UBA – Colegio Nacional Buenos Aires – Pensamiento Computacional
Primer parcial– Primer cuatrimestre de 2023

1a	1b	1a	2a	2b	2c

Tema: Sadosky mochilero
Apellido ..... **Nombre** .....

Ejercicio 1. [15 puntos]
Juan, Laura y Tobías son un grupo de estudiantes que tiene que entregar un ejercicio de programación en Gobstones. Este ejercicio consiste hacer que el personaje de un juego recolecte objetos en su mochila a medida que avanza y los encuentra en un mapa. Si su mochila se llena, el personaje debe dejar de explorar el mapa. El ejercicio consiste en hacer un procedimiento que modele la recolección de objetos [1] que hace el personaje, el cual recibe como parámetro la capacidad de la mochila.

Juan ya tenía experiencia programando así que hizo una solución muy rápidamente, pero dijo que tenía recuperatorio de Álgebra y que no iba a poder revisarlo. Laura y Tobías, quienes están dando sus primeros pasos programando, corrieron un par de pruebas y notaron que no solo no funciona, sino que tampoco entienden bien qué trató de hacer.

```
procedure ejercicio(x){c0 := x
while(sigueOK() && c0>0){
    if(nroBolitas(Negro)>0){
    ojs := nroBolitas(Negro)
    c0 := c0 - ojs}
    repeat(ojs){Sacar(Negro)}
    sigue()
}}
```

Nota: Suponer que la función `sigueOK` devuelve verdadero si puede seguir avanzando el personaje y que el procedimiento `sigue` avanza al personaje a la posición siguiente.

a Indicar qué errores y malas prácticas de programación sigue esta propuesta de solución (marcarlas en el código).

b Proponer modificaciones en el código para que este pueda funcionar y para que la solución sea lo más comprensible posible.

c Justifique por qué su propuesta es superadora y qué ventajas tiene programar así.

Solución
(a) El código sigue las siguientes malas prácticas y errores:

- Errores de sintaxis en lenguaje Gobstones: los procedimientos deberían arramcar con mayúscula (por ej. sigue() y ejercicio())
- Nombres ambiguos para procedimientos, variables y parámetros. Por ejemplo, ejercicio, ojs, c0, x, sigue, sigueOK, entre otros.
- Inapropiada indentación del código: las estructuras de control (condicional, repetición condicional, repetición simple) y los procedimiento deberían tener su cuerpo en una indentación adentro para facilitar su lectura. Por ejemplo:

```
while(condicion){
|    cuerpo del ciclo
}

if(condicion){
|    cuerpo del condicional en caso positivo
}
procedure NombreDelProc(parametros){
|   cuerpo del procedimiento
}
```

- Mala abstracción del problema: está mezclado el código que habla sobre la representación del problema en el lenguaje Gobstones (usa funciones vinculadas al tablero o las bolitas) y loel código que tiene la lógica general del problema (mochila, objetos, recolección, capacidad, etc).

[1] Representados con bolitas de color negro.

(b) En base al punto (a), modificaría las indentaciones, los nombres de las variables y de los procedimientos, agregando algunos para modularizar más el código. Una propuesta es la siguiente:

```
1  procedure RecolectarObjetosDelMapa(capacidadDeMochila){
2      capacidadActualDeMochila := capacidadDeMochila
3      while(puedeAvanzarEnElMapa() && capacidadActualDeMochila>0){
4          cantObjetosEnCelda := nroObjetos()
5          if (cantObjetosEnCelda>0){
6              capacidadActualDeMochila := capacidadActualDeMochila - cantObjetosEnCelda
7              SacarObjetos(cantObjetosEnCelda)
8          }
9      }
10 }
11
12 function nroObjetos(){
13     return(nroBolitas(Negro))
14 }
15
16 procedure SacarObjetos(cantObjetosASacar){
17     repeat(cantObjetosASacar){
18         Sacar(Negro)
19     }
20 }
21
```

(c) Esta propuesta es superadora por las siguientes razones:

- Es más legible y comprensible para cualquiera que conozca el problema
- No es necesario conocer Gobstones para entender la idea general de la solución dado que están separados/encapsulados los procedimiento de la representación del problema
- Facilita encontrar errores de sintaxis y/o lógica en el código
- Posibilita que con mayor facilidad otras personas puedan usarlo y/o modificarlo

---

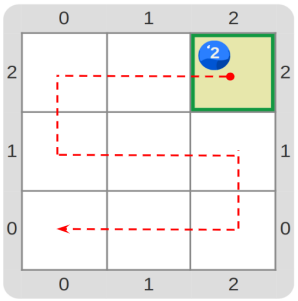
Ejercicio 2. [20 puntos]
Se tiene una aspiradora 2.0 que puede aspirar y lustrar pisos de madera sin ningún problema. Para ello tiene un sistema de recorrido zig-zag que le permite ir de lado a lado, dividiendo la habitación en celdas de 1 metro cuadrada (ver figura). Por cada celda, primero aspira las basuras que puedan haber y luego lustra la madera de esa sección.

Además, este aparato cuenta con una modalidad que le permite aspirar en *modo cuidado*. De esta forma, si la madera está dañada solamente aspira pero no lustra la celda.

a Escribir la especificación de un procedimiento que se encargue de hacer que la aspiradora limpie una habitación.

b Escribir el procedimiento del ítem anterior.

c Escribir un procedimiento para que la aspiradora limpie en *modo cuidado*.



Nota: Puede suponer dados los siguientes procedimientos y funciones:

- `procedure MoverAspiradoraHacia(sentido)`
- `procedure AspirarCelda()`
- `procedure LustrarCelda()`
- `function puedeMoverAspiradoraHacia(sentido)`

Solución
(a) Especificación:

- Pre:
  - Aspiradora está en extremo NE (2 Azules), junto al cabezal.

- En la habitación puede haber basuras (Rojo).
- Opcional: Se podría agregar un comentario sobre el tamaño de la habitación. Esta solución funciona incluso para un tablero 1x1, por lo que no es necesario. Pero depende de su solución.

- Post:
  - Aspiradora está en extremo SO (2 Azules), junto al cabezal.
  - Las celdas están limpias: sin basura y lustradas (no hay bolitas rojas).
  - Cualquier otra cosa sigue sin modificación.

(b)

```
procedure LimpiarHabitacion(){
    LimpiarFila()
    while(puedeMoverAspiradoraHacia(Sur)){
        MoverAspiradoraHacia(Sur)
        LimpiarFila()
    }
}

procedure LimpiarFila(){
    if(puedoMoverAspiradoraHacia(Este)){
        LimpiarFilaEnSentido(Este)
    }else{
        LimpiarFilaEnSentido(Oeste)
    }
}

procedure LimpiarFilaEnSentido(sentido){
    while(puedoMoverAspiradoraHacia(sentido)){
        LimpiarCelda()
        MoverAspiradoraHacia(sentido)
    }
    # para limpiar ultima celda de cada fila
    LimpiarCelda()
}

procedure LimpiarCelda(){
    AspirarCelda()
    LustrarCelda()
}
```

(c) El *modo cuidado* es idéntico al anterior, solo hay que modificar cómo limpiar cada celda.

```
procedure LimpiarCeldaEnModoCuidado(){
    AspirarCelda()
    if(estaMaderaDaniada()){
        LustrarCelda()
    }
}

function estaMaderaDaniada(){
    return(nroBolitas(Negro)>0)
}
```
