY FLOR

| SI ¿hay una flor acá? ENTONCES  
|       | RECOGER POLEN

## DEFINIR VOLAR A LA POSICIÓN POSIBLE SIN PASTO

| SI ¿continúa el camino sin pasto arriba? ENTONCES  
|       | VOLAR ARRIBA  
| SINO  
|       | VOLAR A LA UBICACIÓN POSIBLE SIN ARRIBA SIN PASTO

## DEFINIR VOLAR A LA UBICACIÓN POSIBLE SIN ARRIBA SIN PASTO

| SI ¿continúa el camino sin pasto abajo? ENTONCES  
|       | VOLAR ABAJO  
| SINO  
|       | VOLAR A LA UBICACIÓN POSIBLE SIN ABAJO SIN PASTO

## DEFINIR VOLAR A LA UBICACIÓN POSIBLE SIN ABAJO NI ARRIBA SIN PASTO

| SI ¿continúa el camino sin pasto a la izquierda? ENTONCES  
|       | VOLAR A LA IZQUIERDA  
| SINO  
|       | VOLAR A LA DERECHA



## PARCELES NUEVAS PARCELA I CHANCHO

**Ejercicio 1)** Resuelva el siguiente ejercicio: El cerdito Babe recargado Babe, el cerdito comelón, vive en una granja en donde sus dueños disponen para él de **manzanas, naranjas, bananas y paltas**, todo un festín para un cerdito. Babe come todo lo encuentra a su paso, no deja absolutamente nada. Pero claro, sus dueños dejan la comida en cualquier lugar del corral, por lo que Babe debe aprender que no siempre las frutas están en las mismas ubicaciones. **En una ubicación puede haber sólo un tipo de fruta o nada.** El corral, desde ya, tiene siempre **9 lugares de ancho y 9 lugares de alto**, Babe siempre espera su almuerzo en la ubicación más a la izquierda y arriba del escenario, **donde con seguridad no hay comida.** Esto se muestra en el siguiente ejemplo. Cualquier otro escenario es viable, lo que **nunca cambia es el tamaño del corral, la posición inicial del cerdito y la ubicación donde se encuentran las flechas.**

Babe está entrenado y sabe que **si hay una flecha roja, hay una manzana en la ubicación vecina a la derecha, si el color es naranja, hay una naranja a la izquierda, si es verde, una palta abajo y si es amarilla, una banana arriba.** Se pide que ayude a Babe a consumir las frutas del corral, para ello se cuenta con las siguientes primitivas.

### Mover arriba

Hace que Babe se mueva una ubicación para arriba. Falla si no hay más ubicaciones hacía allí.

### Mover a la derecha

Hace que Babe se mueva una ubicación para la derecha. Falla si no hay más ubicaciones hacía allí.

### Mover abajo

Hace que Babe se mueva una ubicación para abajo. Falla si no hay más ubicaciones hacía allí.

### Mover a la izquierda

Hace que Babe se mueva una ubicación para la izquierda. Falla si no hay más ubicaciones hacía allí.

### ¿la flecha es roja?

indica si la flecha en donde está Babe es de color rojo.

### ¿la flecha es naranja?

indica si la flecha en donde está Babe es de color naranja.

### ¿la flecha es naranja?

indica si la flecha en donde está Babe es de color naranja.

### ¿la flecha es verde?

indica si la flecha en donde está Babe es de color verde.

### Comer banana

Hace que Babe coma una banana donde está ubicado. Falla si no hay banana.

### Comer naranja

Hace que Babe coma una naranja donde está ubicado. Falla si no hay manzana.

### Comer manzana

Hace que Babe coma una manzana donde está ubicado. Falla si no hay manzana.

### Comer palta

Hace que Babe coma una palta donde está ubicado. Falla si no hay palta.



$\wedge = Y$   $\neg = \text{NO}$   $\vee = \text{O}$

## AL EMPEZAR A EJECUTAR

|COMER TODAS LAS FRUTAS DEL CORRAL

## DEFINIR COMER TODAS LAS FRUTAS DEL CORRAL

|REPETIR 2 VECES

| |POSICIONAR Y COMER LAS FRUTAS DE LAS FLECHAS DE LA COLUMNA ACTUAL

| |IR AL PUNTO INICIAL DE LA SIGUIENTE COLUMNA

|COMER TODAS LAS FRUTAS DE LAS FLECHAS DE LA COLUMNA ACTUAL

## DEFINIR POSICIONAR Y COMER LAS FRUTAS DE LAS FLECHAS DE LA COLUMNA ACTUAL

|POSICIONARSE EN 1ER FLECHA DE LA COLUMNA

|COMER LA FRUTA DE ACUERDO CON EL COLOR DE LA FECHA

|REPETIR 2 VECES

| |BAJAR 3 CASILLEROS

| |COMER LA FRUTA DE ACUERDO CON EL COLOR DE LA FECHA

## DEFINIR POSICIONARSE EN 1ER FLECHA DE LA COLUMNA

|MOVER ABAJO

|MOVER A LA DERECHA

## DEFINIR BAJAR 3 CASILLEROS

|REPETIR 3 VECES

| |MOVER ABAJO

## DEFINIR COMER LA FRUTA DE ACUERDO CON EL COLOR DE LA FLECHA

|SI ¿LA FLECHA ES ROJA? ENTONCES

| |COMER MANZANA Y VOLVER

|SI NO

| |COMER LA FRUTA QUE NO ES MANZANA

## DEFINIR COMER MANZANA Y VOLVER

| |MOVER A LA DERECHA

| |COMER MANZANA

| |MOVER LA IZQUIERDA

## DEFINIR COMER LA FRUTA QUE NO ES MANZANA

|SI ¿LA FLECHA ES NARANJA? ENTONCES

| |COMER NARANJA Y VOLVER

|SI NO

| |COMER LA FRUTA QUE NO ES MANZANA NI NARANJA

## DEFINIR COMER NARANJA Y VOLVER

| |MOVER A LA IZQUIERDA

| |COMER NARANJA

| |MOVER A LA DERECHA

## DEFINIR COMER LA FRUTA QUE NO ES MANZANA NI NARANJA

|SI ¿LA FLECHA ES VERDE? ENTONCES

| |COMER PALTA Y VOLVER

|SINO COMER LA FRUTA QUE NO ES MANZANA NI NARANJA NI PALTA

| |COMER BANANA Y VOLVER

## DEFINIR COMER PALTA Y VOLVER

|MOVER ABAJO

| |COMER PALTA

| |MOVER ARRIBA

## DEFINIR COMER BANANA Y VOLVER

| |MOVER ARRIBA

| |COMER BANANA

| |MOVER ABAJO



$\wedge = Y$   $\neg = \text{NO}$   $V = O$

## DEFINIR IR AL PUNTO INICIAL DE LA SIGUIENTE COLUMNA

| MOVER 2 CASILLEROS A LA DERECHA  
| SUBIR 7 CASILLEROS

## DEFINIR SUBIR 7 CASILLEROS

| REPETIR 7 VECES  
| MOVER ARRIBA

## DEFINIR MOVER 2 CASILLEROS A LA DERECHA

| REPETIR 2 VECES  
| MOVER A LA DERECHA

# MODELO DE PROCIN 2 OVNI

Ejercicio 1) Conteste las siguientes preguntas:

- a) ¿con que se describen las acciones en un lenguaje de programacion? ¿y los datos?

Las acciones se describen con comandos. Y los datos con expresiones.

EXPRESIONES NO?

- b) ¿Qué beneficios se perderían si no hubiera procedimientos? Mencione 4 cosas:

Si no hubiera procedimientos se perdería:

- 1) Claridad
- 2) Legibilidad
- 3) Modificabilidad
- 4) Reutilización/o capacidad de separar los problemas en partes más pequeñas

- c) Indique si el siguiente código es adecuado o no y justifique

AL COMENZAR A EJECUTAR

Sí ¿está pintado de azul?

Repetir 3 veces

Pasar rodillo rojo

El código no es adecuado.

- 1) Falla la sintaxis de la escritura en papel

- a. Le faltan las líneas en cada renglón
- b. A la alternativa condicional le falta la palabra ENTONCES
- c. No debería estar anidado un repetir dentro de una alternativa condicional se debe crear un procedimiento

## Ejercicio 2) RESUELVA

OVNI es la sigla de Objeto Volador No identificado, y es también el objeto que será nuestro autómatas en este ejercicio. Y es que el OVNI es una nave extraterrestre que quiere volver a Marte, el planeta rojo, pero ha olvidado a varios de sus tripulantes en el camino, y deberá recogerlos antes de regresar.

El escenario es sencillo, hay diversas ubicaciones, sectorizadas mediante una grilla cuadrículada. Hay un camino que va **desde donde arranca el OVNI hasta Marte, el cual es único (no hay bifurcaciones) y tiene exactamente 29 ubicaciones** (desde la ubicación donde está el ovni, incluyendo esta, hasta la ubicación donde está el planeta rojo). La forma del camino es aleatoria, y no sabemos qué vueltas deberá realizar el ovni para llegar a Marte. En las ubicaciones que no forman parte del camino hay estrellas, las cuales **el ovni no puede atravesar**. Por otro lado, en las ubicaciones del camino, y de forma aleatoria, pueden haber extraterrestres esperando ser recogidos (es decir **hay ubicaciones con extraterrestres y ubicaciones sin extraterrestres**, sin estar claro en dónde, ya que pueden variar).

A la derecha se muestra un posible escenario inicial aunque otros escenarios iniciales son posibles siempre que se ajusten a la descripción arriba mencionada.

Lo que se pide es que realice un programa que lleve al ovni de su ubicación actual a Marte, recoja a los extraterrestres que encuentre en el camino. Para ello se cuentan con las siguientes primitivas y sensores:

**volar arriba**

hace que el ovni se mueva una ubicación para arriba. Falla y hay una estrella hacia arriba, o no hay más ubicaciones hacia allí.

**Volar a la derecha**

hace que el ovni se mueva una ubicación para la derecha. Facha si hay una estrella hacia la derecha o no hay más ubicaciones hacia allí.

**Volar hacia abajo**

hace que el ovni se mueva hacia abajo. Falla si hay una estrella hacia abajo o no hay más ubicaciones

**volar a la izquierda**

hace que el ovni se mueva un casillero a la izquierda. Falla si hay una estrella hacia la izquierda o no hay más ubicaciones hacia allí.

**Recoger extraterrestres**

hace que el ovni recoja un extraterrestre de donde se encuentra. Fallas y el ovni no está sobre un extraterrestre

**Aterrizar en Marte**

hace que el ovni aterrizas en Marte

**¿puede volar hacia arriba?**

Indica si el camino, desde la ubicación actual del ovni continúa hacia arriba

**¿puede volar hacia abajo?**

Indica si el camino desde la ubicación actual del ovni continua hacia abajo

**¿Puede volar a la izquierda?**

Indica si el camino desde la ubicación actual del ovni continua hacia la izquierda


**¿Puede volar a la derecha?**

Indica si el camino desde la ubicación actual del ovni continua hacia la derecha

**¿hay un extraterrestre acá?**

Indica si hay un extraterrestre en la ubicación donde este el OVNI

$\wedge = Y$   $\neg = \text{NO}$   $\vee = \text{O}$

ovni								
								

## AL EMPEZAR A EJECUTAR

| LLEVAR OVNI A MARTE RECOGIENDO LOS EXTRATERRESTRES QUE ENCUENTRE

## DEFINIR LLEVAR OVNI A MARTE RECOGIENDO LOS EXTRATERRESTRES QUE ENCUENTRE

| IR A MARTE RECOGIENDO LOS EXTRATERRESTRES QUE ENCUENTRE

| ATERRIZAR EN MARTE

## DEFINIR IR A MARTE RECOGIENDO LOS EXTRATERRESTRES QUE ENCUENTRE

| REPETIR 28 VECES

| RECOGER LOS EXTRATERRESTRES QUE HAYA

| VOLAR A LA POSICION SIN ESTRELLAS

## DEFINIR RECOGER LOS EXTRATERRESTRES QUE HAYA

| SI ¿HAY UN EXTRATERRESTRE ACA? ENTONCES

| RECOGER EXTRATERRESTRE

## DEFINIR VOLAR A LA UBICACIÓN SIN ESTRELLAS

| SI ¿puede volar a la derecha? ENTONCES

| VOLAR A LA DERECHA

| SINO

| VOLAR A LA UBICACIÓN POSIBLE SIN DERECHA

## DEFINIR VOLAR A LA UBICACIÓN POSIBLE SIN DERECHA

| SI ¿PUEDE VOLAR A LA IZQUIERA? ENTONCES

VOLAR A LA IZQUIERA

| SINO

| VOLAR A LA UBICACIÓN POSIBLE SIN DERECHA SIN IZQUIERDA

## DEFINIR VOLAR A LA UBICACIÓN POSIBLE SIN DERECHA SIN IZQUIERDA

SI ¿PUEDE VOLAR ABAJO? ENTONCES

| VOLAR ABAJO

SINO

| VOLAR ARRIBA

27 28 O 29

MARTE ESTA EN EL  
CASILLERO 29?

ESE INCLUSIVE ME  
HACE RUIDO

EN EL CASILLERO  
29

## MODELO DE PROCIN = PLOMERO

### EJERCICIO 1)

- A) ¿para qué sirven los procedimientos? Los procedimientos dan:
- Claridad
  - Legibilidad
  - Modificabilidad
  - Reutilización/o capacidad de separar los problemas en partes más pequeñas
- B) ¿qué diferencia hay entre comando y expresión? Un comando es una descripción de una acción y una expresión es una descripción de información
- C) indique si el siguiente código es adecuado o no y justifique

#### Al comenzar a ejecutar

```
| repetir 5 veces
    | si hay fuego acá?
        | apagar fuego
    | dar un paso a la derecha
| si hay fuego acá
    | apagar fuego
```

El código no es adecuado.

- 2) Falla la sintaxis de la escritura en papel
- A la alternativa condicional le falta la palabra ENTONCES
  - El ultimo condicional simple debería sumarse al bloque de repetir
  - De debería separar movimiento de procedimiento

### EJERCICIO 2

Mario es un plomero esquizofrénico, que sueña que una raza de tortugas quiere destruir el mundo y que debe pisarles la cabeza para destruirlas a todas. Así Mario es buscado por distintas sociedades protectoras de los animales por sus numerosos crímenes contra las tortugas. En este ejercicio, Mario será nuestro autómatas y debemos ayudarlo a llegar a la tubería antes de que lo atrapen los ambientalistas.

El escenario es sencillo, hay diversas ubicaciones sectorizadas mediante una grilla cuadrículada. Hay un camino que va desde donde arranca Mario hasta la tubería, el cual es único (no hay bifurcaciones) y **tiene exactamente 26 ubicaciones desde la ubicación donde está Mario inclusive hasta la ubicación donde está la tubería inclusive**). La forma del camino es aleatoria y no sabemos qué giros deberá realizar Mario para llegar a la tubería.

En las ubicaciones que no forman parte del camino hay ladrillos con los cuales Mario no quiere estrellarse. Por otro lado en las ubicaciones del camino y de forma aleatoria pueden haber tortugas (es decir hay ubicaciones con tortugas y ubicaciones sin tortugas sin estar Claro en dónde ya que pueden variar).

A la derecha se muestra un posible escenario inicial, aunque otros escenarios iniciales son posibles siempre que se ajusten a la descripción arriba mencionada.

$\wedge=Y \neg=NO \vee=O$

Lo que se pide es que realice un programa que lleve a Mario de su ubicación actual a la tubería, para escapar por la misma, aplastando toda tortuga que se cruce por su camino. Por ello se cuenta con las siguientes primitivas y sensores

**mover hacia arriba**

hace que Mario se mueva una ubicación para arriba falla hacia ahí ladrillos hacia arriba o no hay más ubicaciones hacia allí

**mover a la derecha**

**mover hacia abajo**

**mover a la izquierda**

**aplastar tortuga**

**continúa el camino arriba?**

Indica si el camino desde la ubicación actual de Mario continúa hacia arriba

**continúa el camino a la izquierda?**

Indica si el camino desde la ubicación actual de Mario continúa hacia la izquierda

**continúa el camino abajo?**

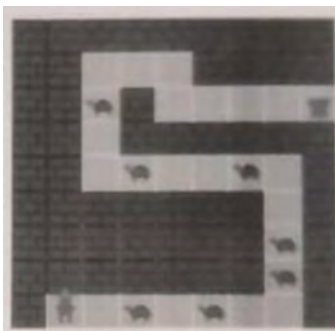
Indica si el camino desde la ubicación actual de Mario continúa hacia abajo

**continúa el camino a la derecha?**

indica si el camino desde la ubicación actual de Mario continuó hacia la derecha

**hay tortuga?**

Indica si hay una tortuga en la ubicación en donde se encuentra Mario  
escapar por tubería



$\wedge=Y$   $\neg=$  NO  $\vee=O$

**AL EMPEZAR A EJECUTAR**

| ESCAPAR APLASTANDO LAS TORTUGAS QUE HAYA

**DEFINIR ESCAPAR APLASTANDO LAS TORTUGAS QUE HAYA**

| APLASTAR TODA TORTUGA QUE HAYA

| ESCAPAR POR TUBERIA

**DEFINIR APLASTAR TODA TORTUGA QUE HAYA**

| REPETIR 25 VECES

| APLASTAR TORTUGA SI HAY

| MOVERSE A LA UBICACIÓN DONDE NO HAYA LADRILLOS

**DEFINIR APLASTAR TORTUGA SI HAY**

| SI ¿HAY TORTUGA? ENTONCES

APLASTAR TORTUGA

**DEFINIR MOVERSE A LA UBICACIÓN DONDE NO HAYA LADRILLOS**

| SI ¿continúa el camino a la derecha? ENTONCES

| MOVER A LA DERECHA

| SINO

| MOVER A LA UBICACIÓN DONDE NO HAYA LADRILLOS SIN DERECHA

**DEFINIR MOVER A LA UBICACIÓN DONDE NO HAYA LADRILLOS SIN DERECHA**

SI ¿continúa el camino a la izquierda? ENTONCES

MOVER A LA IZQUIERDA

SINO

| MOVER MOVER A LA UBICACIÓN DONDE NO HAYA LADRILLOS SIN DERECHA  
NI IZQUIERDA

**DEFINIR MOVER A LA UBICACIÓN DONDE NO HAYA LADRILLOS SIN DERECHA NI IZQUIERDA**

SI ¿continúa el camino abajo? ENTONCES

| MOVER ABAJO

SINO

MOVER ARRIBA

## MODELO DE PROBLEMA 4 FROGGY LA RANA

### Ejercicio 1

- a)
- b)
- c)

definir hacer algo

- | si -¿hay queso acá?
  - | dar un paso en el laboratorio
- | sino
  - | comer el queso
  - | dar un paso en el laboratorio

### Ejercicio 2)

Froggy La rana vuelve a la acción. Esta vez el estanque está lleno de menu Juárez, y freddy no ve la hora de comer todos los insectos del lugar.

El escenario es sencillo, froggy puede moverse por cualquier lugar, y en algunas ubicaciones específicas hay insectos y en otras no. El escenario es variable, y no sabemos a priori ni su tamaño ni tampoco en qué lugares están los insectos. Más aún, en los lugares donde hay insectos, tampoco sabemos cuántos, ya que pueden haber uno o más. Freddy siempre comienza en la esquina superior izquierda del escenario, donde también puede o no haber insectos.

Pero no todo es tan fácil para froggy cómo ir saltando de un nenúfar a otro. Junto con los insectos puede haber depredadores, como los pelícanos o los erizos. Freddy debe salir disparado de esos lugares, sin poder comer los insectos que allí se encuentran.

A la derecha se muestra una posible escenario inicial aunque otros escenarios iniciales son posibles siempre que se ajusten a la descripción arriba mencionada. En este ejemplo, el escenario tiene 9 columnas por 9 filas.

Lo que se pide es que realice un programa que haga que froggy coma, todos los insectos del escenario que sean posible.

sean posible. Para ello puede disponer de las siguientes primitivas y sensores.

<b>Saltar abajo</b> Hace que Froggy se mueva una ubicación para arriba. Falla si no hay más lugar hacia arriba.	<b>Saltar a la derecha</b> Hace que Froggy se mueva una ubicación para la derecha. Falla si no hay más lugar a la derecha o si Froggy no está en la fila más arriba del escenario.
<b>Ir a la fila superior</b> Hace que Froggy se mueva hasta la fila más arriba del escenario.	<b>Comer insecto</b> Hace que Froggy coma un único insecto de la ubicación que encuentra. Falla si Froggy no está sobre un insecto o si todos los insectos de la ubicación ya fueron comidos.
<b>cantidad de columnas del escenario</b> Sensor numérico que describe la cantidad de columnas totales del escenario.	<b>¿se puede saltar hacia abajo?</b> Indica si Froggy puede moverse un lugar hacia abajo.
<b>¿hay insecto acá?</b> Indica si hay insectos (uno o más) en la ubicación donde está Froggy.	<b>¿hay pelicano acá?</b> Indica si hay un pelicano en la ubicación donde está Froggy.
<b>¿hay erizo acá?</b> Indica si hay un erizo en la ubicación donde está Froggy.	<b>cantidad de insectos</b> Sensor numérico que describe la cantidad de insectos en la ubicación donde está Froggy. Falla si no hay insectos en la ubicación.



## **PARCIAL 5 MARIO EL CARTERO**

Mario el cartero nuestro autómatas para esta actividad quiere llegar al correo recogiendo todas las cartas y paquetes que pueda en el camino.

El escenario es sencillo hay diversas ubicaciones sectorizadas mediante una grilla cuadrículada. Hay un camino que va desde donde arranca Mario hasta el correo, el cual es único no hay bifurcaciones y **tiene exactamente 23 ubicaciones incluye la ubicación donde están Mario hasta la ubicación donde está el correo inclusive**. La forma del camino es aleatoria y no sabemos que giros deberá realizar Mario para llegar al correo.

En las ubicaciones que no forman parte del camino hay casas las cuales Mario no puede atravesar. Por otro lado, en las ubicaciones del camino y de forma aleatoria puede haber cartas Y/O paquetes es decir hay ubicaciones con cargas paquetes o ambas y hay ubicaciones sin nada sin estar claro dónde ya que pueden variar)

a la derecha se muestra un posible escenario inicial, aunque otros escenarios iniciales son posibles siempre que sea ajusten a la descripción arriba mencionada.

Lo que se pide que realice es un programa que lleve a Mario de su ubicación actual, al correo recogiendo las cartas y los paquetes del camino. Para ello se cuenta con las siguientes primitivas y sensores

### **Mover arriba**

hace que Mario se mueva una ubicación para arriba falla si hay casa hacia arriba o no hay más ubicaciones allí.

**Mover a la derecha** hace que Mario en ubicación para la derecha falla si hay una casa hacia la derecha o no hay más ubicaciones hacia allí

recoger carta hace que Mario recoge una carta fallas y Mario no está sobre una carta

**mover abajo** hace que Mario se mueva una ubicación para abajo falla si hay casa hacia abajo o no hay más ubicaciones hacia allí

**mover a la izquierda** hace que Mario se mueva una ubicación a la izquierda falla si hay casas izquierdas o no hay más ubicaciones hacia allí recoger paquete hace que Mario recoja un paquete fallas y Mario no está sobre un paquete

sensores

**¿continúa el camino arriba?**

**¿Continúa el camino a la izquierda?**

**¿continúa el camino abajo?**

**¿continúa el camino a la derecha?**

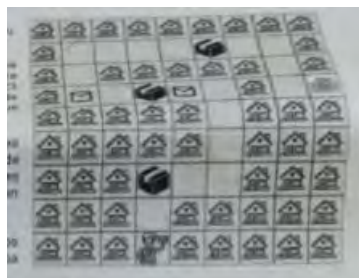
**¿hay una carta acá?**

**¿hay un paquete acá?**

**entrar al correo y dejar lo recolectado**

$\wedge = Y$   $\neg = \text{NO}$   $V = O$

<b>Mover arriba</b> Hace que Maui se mueva una ubicación para arriba. Falla si hay casa hacia arriba, o no hay más ubicaciones hacia allí.	<b>Mover abajo</b> Hace que Maui se mueva una ubicación para abajo. Falla si hay casa hacia abajo, o no hay más ubicaciones hacia allí.
<b>Mover a la derecha</b> Hace que Maui se mueva una ubicación para la derecha. Falla si hay casa hacia la derecha, o no hay más ubicaciones hacia allí.	<b>Mover a la izquierda</b> Hace que Maui se mueva una ubicación para la izquierda. Falla si hay casa hacia la izquierda, o no hay más ubicaciones hacia allí.
<b>Recoger Carta</b> Hace que Maui recoja una carta. Falla si Maui no está sobre una carta.	<b>Recoger Paquete</b> Hace que Maui recoja un paquete. Falla si Maui no está sobre un paquete.
<b>¿continúa el camino arriba?</b> Indica si el camino, desde la ubicación actual de Maui, continúa hacia arriba.	<b>¿continúa el camino abajo?</b> Indica si el camino, desde la ubicación actual de Maui, continúa hacia abajo.
<b>¿continúa el camino a la izquierda?</b> Indica si el camino, desde la ubicación actual de Maui, continúa hacia la izquierda.	<b>¿continúa el camino a la derecha?</b> Indica si el camino, desde la ubicación actual de Maui, continúa hacia la derecha.
<b>¿hay una carta acá?</b> Indica si hay una carta en la ubicación en donde se encuentra Maui.	<b>¿hay un paquete acá?</b> Indica si hay un paquete en la ubicación en donde se encuentra Maui.
<b>Entrar al correo y dejar lo recolectado</b> Hace que Maui entre al correo. Falla si Maui no está sobre el correo.	



## MODELO DE PARCIAL 6 BOMBERO

### EJERCICIO 1)

- qué diferencia hay entre un comando y una expresión  
 un comando es una descripción de acción y una expresión es una descripción de información
- mentené cuatro más motivos por los que es conveniente contar con un lenguaje que tenga procedimientos  
 modificabilidad  
 claridad  
 legibilidad  
 reutilización de Código
- Indique si es adecuado y justifique
- Al empezar a ejecutar

Repetir 3 veces

Repetir 3 veces

Recolectar flor

Mover a derecha

Definir recolectar flores acá

Repetir 3 veces

Recolectar flor

### Ejercicio 2

Resuelve el siguiente ejercicio



El camión de bomberos o autobomba es uno de los más útiles vehículos en las grandes urbes. El fuego es algo que puede ocurrir en cualquier momento y sólo una autobomba puede apagarlo.

El escenario es sencillo, hay diversas ubicaciones sectorizadas de forma triangular. Hay 2 tipos de ubicaciones las que son triángulos hacia arriba y las que son triángulos hacia abajo. Cada triángulo hacia arriba tiene un vecino a la izquierda uno a la derecha y uno hacia abajo salvo que esté en algún borde los que están hacia abajo tienen a la izquierda y derecha y un vecino hacia arriba.

Hay un camino que va desde donde arranca el autobomba hasta el cuartel de bomberos el cual es único no hay bifurcaciones y tiene exactamente 26 ubicaciones desde la ubicación donde está el camión de bomberos hasta la ubicación donde está el cuartel. La forma del camino es aleatorio y no sabemos qué giros deberá realizar el camión para llegar al cuartel

en las ubicaciones que no forman parte del camino hay casas por las cual es el camión no puede cruzar. Por otro lado, en las ubicaciones del camino y de forma aleatoria puede haber fuego es decir hay ubicaciones con fuego y ubicaciones sin fuego sin estar Claro en dónde ya que pueden variar.

A la derecha se muestra un posible escenario inicial aunque otros escenarios iniciales son posibles siempre que se ajusten a la descripción arriba menciona.

Lo que pide es que realice un programa que lleve la autobomba de su ubicación al cuartel de bomberos donde se recargara agua apagando todo incendio que esté en el camino. Primitivas y sensores

$\wedge = Y$   $\neg = \text{NO}$   $V = O$

<b>Mover autobomba arriba</b> Hace que la autobomba se mueva una ubicación para arriba. Falla si hay una casa hacia arriba, si no hay más ubicaciones hacia allí o si la ubicación es un triángulo hacia arriba.	<b>Mover autobomba abajo</b> Hace que la autobomba se mueva una ubicación para abajo. Falla si hay una casa hacia abajo, si no hay más ubicaciones hacia allí o si la ubicación es un triángulo hacia abajo.
<b>Mover autobomba a la derecha</b> Hace que la autobomba se mueva una ubicación para la derecha. Falla si hay una casa hacia la derecha, o no hay más ubicaciones hacia allí.	<b>Mover autobomba a la izquierda</b> Hace que la autobomba se mueva una ubicación para la izquierda. Falla si hay una casa hacia la izquierda, o no hay más ubicaciones hacia allí.
<b>Apagar incendio</b> Hace que la autobomba apague el fuego. Falla si no hay fuego en la ubicación donde está la autobomba.	<b>Recargar autobomba</b> Hace que la autobomba se recargue. Falla si la autobomba no está en el cuartel de bomberos.
<b>¿sigue calle arriba?</b> Indica si la calle por la que se mueve la autobomba continúa hacia arriba.	<b>¿sigue calle abajo?</b> Indica si la calle por la que se mueve la autobomba continúa hacia abajo.
<b>¿sigue calle a la derecha?</b> Indica si la calle por la que se mueve la autobomba continúa hacia la derecha.	<b>¿sigue calle a la izquierda?</b> Indica si la calle por la que se mueve la autobomba continúa hacia la izquierda.

$\wedge=Y \rightarrow \text{NO } V=0$

## Ejercicio 1) Resuelva el siguiente ejercicio: Ailín la repositora

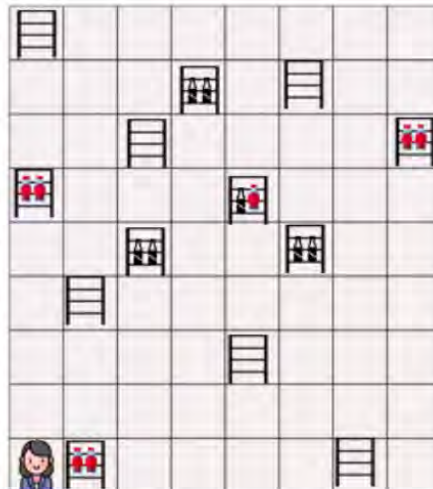
Ailín, la repositora, trabaja en un mercado gigante y se ocupa de que las góndolas tengan gaseosas para los clientes. De ella depende que las personas interesadas puedan comprar gaseosa.

En el local, en cada ubicación puede haber o no una góndola, y cuando hay una, puede que no esté completa. Está completa cuando tiene gaseosas de cola y gaseosas de lima-limón.

Hay ubicaciones que no tienen góndolas, otras con una góndola con sólo gaseosas de cola, otras con una góndola que sólo tiene gaseosas de lima-limón, otras que tienen una góndola completa y otras con la góndola vacía.

El mercado, desde ya, tiene siempre 8 lugares de ancho y 9 lugares de alto, Ailín siempre empieza en la ubicación más a la izquierda y abajo del escenario. Cualquier otro escenario es viable, lo que nunca cambia es el tamaño del mercado y la posición inicial de Ailín.

Se pide generar un programa que haga que Ailín deje el local completo preparado para recibir a los clientes y avise cuando termine. Para lograr el objetivo se cuenta con las siguientes primitivas:



<b>Mover arriba</b> Hace que Ailín se mueva una ubicación para arriba. Falla si no hay más ubicaciones hacia allí.	<b>Mover a la derecha</b> Hace que Ailín se mueva una ubicación para la derecha. Falla si no hay más ubicaciones hacia allí.
<b>Mover abajo</b> Hace que Ailín se mueva una ubicación para abajo. Falla si no hay más ubicaciones hacia allí.	<b>Mover a la izquierda</b> Hace que Ailín se mueva una ubicación para la izquierda. Falla si no hay más ubicaciones hacia allí.
<b>¿hay góndola?</b> Indica si hay una góndola donde está Ailín.	<b>¿faltan gaseosas de cola?</b> Indica si en la góndola donde está Ailín, faltan gaseosas de cola. Falla si no hay góndola.
<b>¿faltan gaseosas de lima-limón?</b> Indica si en la góndola donde está Ailín, faltan las gaseosas de lima-limón. Falla si no hay góndola.	<b>Reponer gaseosa de cola</b> Hace que Ailín reponga las gaseosas de cola. Falla si no faltan las gaseosas de cola.
<b>Reponer gaseosa de lima-limón</b> Hace que Ailín reponga las gaseosas de lima-limón. Falla si no faltan las gaseosas de lima-limón.	<b>Avisar que se repusieron las gaseosas</b> Hace que Ailín avise que está todo listo para recibir a los clientes. Falla si no se repusieron todas las gaseosas.

## LÓGICA Y CONECTIVAS

### Ejercicio 1) A cocinar

Considere las siguientes preguntas como las únicas que pueden ser entendidas por un almacenero a quien debe comprar los ingredientes para varias preparaciones.

- ¿Hay harina?
- ¿Hay manteca?
- ¿Hay aceite?
- ¿Hay agua?
- ¿Hay huevos?
- ¿Hay yerba?
- ¿Hay chocolate?
- ¿Hay azúcar?

Se pide que escriba las siguientes preguntas en términos de las anteriores, utilizando las conectivas vistas. Puede reutilizar la pregunta de un punto anterior o inventar preguntas intermedias para contestar aquellas más complejas si lo considera útil.

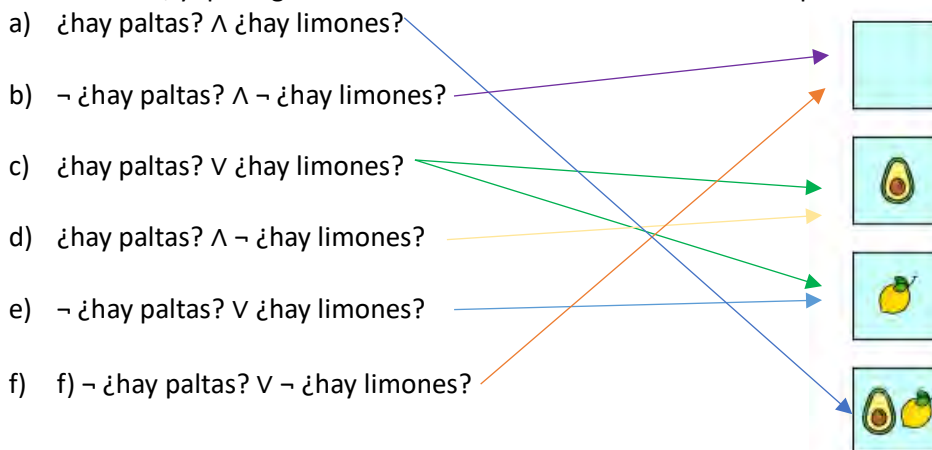
- a. ¿hay para hacer una torta? (Una torta requiere harina, huevos y manteca)  
 $\text{¿Hay harina?} \wedge \text{¿Hay manteca?} \wedge \text{¿Hay huevos?}$
- b. ¿hay para hacer huevos fritos? (Requiere huevos y aceite)  
 $\text{¿Hay aceite?} \wedge \text{¿Hay huevos?}$
- c. ¿hay para hacer huevos duros? (Requiere huevos y agua)  
 $\text{¿Hay agua?} \wedge \text{¿Hay huevos?}$
- d. ¿Puedo almorzar huevos? (Ya sean duros o fritos)  
 $\text{¿Hay huevos?}$
- e. ¿hay para hacer una torta de chocolate? (Idéntico a una torta, más chocolate)  
 $\text{¿Hay harina?} \wedge \text{¿Hay manteca?} \wedge \text{¿Hay huevos?} \wedge \text{¿Hay chocolates?}$
- f. ¿Solo se puede tomar mate amargo? (Cuando se puede tomar mate, es decir, hay agua y yerba, pero **no hay azúcar??**)  
 $\text{¿Hay yerba?} \wedge \text{¿Hay agua?} \wedge \neg \text{¿Hay azucar?}$
- g. ¿No hay nada para el mate? (Cuando se puede tomar mate, pero no hay torta de ningún tipo)  
 $\text{¿Hay yerba?} \wedge \text{¿Hay agua?} \wedge \neg \text{¿Hay harina?} \wedge \neg \text{¿Hay manteca?} \wedge \neg \text{¿Hay huevos?}$

Ayuda: La respuesta debe estar formulada de forma similar a: ¿Hay bananas?  $\wedge$  ¿Hay manzanas?



$$\wedge=Y \quad \neg=NO \quad \vee=O$$

**Ejercicio 2)** Las paltas de Patricia Patricia tiene un hermoso terreno con un árbol de paltas y un limonero. Se pide que una entonces con flechas las expresiones de la izquierda cuando son verdaderas con los escenarios a la derecha. Puede que algunos elementos se relacionen con varios escenarios, y que algunos escenarios se relacionen con varias expresiones.



**Ejercicio 3)** Preparándose para la batalla Considere las siguientes preguntas como las únicas que pueden ser realizadas en principio.

- |                                   |                                  |
|-----------------------------------|----------------------------------|
| • ¿Hay ejército enemigo al Norte? | • ¿Hay ejército aliado al Norte? |
| • ¿Hay ejército enemigo al Este?  | • ¿Hay ejército aliado al Este?  |
| • ¿Hay ejército enemigo al Sur?   | • ¿Hay ejército aliado al Sur?   |
| • ¿Hay ejército enemigo al Oeste? | • ¿Hay ejército aliado al Oeste? |

Se pide que escriba las siguientes preguntas en términos de las anteriores. Puede reutilizar la pregunta de un punto anterior o inventar preguntas intermedias para contestar aquellas más complejas si lo considera útil.

h. ¿Se está amenazado? (Cuando hay ejército enemigo en alguna dirección)  
**¿Hay ejército enemigo al Norte?  $\wedge$  ¿Hay ejército enemigo al Este?  $\wedge$  ¿Hay ejército enemigo al Sur?  $\wedge$  ¿Hay ejército enemigo al Oeste?**

i. ¿Se está libre de peligro? (Cuando no hay ejército enemigos en ninguna dirección)  
 **$\neg$ ¿Hay ejército enemigo al Norte?  $\wedge$   $\neg$ ¿Hay ejército enemigo al Este?  $\wedge$   $\neg$ ¿Hay ejército enemigo al Sur?  $\wedge$   $\neg$ ¿Hay ejército enemigo al Oeste?**

j. ¿Se tiene apoyo? (Cuando hay un ejército aliado en alguna dirección)  
**¿Hay ejército aliado al Norte?  $\vee$  ¿Hay ejército aliado al Este?  $\vee$  ¿Hay ejército aliado al Sur?  $\vee$  ¿Hay ejército aliado al Oeste?**

Esto significa que se usara "o"?

k. ¿Se está hasta las manos? (Cuando no hay apoyo y se está amenazado)  
**¿Hay ejército enemigo al Norte?  $\wedge$  ¿Hay ejército enemigo al Este?  $\wedge$  ¿Hay ejército enemigo al Sur?  $\wedge$  ¿Hay ejército enemigo al Oeste?  $\wedge$   $\neg$ ¿Hay ejército aliado al Norte?  $\wedge$   $\neg$ ¿Hay ejército aliado al Este?  $\wedge$   $\neg$ ¿Hay ejército aliado al Sur?  $\wedge$   $\neg$ ¿Hay ejército aliado al Oeste?**

l. ¿Se puede neutralizar alguna amenaza? (Cuando hay un ejército enemigo en alguna dirección, pero también hay un ejército aliado allí)

$$\wedge = Y \quad \neg = \text{NO} \quad \vee = \text{O}$$

(¿Hay ejército enemigo al Norte?  $\wedge$  ¿Hay ejército aliado al Norte?)  $\vee$  (¿Hay ejército enemigo al este?  $\wedge$  ¿Hay ejército aliado al este?)  $\vee$  (¿Hay ejército enemigo al oeste?  $\wedge$  ¿Hay ejército aliado al oeste?)  $\vee$  (¿Hay ejército enemigo al sur?  $\wedge$  ¿Hay ejército aliado al sur?)

m. ¿Se puede neutralizar todas las amenazas? (Cuando se puede neutralizar en todas las direcciones)

(¿Hay ejército enemigo al Norte?  $\wedge$  ¿Hay ejército aliado al Norte?)

$\wedge$  (¿Hay ejército enemigo al Este?  $\wedge$  ¿Hay ejército aliado al Este?)

$\wedge$  (¿Hay ejército enemigo al Oeste?  $\wedge$  ¿Hay ejército aliado al Oeste?)

$\wedge$  (¿Hay ejército enemigo al Sur?  $\wedge$  ¿Hay ejército aliado al Sur?)

DUDA!  
¿esta  
correcto?

**Ejercicio 4)** Conectivas o alternativas Un nuevo programador que entró a una empresa viene bastante confundido con los conceptos elementales, y no termina de entender las diferencias entre conectivas y alternativas. Queremos ayudarlo y mostrarle que no siempre las alternativas actúan igual que una condición con conectivas. Para esto, se pide que analice las siguientes soluciones de código y busque escenarios en donde quede en claro que no son equivalentes.

<p>a)</p> <pre> Al empezar a ejecutar   Si ¿hay tomate? Entonces     Cosechar tomate   Si ¿Hay berenjena? Entonces     Cosechar berenjena   Mover a parcela a la derecha                     </pre>	<p>c)</p> <pre> Al empezar a ejecutar   Si ¿hay tomate? <math>\vee</math> ¿Hay berenjena? Entonces     Cosechar tomate     Cosechar berenjena   Mover a parcela a la derecha                     </pre>
<p>b)</p> <pre> Al empezar a ejecutar   Si ¿hay tomate? <math>\wedge</math> ¿Hay berenjena? Entonces     Cosechar tomate     Cosechar berenjena   Mover a parcela a la derecha                     </pre>	

No serian equivalentes... tal vez el más parecido es el c pero al tener el mover parcela dentro del condicional si no hay tomate ni berenjena no podrapa moverse a la derecha

**Ejercicio 5)** Píxeles Los televisores y monitores, bajo un microscopio, se componen de millones de ubicaciones en donde, en cada una, hay 3 lamparitas, una roja, una verde y una azul. Según se prendan o se apaguen dichas lamparitas, y con distinta intensidad, se logran todos los colores posibles. Los primeros monitores solo contemplaban un encendido o apagado, y no intensidades diversas. Nos basamos en un monitor antiguo con esa última característica., y lo que buscamos es poder comprender el estado de una ubicación puntual, asumiendo que la expresión dada es verdadera. Así, se pide que una con flechas las expresiones de la izquierda con uno o más estados de la derecha.

No entiendo.. azul prendida o roja o verde o sea cuando hay una sola?

- a) ¿azul prendida?  $\vee$  ¿roja prendida?  $\vee$  ¿verde prendida?
- b) ¿azul prendida?  $\wedge$  ¿roja prendida?  $\wedge$  ¿verde prendida?
- d) ¿azul prendida?  $\vee$  ( $\neg$  ¿roja prendida?  $\wedge$   $\neg$  ¿verde prendida?)
- c)  $\neg$  ¿azul prendida?  $\wedge$   $\neg$  ¿roja prendida?  $\wedge$   $\neg$  ¿verde prendida?
- e) ¿azul prendida?  $\wedge$  ¿roja prendida?  $\wedge$   $\neg$  ¿verde prendida?
- d) (¿azul prendida?  $\vee$   $\neg$  ¿roja prendida?)  $\wedge$   $\neg$  ¿verde prendida?



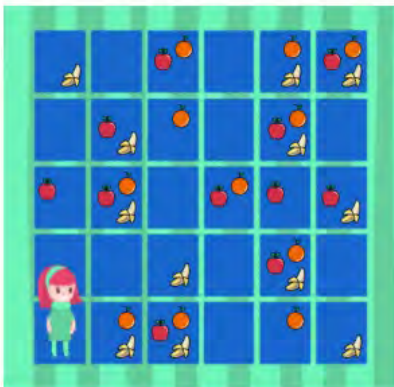
Cuando hay azul o no hay roja—este paréntesis que valor de verdad da?

O no hay



## Ejercicio 6) María y sus ensaladas de frutas

María sigue enganchada con las frutas, no solo con las sandías, sino con todas. Su último descubrimiento es la ensalada de frutas, y le gustó tanto, que ahora no quiere comer otra cosa. Ahora en el escenario hay varias frutas dispersas por el mismo, y puede haber una o varias en una misma ubicación. Para preparar una ensalada de frutas **deben haber en una misma ubicación tanto bananas, como manzanas y naranjas**. Si falta alguna de las frutas en la ubicación, se vuelve imposible preparar la ensalada de frutas. **El escenario inicial tiene 5 filas y 6 columnas, y María comienza en la esquina inferior izquierda.** En cada ubicación puede no haber nada, o haber frutas, combinadas de diversas formas: sólo manzanas, solo naranjas, solo bananas, bananas y naranjas, bananas y manzanas, manzanas y naranjas o bananas, naranjas y manzanas. A la derecha se muestra un ejemplo posible de escenario inicial. Se pide escriba un procedimiento llamado **Comer ensaladas de frutas** que ayude a María a comer todas las ensaladas de frutas que puedan ser preparadas en el escenario, para lo cual se dispone de las siguientes primitivas.



<b>Mover arriba</b> Hace que María se mueva hacia arriba una ubicación. Debe haber una ubicación hacia arriba.	<b>Mover abajo</b> Hace que María se mueva hacia abajo una ubicación. Debe haber una ubicación hacia abajo.
<b>Mover a la izquierda</b> Hace que María se mueva hacia la izquierda una ubicación. Debe haber una ubicación hacia la izquierda.	<b>Mover a la derecha</b> Hace que María se mueva hacia la derecha una ubicación. Debe haber una ubicación hacia la derecha.
<b>¿hay banana?</b> Indica si hay una banana en la ubicación donde está María.	<b>¿hay manzana?</b> Indica si hay una manzana en la ubicación donde está María.
<b>¿hay naranja?</b> Indica si hay una naranja en la ubicación donde está María.	<b>Preparar y comer ensalada de frutas</b> Prepara una ensalada de frutas con los ingredientes de la ubicación donde está María, y se la come. Debe haber una manzana, una naranja y una banana en la ubicación donde está María.

### AL EMPEZAR A EJECUTAR

| COMER TODAS LAS ENSALADAS DE FRUTAS DEL ESCENARIO

### DEFINIR TODAS LAS ENSALADAS DE FRUTAS DEL ESCENARIO

| REPETIR 5 VECES

| COMER ENSALADAS DE FRUTAS POSIBLES DE LA COLUMNA ACTUAL

| IR AL INCIO DE LA COLUMNA SIGUIENTE

| COMER ENSALADAS DE FRUTAS POSIBLES DE LA COLUMNA ACTUAL

### DEFINIR COMER ENSALADAS DE FRUTAS POSIBLES DE LA COLUMNA ACTUAL

| REPETIR 4 VECES

| COMER ENSALADA DE FRUTAS SI HAY TODOS LOS INGREDIENTES EN LA UBICACIÓN

| MOVER ARRIBA

| COMER ENSALADA DE FRUTAS SI HAY TODOS LOS INGREDIENTES EN LA UBICACIÓN

### DEFINIR COMER ENSALADA DE FRUTAS SI HAY TODOS LOS INGREDIENTES EN LA UBICACION

| SI ¿HAY BANANA?  $\wedge$  ¿HAY MANZANAS?  $\wedge$  ¿HAY NARANJA?

| PREPARAR Y COMER ENSALADA DE FRUTA

### DEFINIR IR AL INCIO DE LA COLUMNA SIGUIENTE

| REPETIR 4 VECES

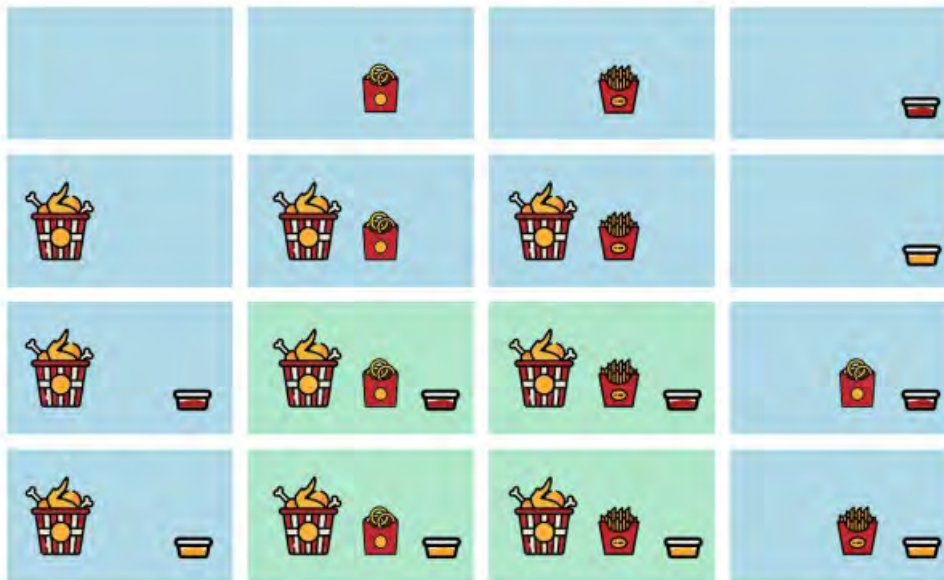
| MOVER ABAJO

| MOVER A LA DERECHA

## DEFINICIÓN DE EXPRESIONES

### Ejercicio 1) Hurlingham's Fried Chicken

El pollo frito, uno de los platos más populares de la gastronomía estadounidense, ha dado ya la vuelta al mundo y es comercializado a lo largo del globo en distintas cadenas y establecimientos. Hurlingham's Fried Chicken es un nuevo restaurante que quiere dar al comensal la mejor experiencia en este tipo de platos, y para que lo disfrutes como se debe, no solo sirve un par de piezas de pollo frito, sino que este debe ser servido en un combo de forma adecuada. Un buen combo de pollo frito debe incluir un acompañamiento, que pueden ser papas fritas o aros de cebolla, y además tener una salsa donde mojar el pollo, la cual suele incluir salsa barbacoa o queso cheddar fundido. Esto, junto con el pollo frito, presenta una experiencia óptima según los dueños del establecimiento. Un robot ha sido puesto sobre el mostrador para que notifique a los comensales que el combo está listo para retirar. Para ello, el robot debe identificar cuando un combo está listo, cuando un combo aún no está terminado y se encuentra en proceso de preparación. El robot mira la mesada, que en este caso es la totalidad de nuestro escenario (o sea, hay una única ubicación en el escenario), y debe notificar "combo listo" o "combo en espera". Los escenarios posibles son cualquiera de los siguientes:



Sí lo analizamos vemos que:

- Puede o no estar el balde de pollo frito.
- Puede haber aros de cebolla o papas fritas, o ninguna de las dos.
- Puede haber salsa de cheddar o barbacoa, o ninguna de las dos.
- No hay nunca aros de cebolla y papas fritas al mismo tiempo.
- No hay nunca salsa barbacoa y salsa de cheddar al mismo tiempo.
- Los escenarios en donde debe indicarse que el combo está listo son aquellos que aparecen en verde, los azules son escenarios en donde falta alguno de los elementos

Vamos a contar con la siguiente lista de sensores primitivos:

**¿está el pollo frito?**

Indica sí está el balde de pollo frito en el escenario.

**¿están las papas fritas?**

$\wedge = Y$   $\neg = \text{NO}$   $V = O$

Indica si están las papas fritas en el escenario.

**¿están los aros de cebolla?**

Indica si están los aros de cebolla en el escenario.

**¿está la salsa de cheddar?**

Indica si está la salsa de cheddar en el escenario.

**¿está la salsa de BBQ?**

Indica si está la salsa de barbacoa en el escenario.

Lo que se pide entonces es que **defina las expresiones:**

a) **¿hay salsa? (que indica si hay salsa, ya sea cheddar o BBQ)**

**DEFINIR ¿HAY SALSA? = ¿HAY CHEDDAR? V ¿HAY BBQ?**

b) **¿hay acompañamiento? (que indica que hay acompañamiento, ya sean papas fritas o aros de cebolla)**

**DEFINIR ¿HAY ACOMPAÑAMIENTO? = ¿HAY PAPAS FRITAS? V ¿HAY AROS DE CEBOLLA?**

c) **¿está listo el combo? (que indica si el combo está listo, tal cual se detalla en el enunciado)**

**DEFINIR ¿ESTA LISTO EL COMBO? = ¿HAY POLLO FRITO?  $\wedge$  ¿HAY ACOMPAÑAMIENTO?  $\wedge$  ¿HAY SALSA?**

Ahora, utilizando los siguientes comandos primitivos, realice un programa que solucione el problema de avisar que el combo está listo si este efectivamente lo está o dejar claro que está en preparación en caso contrario.

**Avisar que “el combo está listo”**

Hace que el robot avise que el combo está listo.

**Avisar que “el combo está en preparación”**

Hace que el robot avise que el combo está en preparación

**AL EMPEZAR A EJECUTAR**

**| AVISAR QUE EL COMBO ESTÁ LISTO SI ESTE EFECTIVAMENTE LO ESTÁ**

**DEFINIR AVISAR QUE EL COMBO ESTÁ LISTO SI ESTE EFECTIVAMENTE LO ESTÁ**

**| SI ¿está listo el combo? ENTONCES**

**| Avisar que “el combo está listo”**

**SINO**

**| Avisar que “el combo está en preparación”**

ASI ESTARIA BIEN?

## Ejercicio 2) Operación Nuevo Amanecer

Jorge Arbusto Caminante llegó al poder del país más beligerante del mundo, y está dispuesto a todo para conseguir grandes negocios para su nación a través de la guerra. Jorge se puso a analizar el mapa para ver qué naciones invadir para robar sus recursos naturales. El escenario tiene siete países representados, y pueden o no tener recursos naturales, ya sea petróleo (representado en el mapa por una torre petrolera) o diamantes (representado por un diamante). Además un país puede o no estar colonizado previamente por el país de Jorge (representado por una pequeña bandera). A continuación hay un escenario de ejemplo.

$\wedge=Y \rightarrow \text{NO } V=O$



Jorge quiere invadir todos aquellos países que no estén previamente colonizados y que tengan recursos naturales, para lo cual le pide a usted que escriba un procedimiento **Marcar países interesantes para invadir**. En este ejemplo serían Creta, Aerugo, Desertis y Xian, pero tenga en cuenta que el mapa podría variar, siendo otros los países colonizados y otros los recursos en los países, algo que su procedimiento deberá contemplar.

**Aclaración:** Al inicio, se está mirando un país, aunque no sepamos a cual. Las primitivas a utilizar son las siguientes:

**Mirar siguiente país**

Cambia el país que se está mirando actualmente, al siguiente. Falla sí ya se han mirado 7 países.

**Marcar país para invadir**

Marca el país que se está mirando actualmente como interesante para invadir.

**¿tiene petróleo?**

Indica sí el país que se está mirando tiene petróleo.

**¿tiene diamantes?**

Indica sí el país que se está mirando tiene diamantes.

**¿está colonizado?**

Indica sí el país que se está mirando está previamente colonizado.

Notar que Jorge está mirando un único país a la vez, y debe ir cambiando el país del mapa que está mirando para determinar si conviene o no invadirlo. Recordar que independientemente de cuál sea el mapa en cuestión, el mismo representa exactamente siete países.

**Pista:** Realice definiciones que simplifiquen cuándo un país debe ser invadido y cuándo no, tantas como le sean convenientes.

$\wedge = Y$   $\neg = \text{NO}$   $\vee = \text{O}$

AL EMPEZAR A EJECUTAR

| MARCAR PAISES INTERESANTES PARA INVADIR

DEFINIR MARCAR PAISES INTERESANTES PARA INVADIR

| REPETIR 6 VECES

| | MARCAR PAIS PARA INVADIR SI ES INTERESANTE

| | MIRAR SIGUIENTE PAIS

| MARCAR PAIS SI ES INTERESANTE

DEFINIR MARCAR PAIS SI ES INTERSANTE

| SI ¿DEBE INVADIR? ENTONCES

| | MARCAR PAIS PARA INVADIR

ASI ESTARIA  
MAL? Una  
expresión con  
parentesis

DEFINIR ¿DEBE INVADIR?  $= \neg$  ¿ESTA COLONIZADO?  $\wedge$  (¿TIENE PETRÓLEO?  $\vee$  ¿TIENE DIAMANTES?)

OPCIÓN MÁS ADECUADA :

DEFINIR ¿TIENE PETROLEO SIN COLONIZAR?  $=$  ¿TIENE PETRÓLEO?  $\wedge$   $\neg$  ¿ESTA COLONIZADO?

DEFINIR ¿TIENE DIAMANTES SIN COLONIZAR?  $=$  ¿TIENE DIAMANTES?  $\wedge$   $\neg$  ¿ESTA COLONIZADO?

DEFINIR ¿DEBE INVADIR?  $=$  ¿TIENE PETROLEO SIN COLONIZAR?  $\vee$  ¿TIENE DIAMANTES SIN COLONIZAR?

```

1  2  3  4  5  6  7  8  9  10 11 12 13 14 15 16 17 18 19
Al empezar a ejecutar
| Realizar operación "nuevo amanecer"

Definir ¿tiene diamantes sin colonizar? =
  ¿tiene diamantes?  $\wedge$   $\neg$  ¿está colonizado?

Definir ¿tiene petróleo sin colonizar? =
  ¿tiene petróleo?  $\wedge$   $\neg$  ¿está colonizado?

Definir ¿se debe invadir? =
  ¿tiene diamantes sin colonizar?  $\vee$  ¿tiene petróleo sin colonizar?

Definir Realizar operación "nuevo amanecer"
| Repetir 6 veces
| | Marcar pais para invadir si corresponde
| | Mirar siguiente pais
| | Marcar pais para invadir si corresponde

Definir Marcar pais para invadir si corresponde
| Si ¿se debe invadir?
| | Marcar pais para invadir
  
```

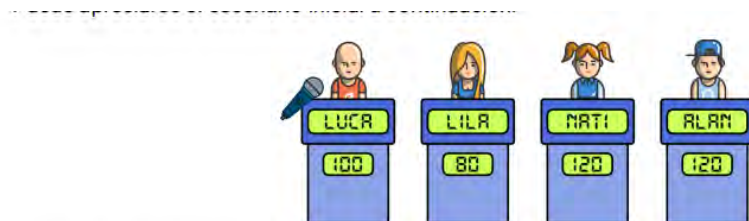
### Ejercicio 3) ¿Quién quiere ser multi-millonario?

Esta vez vamos a manipular al micrófono del presentador de "¿Quién quiere ser multi-millonario?", un popular juego de la televisión con preguntas y respuestas, donde los ganadores se llevan premios multi-millonarios (muchos más altos que otros programas competidores que no adecuaron sus títulos a las sucesivas devaluaciones). El juego es muy simple. El micrófono se va moviendo entre los diversos participantes, para darles lugar a que contesten la pregunta realizada por el presentador. Cada participante emite su respuesta. Las preguntas son todas de "sí" o "no". Cuando un participante sabe la respuesta, la contesta de forma adecuada, sí no la sabe, contesta dependiendo de sí le gusta más el sí o el no. Luego se otorgarán puntos dependiendo de sí la respuesta del participante fue correcta o no, a razón de 20 puntos por pregunta contestada correctamente. Vamos a modelar una ronda de este juego.



$\wedge = Y \quad \neg = \text{NO} \quad \vee = \text{O}$

para cuatro participantes puntuales, Luca, Lila, Nati y Alan. Puede apreciarse el escenario inicial a continuación:



Los sensores son los siguientes:

<b>¿Prefiere contestar con "sí"?</b> Indica si el concursante donde está el micrófono prefiere contestar con "sí". El concursante no debe saber la respuesta a la pregunta.	
<b>¿Sabe la respuesta?</b> Indica si el concursante donde está el micrófono sabe la respuesta.	<b>¿La respuesta es "sí"?</b> Indica si la respuesta a la pregunta es "sí".

Se pide en primer lugar que defina expresiones que respondan:

a) **¿Sabe la respuesta y es "sí"?** (Indica que el jugador sabe la respuesta, y la respuesta es "sí")

**DEFINIR ¿SABE LA RESPUESTA Y ES "SÍ"? = ¿SABE LA RESPUESTA?  $\wedge$  ¿LA RESPUESTA ES SÍ?**

b) **¿Sabe la respuesta y es "no"?** (Indica que el jugador sabe la respuesta, y la respuesta es "no")

**DEFINIR ¿SABE LA RESPUESTA Y ES "NO"? = ¿SABE LA RESPUESTA?  $\wedge$   $\neg$  ¿LA RESPUESTA ES SÍ?**

c) **¿Le atina sí responde con "sí"?** (Indica que el jugador no sabe la respuesta, pero prefiere contestar con "sí", y la respuesta justo es "sí")

**DEFINIR ¿LE ATINA SÍ RESPONDE CON "SÍ"? = ¿PREFIERE CONTESTAR CON SÍ?  $\wedge$  ¿LA RESPUESTA ES SÍ?**

d) **¿Le atina sí responde con "no"?** (Indica que el jugador no sabe la respuesta, pero prefiere contestar con "no", y la respuesta justo es "no")

**DEFINIR ¿LE ATINA SÍ RESPONDE CON "NO"? =  $\neg$  ¿PREFIERE CONTESTAR CON SÍ?  $\wedge$   $\neg$  ¿LA RESPUESTA ES SÍ?**

e) **¿Responde con "sí" de forma correcta?** (Indica que el jugador responderá con "sí", ya sea porque la sabe o porque no la sabe pero prefiere contestar con "sí", y la respuesta es "sí")

**DEFINIR ¿Responde con "sí" de forma correcta? = (¿PREFIERE CONTESTAR CON SÍ?  $\vee$  ¿SABE LA RESPUESTA? )  $\wedge$   $\neg$  ¿LA RESPUESTA ES SÍ?**

f) **¿Responde con "no" de forma correcta?** (Indica que el jugador responderá con "no", ya sea porque la sabe o porque no la sabe pero prefiere contestar con "no", y la respuesta es "no")

**DEFINIR ¿Responde con "NO" de forma correcta? = ( $\neg$ ¿PREFIERE CONTESTAR CON SÍ?  $\vee$   $\neg$ ¿SABE LA RESPUESTA? )  $\wedge$   $\neg$  ¿LA RESPUESTA ES SÍ?**

Considerando ahora las siguientes primitivas:

$\wedge = Y$   $\neg = NO$   $\vee = O$

**Ir a siguiente concursante** Hace que el micrófono se mueva al siguiente concursante, a la derecha. Falla sí se está en el concursante más a la derecha.

### Otorgar un punto

Hace que se le otorgue un punto al concursante sobre el cual está el micrófono.

### Contestar con "sí"

Hace que el concursante donde está el micrófono conteste con "sí".

### Contestar con "no"

Hace que el concursante donde está el micrófono conteste con "no".

Se pide que **escriba un programa que haga que todos los participantes respondan**, según sí la saben o según su preferencia. Sí luego de contestar, la respuesta coincide con lo que el concursante contestó, se otorgan 20 puntos a dicho concursante. Sí utilice las conectivas de forma apropiada y las definiciones anteriores, su código debería tener una única alternativa

### AL EMPEZAR A EJECUTAR

| HACER RESPONDER A TODOS LOS PARTICIPANTES

### DEFINIR HACER RESPONDER A TODOS LOS PARTICIPANTES

```
|
|   | REPETIR 3 VECES
|   |   | HACER RESPONDER A UN PARTICIPANTE Y OTORGARLE PUNTOS SI CORRESPONDE
|   |   | IR A SIGUIENTE CONCURSANTE
|   |   | HACER RESPONDER A UN PARTICIPANTE Y OTORGARLE PUNTOS SI CORRESPONDE
```

### DEFINIR HACER RESPONDER A UN PARTICIPANTE Y OTORGARLE PUNTOS SI CORRESPONDE

```
|   | SI ¿RESPONDE CORRECTAMENTE CON SI? V ¿RESPONDE CORRECTAMENTE CON NO?
|   |   | ENTONCES
|   |   |   | OTORGAR 20 PUNTOS
```

DEFINIR ¿RESPONDE CORRECTAMENTE CON SI? = ¿SABE LA RESPUESTA Y ES "SI"? V ¿Le atina si responde con "sí"? V ¿Responde con "sí" de forma correcta?

DEFINIR ¿RESPONDE CORRECTAMENTE CON NO? = ¿SABE LA RESPUESTA Y ES "NO"? V ¿Le atina si responde con "no"? V ¿Responde con "no" de forma correcta?

### DEFINIR OTORGAR 20 PUNTOS

```
|   | REPETIR 20 VECES
|   |   | OTORGAR PUNTO
```

### LA RESOLUCIÓN DEL PROFE

```
Al empezar a ejecutar
| Hacer que todos los concursantes respondan

Definir ¿Sabe la respuesta y es "sí"? =
    ¿Sabe la respuesta?  $\wedge$  ¿La respuesta es "sí"?

Definir ¿Sabe la respuesta y es "no"? =
    ¿Sabe la respuesta?  $\wedge$   $\neg$  ¿La respuesta es "sí"?

Definir ¿Le atina si responde con "sí"? =
     $\neg$  ¿Sabe la respuesta?  $\wedge$  ¿Prefiere contestar con "sí"?

Definir ¿Le atina si responde con "no"? =
     $\neg$  ¿Sabe la respuesta?  $\wedge$   $\neg$  ¿Prefiere contestar con "sí"?
     $\wedge$   $\neg$  ¿La respuesta es "sí"?

Definir ¿Responde con "sí" de forma correcta? =
    ¿Sabe la respuesta y es "sí"?  $\vee$  ¿Le atina si responde con "sí"?

Definir ¿Responde con "no" de forma correcta? =
    ¿Sabe la respuesta y es "no"?  $\vee$  ¿Le atina si responde con "no"?
```

```
Definir ¿Responde con "sí" de forma correcta? =
    ¿Sabe la respuesta y es "sí"?  $\vee$  ¿Le atina si responde con "sí"?

Definir ¿Responde con "no" de forma correcta? =
    ¿Sabe la respuesta y es "no"?  $\vee$  ¿Le atina si responde con "no"?

Definir ¿El concursante responde correctamente? =
    ¿Responde con "sí" de forma correcta?
     $\vee$  ¿Responde con "no" de forma correcta?

Definir Hacer que todos los concursantes respondan
| Repetir 3 veces
|   | Hacer que el concursante responda y otorgarle puntos si corresponde
|   |   | Ir al siguiente concursante
|   | Hacer que el concursante responda y otorgarle puntos si corresponde

Definir Hacer que el concursante responda y otorgarle puntos si corresponde
| Si ¿El concursante responde correctamente?
|   | Otorgar 20 puntos

Definir Otorgar 20 puntos
| Repetir 20 veces
|   | Otorgar un punto
```

## Ejercicio 4) Al infinito y más allá

La Alianza Galáctica ha decidido mandar a explorar el espacio profundo, en busca de nuevos planetas colonizables. Por supuesto, a cientos de años luz de distancia, sólo una sonda de exploración automatizada podría determinar qué planetas son habitables y cuáles no, ya que enviar personas a realizar el trabajo no es viable. Su misión: programar a dicha sonda previo a su partida. Por suerte hay vasta información disponible previo al envío de la sonda de exploración. Sabemos que el destino es un sistema planetario **formado por 15 planetas**. De estos, **algunos están en la zona habitable**, a una distancia adecuada de la estrella que habitan, para no ser ni muy fríos ni muy cálidos), y otro por supuesto, no. **Algunos tienen agua, y atmósfera**, pudiendo albergar vida, y otros no. **Algunos tienen minerales valiosos, incluso sin ser habitables** y otros son solo un montón de polvo. **Algunos pueden recibir meteoritos de forma constante, y otros no.**

Su objetivo es **lograr que la sonda determine y catalogue los planetas** en una de las siguientes categorías:

- **Habitable:** El planeta debe tener atmósfera y agua, y no debe estar sometido a meteoritos.
- **Explotable:** El planeta debe tener minerales valiosos y no estar sometido a meteoritos.
- **Colonizables:** El planeta debe poder ser habitado o explotado y debe tener una temperatura de entre  $-50^{\circ}$  y  $50^{\circ}\text{C}$  inclusive.

Contando con los siguientes sensores y primitivas, se pide que elabore un programa que haga que la sonda **catalogue todos los planetas del sistema solar**. Tenga en cuenta que **al comenzar, la sonda no se encuentra sobrevolando ningún planeta**. Notar que un planeta puede ser catalogado en más de una categoría.

<b>Sobrevolar siguiente planeta</b> Hace que la sonda sobrevuele el próximo planeta. Debe haber un próximo planeta para sobrevolar.	<b>Catalogar como "habitable"</b> Cataloga el planeta que sobrevuela la sonda como "habitable". La sonda debe estar sobrevolando un planeta.
<b>Catalogar como "explotable"</b> Cataloga el planeta que sobrevuela la sonda como "explotable". La sonda debe estar sobrevolando un planeta.	<b>Catalogar como "colonizable"</b> Cataloga el planeta que sobrevuela la sonda como "colonizable". La sonda debe estar sobrevolando un planeta.
<b>¿tiene temperatura mayor a <math>50^{\circ}</math>?</b> Indica si el planeta que sobrevuela la sonda tiene una temperatura mayor a $50^{\circ}$ . La sonda debe estar sobrevolando un planeta.	<b>¿tiene temperatura menor a <math>-50^{\circ}</math>?</b> Indica si el planeta que sobrevuela la sonda tiene una temperatura menor a $-50^{\circ}$ . La sonda debe estar sobrevolando un planeta.
<b>¿tiene atmósfera?</b> Indica si el planeta que sobrevuela la sonda tiene una atmósfera. La sonda debe estar sobrevolando un planeta.	<b>¿tiene agua?</b> Indica si el planeta que sobrevuela la sonda tiene agua. La sonda debe estar sobrevolando un planeta.
<b>¿tiene minerales valiosos?</b> Indica si el planeta que sobrevuela la sonda tiene minerales valiosos. La sonda debe estar sobrevolando un planeta.	<b>¿sufre bombardeo de meteoritos?</b> Indica si el planeta que sobrevuela la sonda sufre de bombardeo de meteoritos. La sonda debe estar sobrevolando un planeta.



$\wedge = Y$   $\neg = \text{NO}$   $\vee = \text{O}$

AL EMPEZAR A EJECUTAR

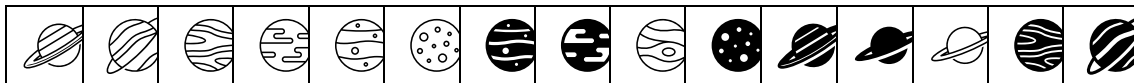
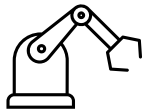
| CATALOGAR TODOS LOS PLANETAS DEL SISTEMA SOLAR

DEFINIR ¿ES HABITABLE? = ¿TIENE ATMOSFERA?  $\wedge$  ¿TIENE AGUA?  $\wedge$   $\neg$  ¿SUFRE BOMBARDEOS DE METEORITOS?

DEFINIR ¿ES EXPLOTABLE? = ¿TIENE MINERALES VALIOSOS?  $\wedge$   $\neg$  ¿SUFRE BOMBARDEOS DE METEORITOS?

DEFINIR ¿TIENE LA TEMPERATURA ADECUADA? =  $(\neg \text{¿TIENE TEMPERATURA MAYOR A } 50^\circ) \wedge (\neg \text{TIENE TEMPERATURA A } -50^\circ)$

DEFINIR ¿ES COLONIZABLE? =  $(\text{¿ES HABITABLE?} \vee \text{¿ES EXPLOTABLE?}) \wedge \text{¿TIENE LA TEMPERATURA ADECUADA?}$



DEFINIR CATALOGAR TODOS LOS PLANETAS DEL SISTEMA SOLAR

| REPETIR 15 VECES

| SOBREVOLAR SIGUIENTE PLANETA

| CATALOGAR PLANETA

DEFINIR CATALOGAR PLANETA

| CATALOGAR PLANETA COMO HABITABLE SI LO ES

| CATALOGAR PLANETA COMO EXPLOTABLE SI LO ES

| CATALOGAR PLANETA COMO COLONIZABLE SI LO ES

DEFINIR CATALOGAR PLANETA COMO HABITABLE SI LO ES

| SI ¿ES HABITABLE? ENTONCES

| CATALOGAR COMO HABITABLE

DEFINIR CATALOGAR PLANETA COMO EXPLOTABLE SI LO ES

| SI ¿ES EXPLOTABLE? ENTONCES

| CATALOGAR COMO "EXPLOTABLE"

DEFINIR CATALOGAR PLANETA COMO COLONIZABLE

| SI ¿ES COLONIZABLE?

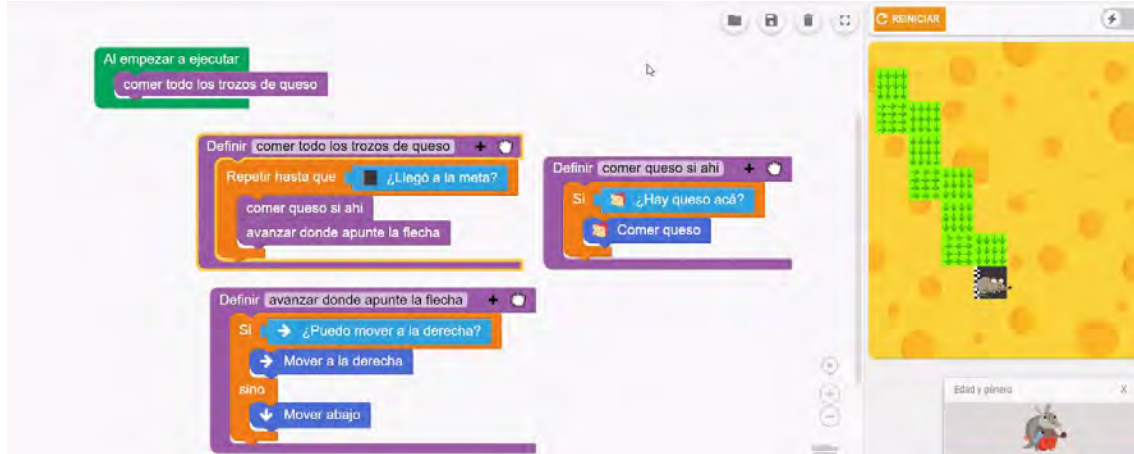
| CATALOGAR COMO "COLONIZABLE"

$\wedge = Y$   $\neg = \text{NO}$   $\vee = \text{O}$

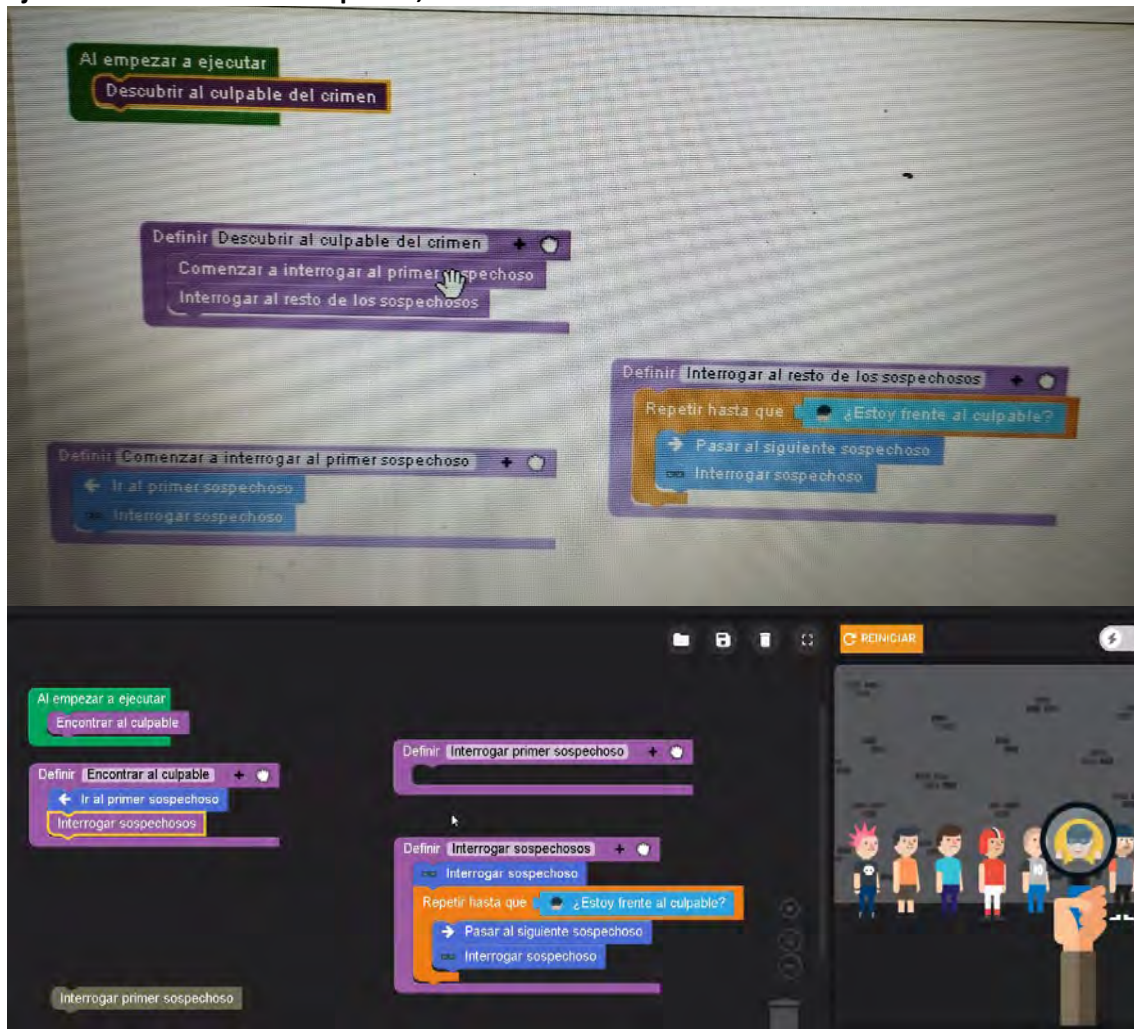
## UNIÓN REPETICIÓN CONDICIONAL-FILAS BLOQUES

Realizamos del libro del “Ciclo de Secundaria” de la sección “Repetición Condicional”

### Ejercicio 1 “Laberinto con queso”,

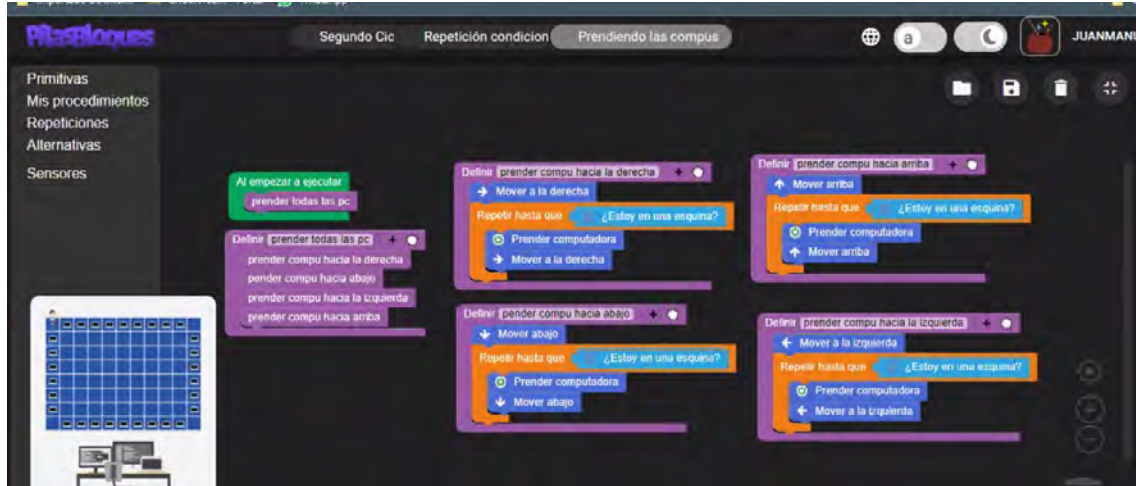


### Ejercicio 2 “El detective chaparro”,



### Ejercicio 3 “Fútbol para robots”,

## Ejercicio 4 “Prendiendo las compus”



## Ejercicio 5 “El mono que sabe contar”

## NUEVA UNIÓN SEGUIMOS REPETIENDO REPETICIONES CONDICIONALES

### Ejercicio 1) ¿Cuándo termina esto?

Vamos a analizar bajo qué condiciones se ejecuta y termina una repetición condicional. Para ello, se pide que analice las siguientes secciones de código y determine qué condición se cumple seguro dentro del cuerpo de la repetición condicional y cuál seguro se cumple a posteriori de la repetición condicional.

- a) Repetir hasta que ¿llegamos a la cima?

Mientras el autómata está dentro de la repetición (significa que no llego a la cima). Por eso se va a repetir o cumplir hasta que llegamos a la cima. Una vez en la cima no se sigue repitiendo. Si la repetición termino nuestro autómata llego a la cima

- b) Repetir hasta que  $\neg$  ¿llegamos a la cima?

Se cumple/repite hasta que no lleguemos a la cima. Mientras que el autómata está dentro de la repetición no estamos en la cima. Siempre que el autómata no este en la cima se va repetir.

- c) Repetir hasta que ¿puede subir un piso?

Mientras se ejecuta la repetición seguramente no puede subir el piso. Cuando termina la ejecución el autómata subió al piso

- d) Repetir hasta que  $\neg$  ¿puede subir un piso?

Si se esta repitiendo el ciclo el automata puede seguir subiendo. Pero cuando sale de la repetición no puede subir un piso

- e) Repetir hasta que ¿hay bananas?  $\wedge$  ¿hay manzanas?

Mientras el autómata está dentro de la repetición (significa que no hay bananas ni manzanas).

Repita hasta que haya banana y manzana (las dos a la vez)

$\wedge = Y \quad \neg = \text{NO} \quad \vee = \text{O}$

f) Repetir hasta que ¿hay bananas?  $\vee$  ¿hay manzanas?

Mientras el autómata está dentro de la repetición (significa que hay bananas o manzanas).

Repite hasta que haya banana o manzana

Va a seguir repitiendo cuando no este ninguna de las dos

g) Repetir hasta que  $\neg$  (¿hay bananas?  $\wedge$  ¿hay manzanas?)

Repite hasta que no haya banana ni manzana (las dos a la vez)

h) Repetir hasta que  $\neg$  ¿hay bananas?  $\wedge$   $\neg$  ¿hay manzanas?

Repite hasta que no haya banana o manzana (puede que falte la banana o que falte la manzana)

## Ejercicio 2) Un lugar jamás visitado por el hombre

El capitán Kirk y la tripulación del U.S.S. Enterprise se han aventurado en una misión al insondable espacio más allá de los límites del universo conocido. Solo ellos pueden llegar a “un lugar jamás visitado por el hombre” (entiéndase “hombre” por homo sapiens, no como género). Para eso, han cruzado un puente de Einstein-Rosen que los ha dejado en algún lugar indeterminado del espacio. El objetivo de la U.S.S. Enterprise a llegar al planeta Altair VI que se sabe se encuentra al frente del camino.

El problema: la indeterminada distancia a la que se encuentra dicho planeta, ya que no sabemos dónde salimos del puente de Einstein-Rosen. El escenario inicial puede variar, y por tanto la nave puede estar ya sobre Altair VI, o a una sola ubicación, o a dos, o a mil, o a un millón. A continuación se muestran 3 posibles escenarios iniciales, pero tenga en cuenta que podría ser cualquiera de los mencionados arriba.



Las primitivas: sólo un comando primitivo y un sensor. **Avanzar Enterprise** (que mueve a la Enterprise un lugar a la derecha) y **¿se está sobre Altair VI?** (que indica que efectivamente la Enterprise está sobre Altair VI)

La misión: Escribir un procedimiento **Llegar a Altair VI** que lleve a la Enterprise a Altair VI, independientemente de cuál sea la distancia a la que se encuentre el mismo.

**AL EMPEZAR A EJECUTAR**

**| LLEGAR A ALTAIR VI**

**DEFINIR LLEGAR A ALTAIR VI**

**| REPETIR HASTA QUE ¿SE ESTÁ SOBRE ALTAIR VI?**

**| AVANZAR ENTERPRISE**

CORRECCIÓN HECHA EN  
CLASE 😊

## Ejercicio 3) Quién me ha robado el mes de Abril

No sabemos si decidimos por team verano o team invierno, pero lo que sí es seguro es que odiamos el otoño. Cuando llega Abril y tenés un árbol en la puerta de tu casa, te vas a aburrir



$\wedge = Y$   $\neg = \text{NO}$   $\vee = \text{O}$

de barrer hojas de hermosos tonos naranja y amarillo, tanto que va a parecer que te pasaste el mes barriendo. Eso es lo que le sucede siempre a Joaquín, que tiene un hermoso maple en la puerta de su casa, que no deja de tirar hojas. Joaquín quiere que **programemos un robot que haga el arduo trabajo de juntar las hojas por él**, y no lo podemos decepcionar, pues de lo contrario tiende a llorar unos 19 días y unas 500 noches.

El robot que construimos tiene las siguientes primitivas y sensores:

<b>¿hay hojas en el árbol?</b> Indica si aún hay hojas en el árbol que aún faltan caer.	<b>¿hay hojas en el suelo?</b> Indica si hay hojas en el suelo que aún faltan juntar.
<b>Sacudir árbol</b> Sacude el árbol, haciendo que caiga una o más hojas del mismo hacia el suelo. Cada vez que se sacude el árbol habrá menos hojas en el mismo, y más hojas en el suelo.	<b>Juntar hoja del suelo a la bolsa</b> Hace que el robot junte una hoja del suelo en la bolsa de residuos. Cada vez que se juntan hojas habrá menos en el suelo.
<b>Cerrar bolsa</b> Cierra la bolsa de residuos dejándola lista para colocar en la basura.	<b>Dejar bolsa en la basura</b> Deja la bolsa de residuos en la basura. La bolsa debe estar cerrada para poder dejarla allí.

Solo nos falta escribir el programa principal del robot. El objetivo es escribir un **programa que tire todas las hojas del árbol al suelo (para que no sigan cayendo todo el tiempo), luego junte las hojas en la bolsa de residuos, para finalmente dejar la bolsa cerrada en la basura.**

## AL EMPEZAR A EJECUTAR

| JUNTAR TODAS LAS HOJAS Y TIRARLAS A LA BASURA

## DEFINIR JUNTAR TODAS LAS HOJAS

| TIRAR TODAS LAS HOJAS DEL ARBOL  
| JUNTAR TODAS LAS HOJAS DEL SUELO  
| TIRAR LAS HOJAS A LA BASURA

CORRECCIÓN HECHA EN  
CLASE 😊 OK

## DEFINIR TIRAR TODAS LAS HOJAS DEL ARBOL

| REPETIR HASTA QUE  $\neg$  ¿HAY HOJAS EN EL ARBOL?  
| SACUDIR ARBOL

## DEFINIR JUNTAR TODAS LAS HOJAS DEL SUELO

REPETIR HASTA QUE  $\neg$  ¿HAY HOJAS EN EL SUELO?  
| JUNTAR HOJA DEL SUELO A LA BOLSA

## DEFINIR TIRAR LAS HOJAS A LA BASURA

| CERRAR BOLSA  
| DEJAR BOLSA A LA BASURA

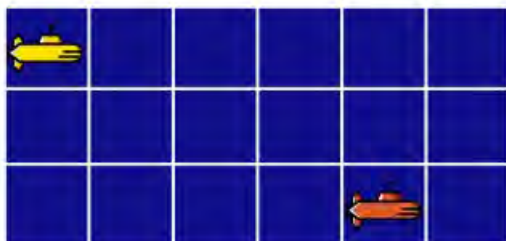
## Ejercicio 4) A la caza del octubre rojo

El Octubre Rojo es un modernísimo submarino que puede “desaparecer” bajo el agua, sin ser detectado por los sonares tradicionales. Así, la única forma de **“encontrar” al submarino, es acercándose lo suficiente al mismo (una ubicación de distancia) y activando el sonar.** Ahí es donde entra el U.S.S. Dallas, un submarino que buscará incansablemente al Octubre Rojo para darle caza. En esta actividad nuestro autómatas será el U.S.S. Dallas, a quien podremos comandar para buscar al Octubre Rojo y destruirlo, usando las siguientes primitivas:

$\wedge = Y$   $\neg = \text{NO}$   $V = O$

<b>Mover submarino abajo</b> Mueve al U.S.S. Dallas una ubicación hacia abajo de la actual. Falla si no existe una ubicación abajo o si la misma se encuentra ocupada.	<b>Disparar Torpedo</b> Hace que el U.S.S. Dallas lance un torpedo hacia la derecha para destruir al Octubre Rojo. Solo se puede disparar un torpedo si efectivamente el Octubre Rojo está en la ubicación de la derecha del U.S.S. Dallas.
<b>Mover submarino a la derecha</b> Mueve al U.S.S. Dallas una ubicación hacia la derecha de la actual. Falla si no existe una ubicación a la derecha o si la misma se encuentra ocupada.	<b>Mover submarino al borde izquierdo</b> Mueve al U.S.S. Dallas hacia la ubicación del borde izquierdo de la fila actual. No falla nunca.
<b>¿el sonar detecta al Octubre Rojo delante?</b> Indica si el Octubre Rojo se encuentra en la ubicación inmediatamente a la derecha del U.S.S. Dallas.	<b>¿hay ubicación a la derecha?</b> Indica si el U.S.S. Dallas es capaz de moverse a la derecha.

El **escenario inicial** cumple las siguientes características. El U.S.S. Dallas (el submarino de amarillo) **siempre comienza en la esquina superior izquierda del escenario**. El **tamaño del escenario es variable**, pudiendo tener cualquier cantidad de filas y columnas. El Octubre Rojo (submarino Rojo) puede encontrarse en cualquier ubicación del escenario, salvo en la columna más a la izquierda. El siguiente es un escenario de ejemplo:



```

Al empezar a ejecutar
| Dar caza al Octubre Rojo

Definir Dar caza al Octubre Rojo
| Repetir hasta que ¿el sonar detecta al Octubre Rojo delante?
| | Avanzar en búsqueda del Octubre Rojo
| | Disparar torpedo

Definir Avanzar en búsqueda del Octubre Rojo
| Si ¿hay ubicación a la derecha?
| | Mover submarino a la derecha
| Sino
| | Mover submarino al borde izquierdo
| | Mover submarino abajo

```

**AL EMPEZAR A EJECUTAR**

| **BUSCAR Y DESTRUIR AL OCTUBRE ROJO**

**DEFINIR BUSCAR Y DESTRUIR AL OCTUBRE ROJO**

| **REPETIR HASTA QUE ¿EL SONAR DETECTA EL OCTUBRE ROJO DELANTE?**

| **AVANZAR Y BUSCAR EL OCTUBRE ROJO**

| **DISPARAR TORPEDO**

**DEFINIR AVANZAR Y BUSCAR EL OCTUBRE ROJO**

| **SI ¿HAY UBICACIÓN A LA DERECHA?**

| **MOVER A LA DERECHA**

| **SINO**

| **MOVER SUBMARINO A LA IZQUIERDA**

| **MOVER SUBMARINO ABAJO**

DUDAS:

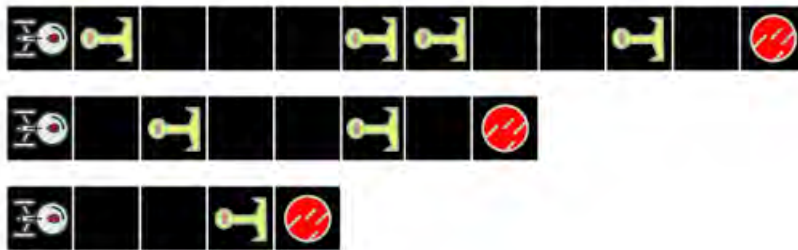
XQ MUEVE 1° A LA IZQUIERDA (XQ NUNCA FALLA?)

Lo que se pide es un programa que mueva al U.S.S. Dallas para que éste destruya al Octubre Rojo. Se sugiere pensar una estrategia para moverse “una ubicación a la vez” hasta encontrar al Octubre Rojo, y luego disparar los torpedos.

$\wedge = Y$   $\neg = \text{NO}$   $V = O$

## Ejercicio 5) Un lugar jamás visitado por el hombre... pero si por los Klingons

La Enterprise quiere realizar un nuevo viaje a Altair VI, pero esta vez no están solos. Los Klingons, un imperio de alienígenas beligerantes ya han llegado a esa región del espacio. Sus naves ahora se encuentran por todos lados. En el escenario podemos encontrar entonces, en el camino hacia Altair, a las naves Klingon, que seguro no están en la ubicación en donde parte la Enterprise ni sobre el planeta Altair VI, pero que sí pueden estar en cualquier otra ubicación (pero no sabemos exactamente en cuáles, ya que el escenario es variable en dicho sentido). A continuación se muestra un ejemplo de algunos posibles escenarios:



Las primitivas: la anteriores. **Avanzar Enterprise** (que mueve a la Enterprise un lugar a la derecha) y **¿se está sobre Altair VI?** (que indica que efectivamente la Enterprise está sobre Altair VI) a las que se suma **Destruir nave Klingon** (que destruye la nave Klingon en la ubicación en donde está la Enterprise) y el sensor **¿hay nave Klingon?** (que indica si hay una nave Klingon en la ubicación donde está la Enterprise).

La misión: Escribir un procedimiento Llegar a Altair VI destruyendo Klingons que lleve a la Enterprise a Altair VI, independientemente de cuál sea la distancia a la que se encuentre el mismo, destruyendo cada una de las naves Klingons en el trayecto.

## ESCRIBIR PROCEDIMIENTO

### DEFINIR LLEGAR A ALTAIR VI DESTRUYENDO KLINGON

REPETIR HASTA QUE ¿SE ESTA SOBRE ALTAIR VI?

AVANZAR ENTERPRISE DESTRUYENDO NAVES KLINGON

### DEFINIR AVANZAR ENTERPRISE DESTUYENDO NAVES KLING

REPETIR HASTA QUE ¿HAY NAVE KLINGON?

AVANZAR ENTERPRISE

DESTRUIR NAVE KLINGON

TENGO DUDAS!!!

DEBERIA USAR HASTA QUE

O CONDICIONAL SI SINO

## Ejercicio 6) Limpiando la pintura

Un visitante medio sacado decidió lanzar un pastelazo a la mismísima Mona Lisa. Ya ni el Louvre, ni sus obras de arte, se salva de la locura. Por suerte el museo cuenta con restauradores de arte de alto renombre, como Restaurabot 2000, el maravilloso robot restaurador.

Restaurabot posee un funcionamiento sencillo. Para poder limpiar toda una pintura, la misma **es dividida en** una grilla rectangular, con varias regiones bien definidas, llamadas **celdas**. El robot posee un cabezal, que se encuentra **siempre en una única celda**. El cabezal es **capaz de**

$\wedge=Y \neg=NO V=O$

**limpiar la mugre que encuentre en esa celda.** También se puede mover el cabezal de una celda a otra, en cualquiera de las direcciones. Para poder determinar cuándo la pintura termina se pueden usar sensores que permiten determinar si se puede mover el cabezal hacia una dirección determinada. Las primitivas son:

<b>Mover cabezal arriba</b> Mueve el cabezal de Restaurabot 2000 a la celda arriba de la celda actual.. Debe haber una celda arriba para que no falle..	<b>Mover cabezal abajo</b> Mueve el cabezal de Restaurabot 2000 a la celda abajo de la celda actual.. Debe haber una celda abajo para que no falle..
<b>Mover cabezal a izquierda</b> Mueve el cabezal de Restaurabot 2000 a la celda a la izquierda de la celda actual.. Debe haber una celda a la izquierda para que no falle..	<b>Mover cabezal a derecha</b> Mueve el cabezal de Restaurabot 2000 a la celda a la derecha de la celda actual.. Debe haber una celda a la derecha para que no falle..
<b>¿se puede mover arriba?</b> Indica si el cabezal se puede mover una celda hacia	<b>¿se puede mover abajo?</b> Indica si el cabezal se puede mover una celda hacia

arriba de su ubicación actual.	abajo de su ubicación actual.
<b>¿se puede mover a izquierda?</b> Indica si el cabezal se puede mover una celda hacia la izquierda de su ubicación actual.	<b>¿se puede mover a derecha?</b> Indica si el cabezal se puede mover una celda hacia la derecha de su ubicación actual.
<b>¿hay mugre?</b> Indica si en la celda actual del cabezal hay mugre que deba ser limpiada.	<b>Limpiar mugre</b> Quita la mugre de la celda actual del cabezal. Debe haber mugre en dicha ubicación.

El término "celda actual" refiere a aquella celda en donde se encuentra ubicado el cabezal al momento de invocar la primitiva o

A continuación se muestra un ejemplo de cómo una pintura se divide en una grilla para su limpieza.



El recuadro verde abajo a la izquierda representa el cabezal.  
En varios lugares de esta pintura hay mugre que debe ser limpiada.

Su trabajo es **realizar el procedimiento Limpiar Pintura** que limpie completamente cualquier pintura (independientemente de su ancho, alto y de los lugares en donde se encuentra sucia).



$\wedge=Y \neg=NO V=O$

**Pista:** piense en cómo recorrer la pintura, lo cual puede lograrse de forma similar a como se ha recorrido escenarios rectangulares en el pasado.

## DEFINIR LIMPIAR PINTURA

```
| REPETIR HASTA QUE ¬¿SE PUEDE MOVER A LA DERECHA?
| LIMPIAR COLUMNA ACTUAL
| IR AL INICIO DE LA COLUMNA SIGUIENTE
| LIMPIAR COLUMNA ACTUAL
```

## DEFINIR LIMPIAR COLUMNA ACTUAL

```
| REPETIR HASTA QUE ¬¿SE PUEDE MOVER ARRIBA?
| LIMPIAR CELDA SI HAY MUGRE ACA
| MOVER CABEZAL ARRIBA
| LIMPIAR CELDA SI HAY MUGRE ACA
```

## DEFINIR LIMPIAR CELDA SI HAY MUGRE ACA

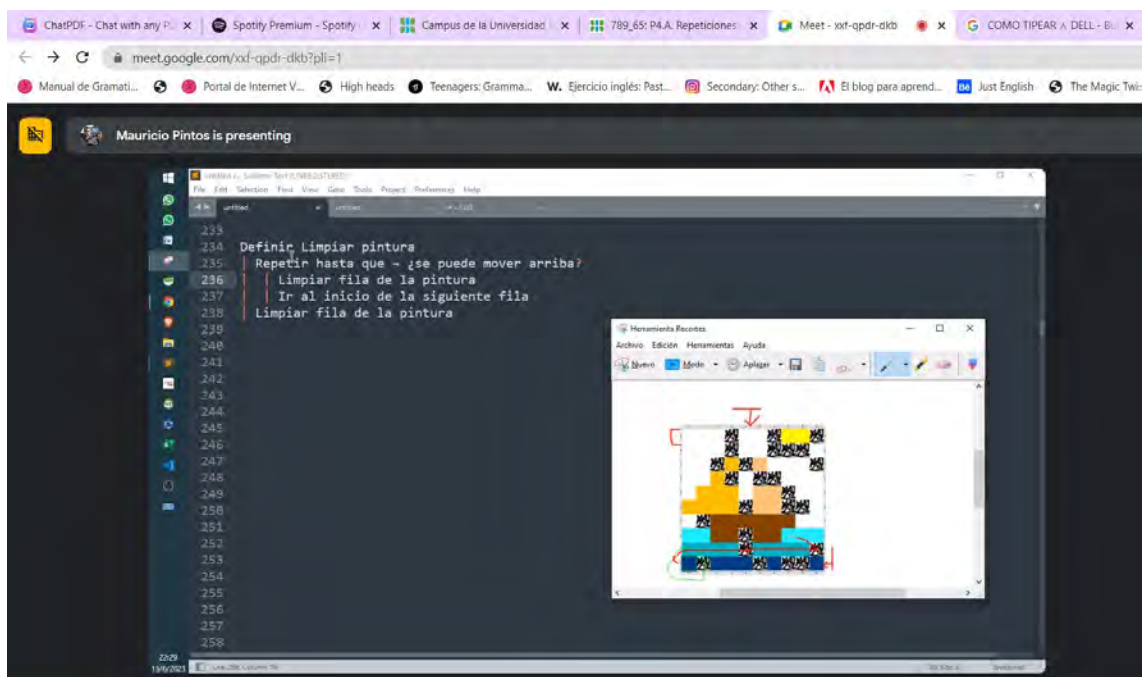
```
| SI ¿HAY MUGRE? ENTONCES
| LIMPIAR MUGRE
```

## DEFINIR IR AL INICIO DE LA COLUMNA SIGUIENTE

```
| IR AL BORDE INFERIOR
| MOVER A LA DERECHA
```

## DEFINIR IR AL BORDE INFERIOR

```
| REPETIR HASTA QUE ¬¿SE PUEDA MOVER ABAJO?
| MOVER CABEZAL ABAJO
```



$\wedge = Y$   $\neg = \text{NO}$   $V = O$

USAR EN  
CASO DE  
NO  
SABER  
DONDE  
ESTA EL  
AUTOMA  
TA

```

Definir Ir a la posicion inicial
| Ir al borde izquierdo
| Ir al borde inferior

Definir Ir al borde izquierdo
| Repetir hasta que ~ ¿se puede mover a la izquierda?
| | Mover cabezal a la izquierda

Definir Ir al borde inferior
| Repetir hasta que ~ ¿se puede mover a abajo?
| | Mover cabezal abajo

Definir Limpiar pintura
| Repetir hasta que ~ ¿se puede mover arriba?
| | Limpiar fila de la pintura
| | Ir al inicio de la siguiente fila
| | Limpiar fila de la pintura

Definir Limpiar fila de la pintura
| Repetir hasta que ~ ¿se puede mover a la derecha?
| | Limpiar mugre si hay acá
| | Mover cabezal a la derecha
| | Limpiar mugre si hay acá

Definir Ir al inicio de la siguiente fila
| Ir al borde izquierdo
| Mover cabezal arriba

Definir Limpiar mugre si hay acá
| Si ¿hay mugre? entonces
| | Limpiar mugre
    
```

## CLASE SENSORES NÚMERICOS

### Ejercicio 1) El pozo de las ánimas

El pozo de las ánimas es una formación geológica que se encuentra en Malargüe, Mendoza, y que consiste, como su nombre lo indica, en un gigantesco pozo. Como turistas curiosos que somos, queremos ir hasta el pozo a sacar una fotografía, pero hay que tener mucha precisión, muy lejos y la fotografía no saldrá bien, un paso de más y podemos caer al pozo y morir solo por sacar una selfie. Así que en lugar de ir nosotros, decidimos programar un robot que vaya hasta el lugar y saque la foto (por supuesto, tampoco queremos que el robot se caiga al pozo), por lo que queremos programarlo.

Las primitivas que este robot entiende son:

#### Acercarse un metro al pozo

Hace que el robot avance hacia el pozo una distancia de un metro.

#### Tomar fotografía

Hace que el robot tome una fotografía.

**cantidad de metros hasta el pozo** Sensor numérico que describe la cantidad de metros que hay desde donde está el robot hasta el pozo de las ánimas (es decir, hasta el punto donde comienza el pozo).

$\wedge = Y$   $\neg = \text{NO}$   $V = O$

El robot debe posicionarse justo **un metro antes** de donde comienza el pozo, tomar la fotografía. Se pide entonces que escriba el procedimiento **Tomar fotografía al pozo de las ánimas** que haga precisamente lo que su nombre indica.

AL EMPEZAR A EJECUTAR

| TOMAR FOTOGRAFIA AL POZO DE LAS ANIMAS

DEFINIR TOMAR FOTOGRAFIA AL POZO DE LAS ANIMAS

| REPETIR (CANTIDAD DE METROS HASTA EL POZO -1) VECES

| | ACERCARSE UN METRO AL POZO

| | TOMAR FOTO

## Ejercicio 2) La peregrinación a Luján

La peregrinación a Luján es una tradicional celebración religiosa en donde diversas congregaciones católicas salen caminando desde Liniers hasta Luján, en un trayecto que implica múltiples kilómetros. En el trayecto hay varias paradas en donde los feligreses pueden detenerse e hidratarse o alimentarse. Nuestra misión es **programar el primer robot que realice la peregrinación a Luján**. Nuestro robot no va a hidratarse, pero sí va a tener que realizar **recambios de aceite y neumáticos en cada parada**. Para ello contamos con un robot con las siguientes primitivas:

**Andar durante un kilómetro** Hace que el robot avance por el camino durante un trayecto de un kilómetro.

**Cambiar neumáticos y aceite** Hace que el robot cambie sus neumáticos y aceite. El robot debe estar en una parada para poder realizar el cambio.

**cantidad de kilómetros hasta llegar a la próxima parada**

Sensor numérico que describe la cantidad de kilómetros que debe recorrer el robot para llegar a la siguiente parada.

**cantidad de paradas en el recorrido** Sensor numérico que describe la cantidad de paradas que hay en el trayecto para llegar a Luján.

Lo que se pide es que realice un procedimiento **Llevar robot hasta Luján** que haga que el robot realice el trayecto hasta Lujan, cambiando el aceite en cada parada, salvo en la última, en donde no hay que hacer nada.

AL EMPEZAR EJECUTAR

| LLEVAR ROBOT HASTA LUJÁN

DEFINIR LLEVAR ROBOT HASTA LUJÁN

| REPETIR (cantidad de paradas en el recorrido-1) VECES

| | ANDAR HASTA PRÓXIMA PARADA

| | CAMBIAR NEUMATICOS Y ACEITE

| | ANDAR HASTA PRÓXIMA PARADA


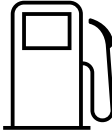
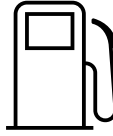

DEFINIR ANDAR HASTA PRÓXIMA PARADA

| REPETIR (CANTIDAD DE KILÓMETROS HASTA LLEGAR A LA PRÓXIMA PARADA) VECES

| | ANDAR DURANTE UN KILÓMETRO

DEBERIA  
CREAR UN  
PROCEDIMIE  
NO PARA  
ESTE  
REPETIR

$\wedge = Y$   $\neg = \text{NO}$   $\vee = \text{O}$

								
---	--	--	--	---	--	--	--	---

### EJERCICIO 3) RESCATANDO AL SOLDADO BRIAN

BRIAN SE ENCUENTRA PERDIDO EN EL FRENTE DE BATALLA. LOS ALEMANES LO TIENEN ESCONDIDO EN LOS COMPLEJOS TÚNELES DEL CASTILLO WOLFENSTEIN, DONDE HAN DEJADO VARIAS MINAS QUE DEBEN SER DESACTIVADAS PREVIO A PODER LLEGAR A BRIAN. AL MENOS ESA ES LA TRATA DE LA NUEVA SUPERPRODUCCIÓN DE JOLLYWOOD, EL LUGAR DONDE SE HACEN LAS MEJORES PELÍCULAS CLASE B DEL MUNDO.

SU OBJETIVO, PROGRAMAR AL ACTOR ROBÓTICO “TOMÁS JANKS”, PARA QUE REALICE LAS PELIGROSAS ESCENAS DE ACCIÓN, QUE IMPLICAN RECORRER EL CASTILLO HASTA LLEGAR DONDE SE ENCUENTRA BRIAN. EN SU CAMINO, EL ROBOT DEBERÁ DESACTIVAR LAS MINAS DEJADAS POR LOS ALEMANES. EL ROBOT CUENTA CON LAS SIGUIENTES PRIMITIVAS:

DAR UN PASO EN EL CAMINO MUEVE EL ROBOT UNA ÚNICA UBICACIÓN EN EL LABERINTO. EL ROBOT NO DEBE HABER LLEGADO A DÓNDE ESTÁ BRIAN.

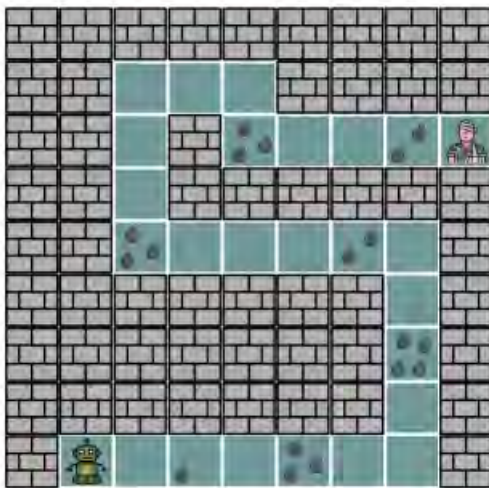
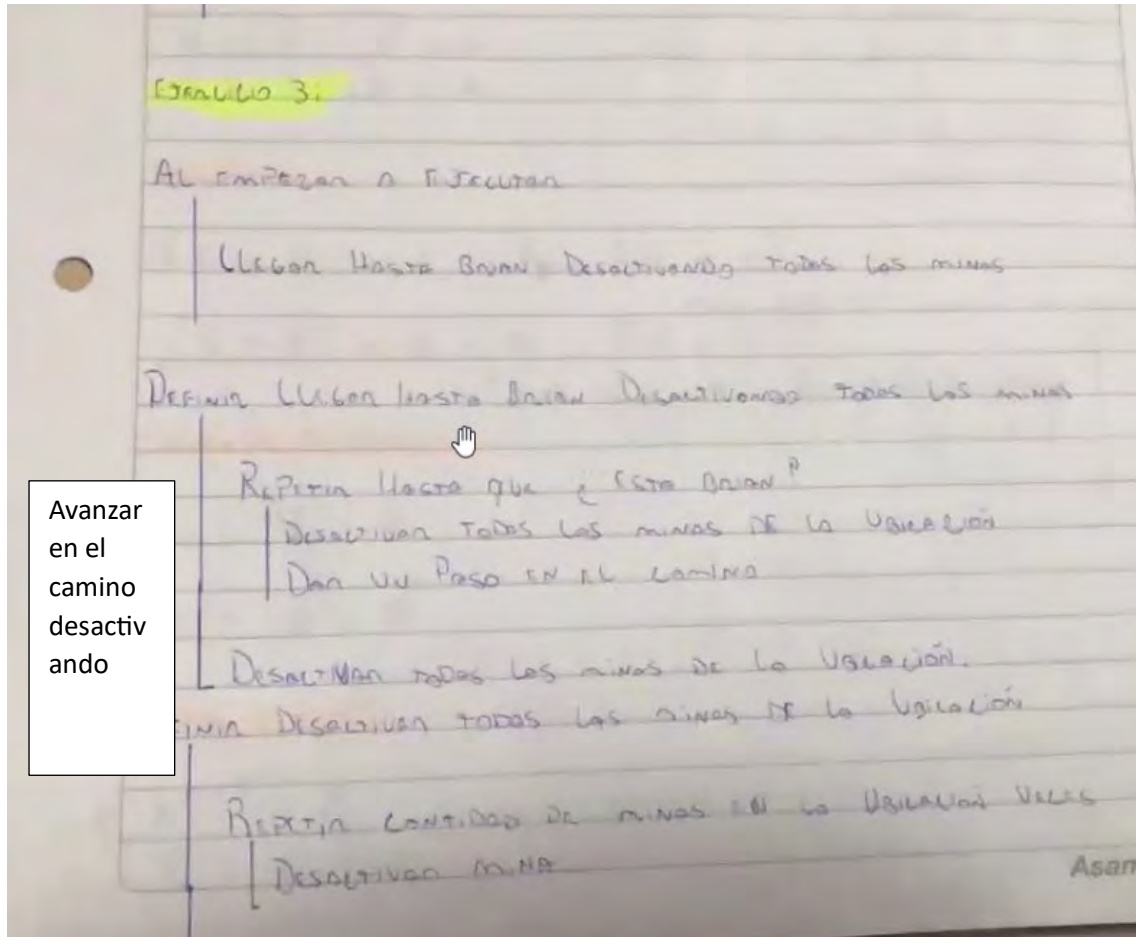
DESACTIVAR MINA HACE QUE EL ROBOT DESACTIVE UNA ÚNICA MINA DE LA UBICACIÓN ACTUAL.

CANTIDAD DE MINAS EN LA UBICACIÓN SENSOR NUMÉRICO QUE DESCRIBE CUÁNTAS MINAS HAY EN LA UBICACIÓN DONDE SE ENCUENTRA EL ROBOT.

¿ESTÁ BRIAN? INDICA SÍ BRIAN ESTÁ EN LA UBICACIÓN ACTUAL.

EL ESCENARIO CONSISTE EN UN LABERINTO, DEL CUAL NO SE SABE A PRIORI SU FORMA, DIVIDIDO EN VARIAS UBICACIONES. EN CADA UBICACIÓN PUEDE HABER MINAS (NINGUNA, UNA, DOS, O MUCHAS MÁS, TAMPOCO LO SABEMOS A PRIORI). AL FINAL DEL LABERINTO, EN LA ÚLTIMA UBICACIÓN, ESTÁ BRIAN, Y ES LA UBICACIÓN EN DONDE DEBE FINALIZAR EL ROBOT. NOTAR QUE PUEDE HABER MINAS EN DONDE SE ENCUENTRA BRIAN, QUE TAMBIÉN DEBEN SER DESACTIVADAS, Y TAMBIÉN EN LA UBICACIÓN EN DONDE COMIENZA EL ROBOT. PARA DARNOS UNA IDEA DEL ESCENARIO NOS BRINDAN EL SIGUIENTE DIBUJO DE EJEMPLO.

$\wedge = Y \rightarrow \text{NO } V = 0$



SE PIDE ENTONCES ESCRIBA UN PROGRAMA QUE LLEVE AL ROBOT HASTA LA UBICACIÓN DE BRIAN, DESACTIVANDO TODAS LAS MINAS QUE ENCUENTRE EN EL CAMINO.

**AL EMPEZAR A EJECUTAR**

**RESCATAR AL SOLDADO BRIAN DESACTIVANDO TODAS LAS MINAS**

**DEFINIR RESCATAR AL SOLDADO BRIAN DESACTIVANDO TODAS LAS MINAS**



$\wedge = Y$   $\neg = \text{NO}$   $V = O$

| REPETIR HASTA QUE ¿ESTA BRIAN?

| AVANZAR DESACTIVANDO TODAS LAS MINAS DE LA UBICACIÓN

| DESACTIVAR TODAS LAS MINAS DE LA UBICACIÓN

DEFINIR AVANZAR DESACTIVANDO TODAS LAS MINAS DE LA UBICACIÓN

| DESACTIVAR TODAS LAS MINAS DE LA UBICACIÓN

| DAR UN PASO EN EL CAMINO

DEFINIR DESACTIVAR TODAS LAS MINAS DE LA UBICACIÓN

| REPETIR CANTIDAD DE MINAS EN LA UBICACIÓN VECES

| DESACTIVAR MINA

HAY ALGO  
QUE NO ME  
CIERRA PERO  
NO SE QUE?

REPITO O  
HAGO DE+  
UN  
PROCEDIMIE  
NTO

EJERCICIO 4) REGRESO A LA ISLA DEL SOL TODO ESTÁ LISTO PARA LLEGAR A LA PARADISIACA ISLA DEL SOL NUEVAMENTE. PERO ESTA VEZ QUEREMOS PODER LLEVAR LA CENA LISTA, POR LO QUE QUEREMOS PESCAR A LOS PECES DEL CAMINO (LOS PECES, NO A LOS TIBURONES). ADEMÁS CONTAMOS ESTA VEZ CON UN NUEVO CONJUNTO DE PRIMITIVAS, EN ESTA ACTIVIDAD EL ESCENARIO PUEDE VARIAR, TANTO EN EL LUGAR DE DONDE ARRANCA EL BOTE, COMO EL LUGAR DONDE SE ENCUENTRA LA ISLA, COMO EL CAMINO A SEGUIR PARA LLEGAR, COMO LOS LUGARES DONDE ESTÁN LOS PECES, Y LA CANTIDAD DE PECES EN CADA LUGAR (PUEDE NO HABER PECES O PUEDEN HABER VARIOS EN UNA MISMA UBICACIÓN). A DIFERENCIA DE LA ÚLTIMA VEZ, TAMPOCO SABEMOS EXACTAMENTE A CUÁNTAS UBICACIONES DE DISTANCIA ESTÁ LA ISLA, AUNQUE SABEMOS QUE SIEMPRE HAY UN CAMINO A LA MISMA. PUEDEN VERSE DOS EJEMPLOS DE ESCENARIO A CONTINUACIÓN, EN EL PRIMERO, EL BOTE AÚN NO HA ARRANCADO EL RECORRIDO, MIENTRAS QUE EN EL SEGUNDO YA HA DADO 3 PASOS POR EL CAMINO SIN TIBURONES.



El bote es nuestro autómatas, y entiende las siguientes primitivas:

<b>Avanzar en el camino</b> Hace que el bote avance una ubicación por el camino sin tiburones. Falla si el bote ya está en la Isla del Sol.	<b>Pescar</b> Hace que el bote pesque un pez de la ubicación donde se encuentra. Falla si no hay un pez en la ubicación del bote o si todos los peces de la ubicación ya fueron pescados.
<b>cantidad de ubicaciones hasta la isla</b> Sensor numérico que describe la cantidad de pasos que hay desde donde comienza el bote hasta la Isla del Sol, sin incluir la ubicación de esta última.	<b>¿hay pez?</b> Indica si hay un pez en la ubicación en donde se encuentra el bote que aún no se haya pescado.
<b>Desembarcar</b> Desembarca en la isla. El bote debe estar en la misma ubicación que la isla.	

$\wedge = Y$   $\neg = \text{NO}$   $\vee = \text{O}$

SE BUSCA UN PROCEDIMIENTO “IR A LA ISLA DEL SOL” QUE LLEVE EL BOTE A LA ISLA DEL SOL, PESCANDO TODOS LOS PECES QUE ENCUENTRE EN EL CAMINO, Y DESEMBARQUE ALLÍ.

DEFINIR IR A LA ISLA DEL SOL

| AVANZAR PESCANDO

| DESEMBARCAR

DEFINIR AVANZAR PESCANDO

| REPETIR CANTIDAD DE UBICACIONES HASTA LA ISLA +1 VECES

| PESCAR TODOS LOS PECES QUE HAYA

| AVANZAR EN EL CAMINO

DEFINIR PESCAR TODOS LOS PECES QUE HAYA

| REPETIR HASTA QUE  $\neg$  ¿NO HAY PEZ?

| PESCAR

```
Al empezar a ejecutar
| Llegar a la Isla del Sol pescando

Definir Llegar a la Isla del Sol pescando
| Avanzar por el camino pescando peces
| Desembarcar

Definir Avanzar por el camino pescando peces
| Repetir cantidad de ubicaciones hasta la Isla + 1 veces
|   | Pescar todos los peces de ubicación
|   | Avanzar en el camino

Definir Pescar todos los peces de ubicación
| Repetir hasta que  $\neg$  ¿hay pez?
|   | Pescar
```

$\wedge = Y$   $\neg = \text{NO}$   $\vee = \text{O}$

**Ejercicio 4:**

Al Empezar a Ejecutar

IR A LA ISLA DEL SOL

Definir IA A LA ISLA DEL SOL

Repetir cantidad de ubicaciones hasta la isla verde

Pescar si hay pez

Avanzar en el camino

Avanzar en el camino

Desembarcar

Definir Pescar si hay pez

Repetir hasta que  $\neg$  hay pez?

Pescar

Se puede usar el +1 sacando el caso de border



$\wedge=Y \neg=NO V=O$

**Ejercicio 5)** Me llama usted, entonces voy... Vuelve el invierno y Don Barredora se pone otra vez en acción. Pero la ciudad cambia constantemente, y ahora las casas están por todos lados. En este problema siguen habiendo filas de calles, pero no sabemos ni cuántas ni el largo de las mismas. Tampoco tenemos información sobre dónde están las casas en dichas calles, que pueden estar en cualquier lugar. No solo eso, sino que algunas de ellas ya fueron limpiadas el día anterior. y no necesariamente tienen nieve. También hay nieve en lugares donde no hay casas, y Don Barredora no trabajará de más, dejando esa nieve donde está.



DUDA: ¿DÓNDE ESTA DON BARREDORA? O SEA SIEMPRE EMPIEZA AHI

Para manipular a Don Barredora están las siguientes primitivas:

**Mover barredora arriba**

Mueve la barredora una ubicación hacia arriba. Falla sí no hay más ubicaciones hacia arriba.

**Mover barredora abajo**

Mueve la barredora una ubicación hacia abajo. Falla sí no hay más ubicaciones hacia abajo.

**Mover barredora izquierda**

Mueve la barredora una ubicación hacia la izquierda. Falla sí no hay más ubicaciones hacia la izquierda.

**Mover barredora derecha** Mueve la barredora una ubicación hacia la derecha. Falla sí no hay más ubicaciones hacia la derecha.

**Quitar nieve** Quita la nieve de la ubicación actual. Falla sí no hay nieve en la ubicación actual.

**¿hay casa?** Indica sí hay una casa en la ubicación de Don Barredora.

**¿hay nieve?** Indica sí hay nieve en la ubicación de Don Barredora

**¿se puede mover a la derecha?** Indica sí Don Barredora tiene lugar para moverse a la derecha.

**¿se puede mover a la izquierda?** Indica sí Don Barredora tiene lugar para moverse a la izquierda.

**cantidad de calles de la ciudad** Sensor numérico que indica cuántas calles (filas) hay en la ciudad

**AL EMPEZAR A EJECUTAR**

| QUITAR NIEVE DE LA CIUDAD

**DEFINIR QUITAR NIEVE DE LA CIUDAD**

| REPETIR CANTIDAD DE CALLES DE LA CIUDAD -1 VECES

| QUITAR NIEVE DE LA FILA ACTUAL

| IR AL INICIO DE LA SIGUIENTE FILA

| QUITAR NIEVE DE LA FILA ACTUAL

$\wedge=Y \neg=NO \vee=O$

## DEFINIR QUITAR NIEVE DE LA FILA ACTUAL

```
| REPETIR HASTA QUE ~¿SE PUEDE MOVER A LA DERECHA?
|   | QUITAR SI HAY CASA CON NIEVE
|   | MOVER BARREDORA DERECHA
```

## DEFINIR AVANZAR QUITANDO NIEVE SI HAY CASA

```
| SI ¿HAY CASA? ^ ¿HAY NIEVE? ENTONCES
|   | QUITAR NIEVE
```

## DEFINIR IR AL INICIO DE LA SIGUIENTE FILA

```
| REPETIR HASTA QUE ~ ¿SE PUEDE MOVER A LA IZQUIERDA?
|   | MOVER BARREDORA IZQUIERDA
|   | MOVER BARREDORA ABAJO
```

5)  
**Al empezar a ejecutar**  
| Quitar nieve de la ciudad

### Definir Quitar nieve de la ciudad

```
| Repetir (cantidad de calles de la ciudad - 1) veces
|   | Limpiar nieve de la calle
|   | Posicionar la barredora al inicio de la siguiente calle
|   | Limpiar nieve de la calle
```

### Definir Limpiar nieve de la calle

```
| Repetir hasta que ~¿se puede mover a la derecha?
|   | Quitar nieve de una casa si hay edificación
|   | Mover barredora derecha
|   | Quitar nieve de una casa si hay edificación
```

### Definir Quitar nieve de una casa si hay edificación

```
| Si ¿hay casa? ^ ¿hay nieve? Entonces
|   | Quitar nieve
```

### Definir Posicionar la barredora al inicio de la siguiente calle

```
| Repetir hasta que ~¿se puede mover a la izquierda?
|   | Mover barredora izquierda
|   | Mover barredora abajo
```



$\wedge = Y$   $\neg = \text{NO}$   $\vee = \text{O}$

## Al empezar a ejecutar

| Quitar nieve de la ciudad

## Definir Quitar nieve de la ciudad

| Repetir (cantidad de calles de la ciudad - 1) veces  
 | | Limpiar nieve de la calle  
 | | Posicionar la barredora al inicio de la siguiente calle  
 | Limpiar nieve de la calle

## Definir Limpiar nieve de la calle

| Repetir hasta que  $\sim$ ¿se puede mover a la derecha?  
 | | Quitar nieve de una casa si hay edificación  
 | | Mover barredora derecha  
 | Quitar nieve de una casa si hay edificación



Definir ¿Hay casa con nieve? = ¿hay casa?  $\wedge$  ¿hay nieve?

## Definir Quitar nieve de una casa si hay edificación

| Si ¿Hay casa con nieve? Entonces  
 | | Quitar nieve

## Definir Posicionar la barredora al inicio de la siguiente calle

| Repetir hasta que  $\sim$ ¿se puede mover a la izquierda?  
 | | Mover barredora izquierda  
 | Mover barredora abajo

## EJERCICIO 6) EL BOMBERO VALIENTE

Hay un edificio en llamas, y un bebé en el último piso. Pero no hay que tener miedo, pues Bob el bombero valiente está listo para la acción. En esta actividad Bob será nuestro autómatas, el escenario representará las escaleras del edificio. **Bob comienza en la punta más abajo y a la izquierda del edificio, y el bebe se encuentra en la punta más arriba.** Nuestro trabajo será ayudar a Bob a **rescatar al bebé**. Eso sí, hay fuego en el camino, y deberemos asegurarnos de **ir apagando todo el fuego** que podamos. Para ello, vamos a contar con baldes de agua o extintores en diversas ubicaciones. El dibujo a la derecha representa el edificio y sus escaleras, la forma del escenario puede variar en el ancho de los pisos, así como en la ubicación de los fuegos, extintores y baldes. Por otro lado, la altura entre un piso y otro siempre es de dos ubicaciones, y siempre hay ocho pisos en el edificio. Por ello, se espera realice una solución general, para cualquier distribución de incendios, y no particular para la que se muestra en el dibujo de ejemplo.

Su trabajo será entonces escribir el procedimiento **Salvar al bebé apagando incendios** que haga que Bob llegue hasta el bebe, y lo salve, a la vez que apague todos los incendios posibles en el camino. El dibujo a la derecha representa el edificio y sus escaleras, la forma del escenario no varía, pero la ubicación de los fuegos, extintores y baldes sí. Por ello, se espera realice una solución general, para cualquier distribución de incendios, y no particular para la que se muestra en el dibujo de ejemplo. Vamos a contar con las siguientes primitivas. Algunas aclaraciones adicionales:

- Puede haber fuego a apagar donde arranca el bombero, pero no dónde está el bebé.
- No puede haber un matafuegos y un balde de agua en la misma ubicación, nunca.

### Mover arriba

Hace que Bob se mueva hacia arriba una ubicación. Debe haber una ubicación hacia arriba.

### Mover abajo

Hace que Bob se mueva hacia abajo una ubicación. Debe haber una ubicación hacia abajo.

### Mover a la izquierda

Hace que Bob se mueva hacia la izquierda una ubicación. Debe haber una ubicación hacia la izquierda.

**Mover a la derecha** Hace que Bob se mueva hacia la derecha una ubicación. Debe haber una ubicación hacia la derecha.

### ¿hay fuego?

Indica sí hay un fuego en la ubicación donde está Bob.

### ¿hay extintor?

Indica sí hay un extintor en la ubicación donde está Bob.

### ¿hay balde de agua?

Indica sí hay un balde de agua en la ubicación donde está Bob.

### Apagar incendio

Apaga el incendio de la ubicación donde está Bob. Falla sí no hay fuego en la ubicación donde está Bob, o sí hay fuego pero no hay un balde de agua o extintor.

### Rescatar bebé

Hace que Bob rescate al bebé. Falla sí Bob no está sobre el bebé.

**ancho del piso** Sensor numérico que describe el ancho del piso actual en el que se encuentra Bob.

## AL EMPEZAR A EJECUTAR

### | SALVAR AL BEBÉ APAGANDO INCENDIOS

$\wedge = Y$   $\neg = \text{NO}$   $\vee = \text{O}$

AL EMPEZAR A EJECUTAR

| SALVAR AL BEBE APAGANDO INCENDIOS

DEFINIR ¿HAY ELEMENTOS PARA APAGAR EL FUEGO? = ¿HAY BALDE DE AGUA?  
¿HAY EXTINTOR?

DEFINIR ¿HAY INCENDIO Y PUEDO APAGARLO? = ¿HAY FUEGO?  $\wedge$  ¿HAY ELEMEN  
PARA APAGAR EL FUEGO?

DEFINIR SALVAR AL BEBE APAGANDO INCENDIOS

| AVANZAR APAGANDO INCENDIOS

| RESCATAR BEBE

DEFINIR AVANZAR APAGANDO INCENDIOS

REPETIR 4 VECES

AVANZAR APAGANDO INCENDIOS HACIA LA DERECHA

SUBIR APAGANDO INCENDIO

AVANZAR APAGANDO INCENDIOS HACIA LA IZQUIERDA

SUBIR APAGANDO INCENDIO

DEFINIR AVANZAR APAGANDO INCENDIOS HACIA LA DERECHA

REPETIR (ANCHO DE PISO -1) VECES

APAGAR INCENDIO SI HAY

MOVER A LA DERECHA

APAGAR INCENDIO SI HAY

DEFINIR AVANZAR APAGANDO INCENDIOS HACIA LA IZQUIERDA

REPETIR ANCHO DE PISO -1 VECES

APAGAR INCENDIO SI HAY

MOVER A LA IZQUIERDA

APAGAR INCENDIO SI HAY

DEFINIR APAGAR INCENDIO SUBIENDO

MOVER ARRIBA

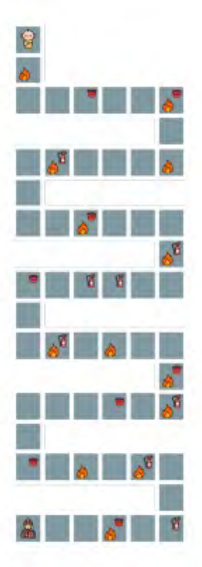
APAGAR INCENDIO SI HAY

MOVER ARRIBA

DEFINIR APAGAR INCENDIO SI HAY

SI ¿HAY INCENDIO Y PUEDO APAGARLO? ENTONCES

APAGAR INCENDIO





## MODELO DE PARCIAL I

**Ejercicio 1)** Conteste las siguientes preguntas

- ¿Por qué motivo los nombres de los comandos siempre comienzan con un verbo?
- ¿Qué diferencia hay entre la repetición condicional y la alternativa condicional?
- Indique si el siguiente código es adecuado o no en términos de lo trabajado en la materia (ignore el nombre del procedimiento). Justifique su respuesta en cualquier caso. Si considera que el código no es adecuado, de una definición equivalente pero adecuada.

```

definir Hacer algo
| Si - ¿hay queso acá?
| | Dar un paso en el laberinto
| Sino
| | Comer el queso
| | Dar un paso en el laberinto
    
```

**Ejercicio 1)** Resuelva el siguiente ejercicio usando todas las buenas prácticas de programación trabajadas.

Froggy, la rana, vuelve a la acción. Esta vez quiere comer todos los insectos de un hermoso estanque.

La situación es la siguiente, el escenario tiene una forma rectangular, con excepción de la fila superior, en donde hay dos ubicaciones adicionales, una en el borde izquierdo (donde comienza Froggy) y otra en el derecho (en donde debe terminar). Fuera de esas ubicaciones, podemos decir que hay una serie de columnas en el escenario en donde algunas no tienen nenúfares (sobre las cuales Froggy no puede moverse) y otras con nenúfares (por donde sí puede pasar). Dichas columnas se encuentran de forma intercalada, una sin y una con. La primera columna sin nenúfares tiene una ubicación con nenúfar en la fila inferior; la segunda tiene una ubicación con nenúfar en la fila superior, la tercera nuevamente en la inferior, y así siguiendo. La cantidad de columnas sin nenúfares siempre es impar (1, 3, 5, 7, 9, etc.); aunque no sabemos a priori la cantidad, como tampoco sabemos la cantidad de filas.

En distintas ubicaciones donde hay nenúfares pueden encontrarse insectos, los cuales Froggy quiere comer. La cantidad de insectos en cada ubicación puede variar (pueden no haber, haber uno, dos, o incluso más). Sin embargo, Froggy no puede comer insectos de cualquier ubicación, ya que en algunas de esas ubicaciones pueden haber depredadores, como los pelícanos o los erizos. Froggy no puede comer los insectos de dichas ubicaciones. En la ubicación donde comienza Froggy y donde termina también puede haber insectos (con o sin pelícanos o erizos). Es posible que haya tanto erizos como pelícanos en la misma ubicación.

A la derecha se muestra un posible escenario inicial, aunque otros escenarios iniciales son posibles, siempre que se ajusten a la descripción arriba mencionada. En este ejemplo, el escenario tiene 11 columnas por 9 filas, con los dos lugares adicionales en la fila superior ya mencionados. Hay 5 columnas sin nenúfares, de las cuales 3 tienen nenúfar en la fila inferior, y 2 en la superior.

Lo que se pide es que realice un programa que haga que Froggy coma absolutamente todos los insectos del escenario que le sean posibles. Para ello puede disponer de las siguientes primitivas y sensores.

**Pista:** Piense en una estrategia en donde repita una misma secuencia tantas veces como espacios sin nenúfares hay en la fila superior. En el ejemplo del dibujo hay 3 ubicaciones sin nenúfares en dicha fila.

<b>Saltar arriba</b> Hace que Froggy se mueva una ubicación para arriba. Falla si no hay más lugar hacia arriba o si no hay nenúfar en dicha ubicación.	<b>Saltar abajo</b> Hace que Froggy se mueva una ubicación para abajo. Falla si no hay más lugar hacia abajo o si no hay nenúfar en dicha ubicación.
<b>Saltar a la derecha</b> Hace que Froggy se mueva una ubicación para la derecha. Falla si no hay más lugar hacia la derecha o si no hay nenúfar en dicha ubicación.	<b>Saltar a la izquierda</b> Hace que Froggy se mueva una ubicación para la izquierda. Falla si no hay más lugar hacia la izquierda o si no hay nenúfar en dicha ubicación.
<b>¿se puede saltar arriba?</b> Indica si Froggy se puede mover un lugar hacia arriba, es decir, si hay una ubicación allí y además este tiene nenúfar.	<b>¿se puede saltar abajo?</b> Indica si Froggy se puede mover un lugar hacia abajo, es decir, si hay una ubicación allí y además este tiene nenúfar.
<b>cantidad de lugares sin nenúfar en la fila superior</b> Sensor numérico que describe la cantidad de lugares sin nenúfar que hay en la fila superior.	<b>¿hay insectos acá?</b> Indica si hay insectos (uno o más) en la ubicación donde está Froggy.
<b>¿hay pelícano acá?</b> Indica si hay un pelícano en la ubicación donde está Froggy.	<b>¿hay erizo acá?</b> Indica si hay un erizo en la ubicación donde está Froggy.
<b>comer insecto</b> Hace que Froggy coma un sólo insecto de la ubicación donde se encuentra. Falso si Froggy no está sobre un insecto o si todos los insectos de la ubicación ya fueron comidos.	<b>cantidad de insectos</b> Sensor numérico que describe la cantidad de insectos en la ubicación donde está Froggy. Falso si no hay insectos en la ubicación.

c) no es adecuado porque falla la sintaxis en papel. Le falta el entonces al condicional si.

No es adecuado unir movimiento de precesamiento

Y no es necesario forzar el “no hay queso”

a) Los comandos empiezan con verbos xq son descripción de acciones

b) la diferencia es que la repetición condicional repite una cantidad de veces indeterminada hasta que se cumpla una condición y la alternativa condicional solo permite elegir entre dos ramas según la condición

$\wedge = Y \quad \neg = \text{NO} \quad \vee = \text{O}$

## AL EMPEZAR A EJECUTAR

| COMER TODOS LOS INSECTOS DEL ESCENARIO

## DEFINICIÓN DE CONECTIVAS

¿HAY DEPREDADORES ACA? = ¿HAY ERIZO?  $\vee$  ¿HAY PELICANO?

¿ES POSIBLE COMER INSECTOS ACA? =  $\neg$  ¿HAY DEPREDADORES?  $\wedge$  ¿HAY INSECTOS ACA?

## DEFINIR COMER TODOS LOS INSECTOS DEL ESCENARIO

| COMER INSECTOS SI ES POSIBLE

| SALTAR LA DERECHA

| REPETIR CANTIDAD DE LUGARES SIN NENUFAR EN LA FILA SUPERIOR VECES

| BAJAR COMIENDO INSECTOS DE LA COLUMNA SI ES POSIBLE

| AVANZAR COMIENDO INSECTOS DE LA FILA

| SUBIR COMIENDO INSECTOS DE LA COLUMNA SI ES POSIBLE

| AVANZAR COMIENDO INSECTOS DE LA FILA

| COMER INSECTO SI ES POSIBLE

EN ESTE  
EJEMPLO  
3 VECES  
XQ HAY 3  
LUGARES

## DEFINIR COMER INSECTO SI ES POSIBLE

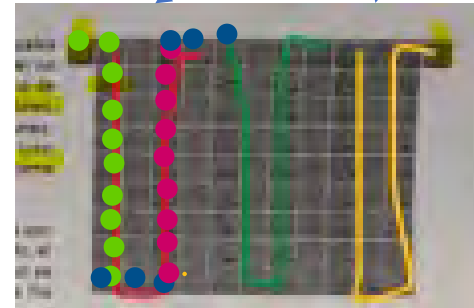
| SI ¿ES POSIBLE COMER INSECTOS ACA? ENTONCES

| COMER LA CANTIDAD DE INSECTOS POSIBLE

## DEFINIR COMER LA CANTIDAD DE INSECTOS POSIBLE

| REPETIR CANTIDAD DE INSECTOS

| COMER INSECTOS



## DEFINIR BAJAR COMIENDO INSECTOS DE LA COLUMNA SI ES POSIBLE ●

| REPETIR HASTA QUE  $\neg$  ¿SE PUEDE SALTAR ABAJO?

| COMER INSECTO SI ES POSIBLE

| SALTAR ABAJO

## DEFINIR AVANZAR COMIENDO INSECTOS DE LA FILA ●

| REPETIR 2 VECES

| COMER INSECTO SI ES POSIBLE

| SALTAR A LA DERECHA

## DEFINIR SUBIR COMIENDO INSECTOS DE LA COLUMNA SI ES POSIBLE ●

REPETIR HASTA QUE  $\neg$  ¿SE PUEDE SALTAR ARRIBA?

| COMER INSECTO SI ES POSIBLE

| SALTAR ARRIBA

## Mapa de Problema

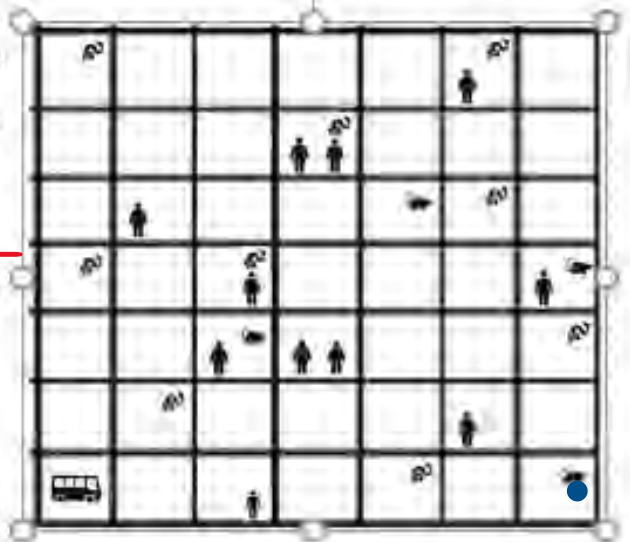


Juan es un conductor de combis privadas. Su trabajo consiste en llevar a los pasajeros y las pasajeras a destino esquivando los peligros que puedan presentarse en las paradas. Los cuales pueden ser serpientes venenosas y ratas rabiosas.

El escenario es sencillo. Juan puede moverse por cualquier lugar. El escenario es variable, no sabemos su tamaño, tampoco en que lugares están los pasajeros y las pasajeras. Juan siempre comienza su recorrido en la esquina izquierda inferior, donde también puede haber pasajeros y pasajeras que quieran subirse al recorrido.

Pero cuidado, en los posibles puntos donde se puedan ubicar los pasajeros y las pasajeras puede haber serpientes venenosas o ratas rabiosas que impidan que los pasajeros puedan subir al micro representando también un peligro para los que ya se encuentran en viaje. Obligando a Juan a no poder detenerse en la parada y continuar con su recorrido.

A la derecha se muestra un posible escenario inicial, aunque otros escenarios iniciales son posibles, siempre que se ajusten a la descripción arriba mencionada.



Lo que se pide es que se realice un programa que haga que Juan pueda recoger a los pasajeros y pasajeras que sea posible durante todo el recorrido.



Para ello se dispone de las siguientes primitivas y sensores.

<b>Mover a la derecha</b> Hace que Juan se mueva una ubicación para la derecha. Falla si no hay más ubicaciones hacia la derecha.	<b>Mover para arriba</b> Hace que Juan se mueva una ubicación para arriba. Falla si no hay más ubicaciones para arriba.
<b>Ir a la columna inicial</b> Hace que Juan vuelva a la primera columna del escenario.	<b>Cantidad de columnas del escenario.</b> Sensor numérico que describe la cantidad de columnas totales del escenario.
<b>Recoger Pasajera</b> Hace que Juan suba a una pasajera. Falla si no hay pasajeras para que suban.	<b>Recoger Pasajero</b> Hace que Juan suba a un pasajero. Falla si no hay pasajeros para que suban.
<b>¿puede avanzar hacia arriba?</b> Indica si Juan puede moverse una ubicación hacia la derecha.	<b>¿Hay pasajera acá?</b> Indica si hay una pasajera en la ubicación en donde se encuentra Juan
<b>¿Hay pasajero acá?</b> Indica si hay una pasajera en la ubicación en donde se encuentra Juan	<b>¿hay rata acá?</b> Indica si hay una rata rabiosa en la ubicación en donde se encuentra Juan
<b>¿hay serpiente acá?</b> Indica si hay una serpiente venenosa en la ubicación donde se encuentra Juan	<b>Cantidad de pasajeras</b> Sensor numérico que indica la cantidad de pasajeras que hay en la ubicación
<b>Cantidad de pasajeros</b> Sensor numérico que indica la cantidad de pasajeros que hay en la ubicación	



**AL EMPEZAR A EJECUTAR**

| RECOGER A TODXS LXS PASAJERXS POSIBLES

**DEFINICION DE CONECTIVAS**

¿HAY ALGÚN PASAJERX ACA? = ¿HAY PASAJERO ACA?  $\vee$  ¿HAY PASAJERA ACA?

¿HAY PELIGRO ACA? = ¿HAY SERPIENTE ACA?  $\vee$  ¿HAY RATA ACA?

¿SE PUEDE RECOGER PASAJERXS ACA? = ¿HAY ALGUN PASAJERX ACA?  $\wedge \neg$  ¿HAY PELIGRO ACA?

**DEFINIR RECOGER A TODXS LXS PASAJERX POSIBLES**

| REPETIR HASTA QUE  $\neg$  ¿PUEDE AVANZAR HACIA ARRIBA?

| RECOGER LXS PASAJERXS POSIBLES DE LA FILA

| IR AL INICIO DE LA SIGUIENTE FILA

| RECOGER PASAJERXS POSIBLES DE LA FILA

**DEFINIR RECOGER LXS PASAJERXS POSIBLES DE LA FILA**

| REPETIR (CANTIDAD DE COLUMNAS DEL ESCENARIO -1) VECES

| VERIFICAR SI HAY PASAJERXS QUE SE PUEDAN RECOGER

| MOVER A LA DERECHA

| VERIFICAR SI HAY PASAJERX QUE SE PUEDAN RECOGER

**DEFINIR IR AL INICIO DE LA SIGUIENTE FILA**

| MOVER ARRIBA

| IR A LA COLUMNA INICIAL

**DEFINIR VERIFICAR SI HAY PASAJERXS QUE SE PUEDAN RECOGER**

| SI ¿SE PUEDE RECOGER PASAJERXS ACA? ENTONCES

| RECOGER LA CANTIDAD DE PASAJERXS POSIBLES

**DEFINIR RECOGER LA CANTIDAD DE PASAJERXS POSIBLES**

| REPETIR CANTIDAD DE PASAJERAS VECES

| RECOGER PASAJERA

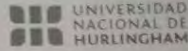
| REPETIR CANTIDAD DE PASAJEROS VECES

| RECOGER PASAJERO

$\wedge = Y \rightarrow \text{NO } V = O$

## MODELO DE PARCIAL 3

ILyPC - Parcial 2 (2023 verano)



### Segundo Parcial

**Ejercicio 1)** Resuelva el siguiente ejercicio: El cerdito Babe el cerdito comelón, vive en una granja en donde sus dueños disponen para él de manzanas, naranjas, bananas y paltas, todo un festín para un cerdito. Babe come todo lo que puede. Pero claro, sus dueños dejan la comida en cualquier lugar del corral pero cuidado que puede haber lugares que se fumigaron y hacen que no deba comer la/s frutas que marca la/s flecha/s de ese lugar. Babe siempre espera su almuerzo en la ubicación más a la izquierda y arriba del escenario. Esto se muestra en el siguiente ejemplo. Cualquier otro escenario que cumpla lo mencionado es viable. Aclaración, puede haber más de una flecha pero de distinto color en el mismo lugar.

Babe está entrenado y sabe que si hay una flecha roja, hay una manzana en la ubicación vecina a la derecha, si el color es naranja, hay una naranja a la izquierda, si es verde, una palta abajo y si es amarilla, una banana arriba.

Se pide que ayude a Babe a consumir las frutas del corral, para ello se cuenta con las siguientes primitivas.

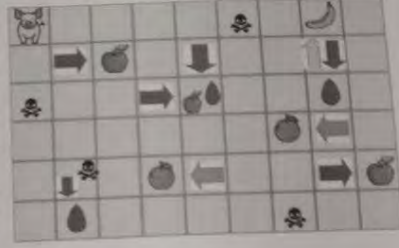
<p><b>Mover arriba</b> Hace que Babe se mueva una ubicación para arriba. Falla si no hay más ubicaciones hacia allí.</p>	<p><b>Mover a la derecha</b> Hace que Babe se mueva una ubicación para la derecha. Falla si no hay más ubicaciones hacia allí.</p>
<p><b>Mover abajo</b> Hace que Babe se mueva una ubicación para abajo. Falla si no hay más ubicaciones hacia allí.</p>	<p><b>Mover a la izquierda</b> Hace que Babe se mueva una ubicación para la izquierda. Falla si no hay más ubicaciones hacia allí.</p>
<p><b>¿hay flecha roja?</b> Indica si hay flecha de color rojo en donde está Babe. Falla si no hay flecha</p>	<p><b>¿hay flecha amarilla?</b> Indica si hay flecha de color amarillo en donde está Babe. Falla si no hay flecha</p>
<p><b>¿hay flecha naranja?</b> Indica si hay flecha de color naranja en donde está Babe. Falla si no hay flecha</p>	<p><b>¿hay flecha verde?</b> Indica si hay flecha de color verde en donde está Babe. Falla si no hay flecha</p>
<p><b>¿hay flecha?</b> Indica si hay una flecha en la ubicación actual.</p>	<p><b>¿está fumigado?</b> Indica si en la ubicación se fumigó.</p>
<p><b>Cantidad de filas</b> Sensor numérico que describe la cantidad de filas del corral.</p>	<p><b>¿estoy en una ubicación de la última columna?</b> Indica si Babe se encuentra en una ubicación de la última columna. Aclaración: La última columna es la que está más a la derecha.</p>
<p><b>Comer banana</b> Hace que Babe coma una banana donde está ubicado. Falla si no hay banana.</p>	<p><b>Comer naranja</b> Hace que Babe coma una naranja donde está ubicado. Falla si no hay naranja.</p>
<p><b>Comer manzana</b> Hace que Babe coma una manzana donde está ubicado. Falla si no hay manzana.</p>	<p><b>Comer palta</b> Hace que Babe coma una palta donde está ubicado. Falla si no hay palta.</p>
<p><b>Ir al inicio de la fila</b> Hace que Babe vaya al inicio de la fila. Aclaración: El inicio de la fila es la ubicación más a la izquierda de la fila.</p>	

**Ejercicio 2)** Conteste las siguientes preguntas en función del punto anterior.

a. ¿Para que se utilizan los sensores? De ejemplo del parcial resuelto

b. ¿Qué diferencia hay entre la repetición simple y la repetición condicional? Ejemplifique

c. ¿Qué es una conectiva? De un ejemplo



Los sensores son la herramienta mediante la cual obtenemos valores de verdad

La repetición simple repite una cantidad de veces finita y fija

Mientras que la repetición condicional repite infinitamente hasta que cumple una condición

Las conectivas son combinaciones de sensores que dan un valor de verdad mas complejo