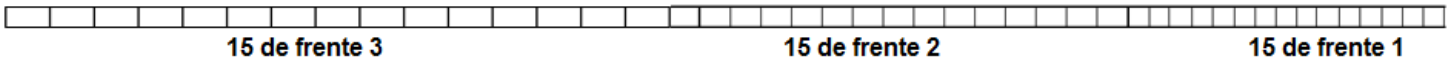
Escuela de Ingeniería		
Segundo Parcial de: Programación I		**Código de materia:**
Fecha: Diciembre 2019	**Grupo: Matutino**	**Hoja 1 de 3**

Duración:	**3 horas**	**Con material:**	**No**	**Puntaje máximo:**	**35 puntos**

Ejercicio 1: Arrays (máximo 14 puntos)

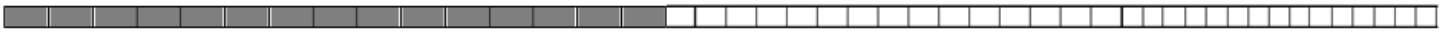
1a) (máximo 8 puntos): Para una feria artesanal que se realizará en la calle, se dispondrán en forma consecutiva sobre una misma acera de 90 mt: 15 stands de 3 m^2 (frente: 3 mts, ancho: 1), 15 stands de 2 m^2 (2 metros de frente y 1 de ancho) y 15 de 1 m^2 (1 mt de frente y 1 de ancho), según se ve en la siguiente imagen:



Se tiene un array indexado booleano que indica por cada stand individual: true si fue alquilado o false si no lo fue (las 15 primeras posiciones son de los stands de frente 3, los siguientes 15 los de frente 2 y las ultimas 15 los de frente 1). A partir de la información de los stands ocupados (en true), implementar en JS la función standsJuntos que retorne cuál sería el tamaño máximo de un stand a alquilar considerando que se pueden unir stands vecinos libres.

Encabezado: function standsJuntos(ocupados){

Ejemplo 1: si estuvieran alquilados los 15 primeros, disponibles todos los de 2 y todos los de 1, el tamaño máximo sería 45 m^2.



Ejemplo 2: si estuvieran alquilados todos los señalados en gris:

el tamaño máximo sería de 9 m^2.

1b) (máximo 6 puntos): En una maratón, los corredores deben ir pasando por cada uno de los 3 puestos de control. A una cierta hora, se tiene esa información en un único array indexado. Los números de atleta van del 1 al 100. No hay inconsistencias.
Ejemplo:
10 15 10 8 20 15 4 15 10 20 representa que el atleta 10 y el atleta 15 ya pasaron por los 3 puestos, el 20 pasó por 2 puestos, el 8 y el 4 pasaron por un solo puesto de control. Los demás atletas no pasaron aún por el primer puesto de control.

Implementar en JS la función cantidadTotal que recibe el array y retorna la cantidad de atletas que pasaron por los 3 puestos.

Encabezado: function cantidadTotal(datos)

Para el ejemplo, retornaría 2.

Ejercicio 2) Luces (máximo 21 puntos)

En el marco del proceso de mejora de iluminación de la ciudad, se necesita un sistema para gestionar las reparaciones y el estado de las luminarias de las diferentes calles.

Diseño HTML+CSS

2a) (máximo 7 puntos): Completar el código HTML de esta única página (index.html) y la hoja de estilo (estilos.css) para obtener un diseño similar al presentado abajo correspondiente a la parte informativa. El código HTML deberá ir dentro de una etiqueta <article>. El resto de la página NO se solicita y se asume disponible.

Archivo **index.html**	Archivo **estilos.css**
<article id="Info"> **2a - CODIGO A COMPLETAR** </article>	**2a - CODIGO A COMPLETAR**

La parte de página solicitada debe ser similar a la mostrada con la siguiente descripción:
Contiene un título (“Información general”) en tamaño 30px y color rojo, dos párrafos con texto centrado (“En la primera etapa. se trabajará en zona centro" y “Los tipos de luminarias están disponibles en este link"). Dentro del texto del segundo párrafo hay un link ("este link") a la página: http://www.luminarias.com. Luego se presenta en una imagen centrada (disponible en img/calle.jpg) con texto alternativo: "Calle iluminada". Debajo hay un titulo más pequeño (“Categoría de luminarias”) y una lista que contiene: luminaria de led, común y de bajo consumo. El fondo de la página es de color #F5F8FB.

Escuela de Ingeniería		
Segundo Parcial de: Programación I		**Código de materia:**
Fecha: Diciembre 2019	**Grupo: Matutino**	**Hoja 3 de 3**

Código JS y HTML (máximo 14 puntos)

En otra parte de la página index.html se tiene un formulario para dar de alta las luminarias.

```
<article >
  <h2>Registro de Luminaria</h2>
 <form id="idFormulario"  >
        <label for="idCodigo" >Código</label>
        <input type="text" id="idCodigo" >
        <label for="idCalle">Calle</label>
        <input type="text" id="idCalle"  >
        <label for="idEstado" >Estado</label>
        <input type="number" id="idEstado"
                    min="1" max="5" >

        <button id="idBotonAgregar" type="button"
                    value="button" >Agregar</button>
  </form>
  <h3 id="idTitulo"> Luminarias de la calle
  </h3>
  <ul id="idLista">
  </ul>
</article>
```

Se tienen además las clases:

```
class Luminaria {
   constructor(codigo, calle,  estado) {
      this.codigo = codigo; // es alfanumérico
      this.calle = calle;
      this.estado = estado;  // estado es un entero entre 1 y 5
   }
   toString(){
        // 2b – A COMPLETAR
  }
}
class Sistema {
   constructor() {
      this.listaLuminarias = [];
   }
    agregarLuminaria(codigo, calle, estado){
          // 2c - A COMPLETAR
          }
    consultaCalles(unaCalle) {
          // 2d - A COMPLETAR
    }
}
```

2b) (máximo 2 puntos) Definir el método toString en Luminaria para que retorne su código en mayúsculas y entre paréntesis la indicación de “bien” si el estado es 5, “regular” si es 2, 3 o 4 y “malo” si es 1.

2c) (máximo 2 puntos): Completar en la clase Sistema el método **agregarLuminaria (codigo, calle, estado)** que da de alta la nueva luminaria con los datos recibidos.

2d) (máximo 5 puntos): Completar en la clase Sistema el método consultaCalles(calle) que retorna un array indexado conteniendo en cada posición una luminaria que esté en la misma calle recibida de parámetro. Debe estar ordenado por estado creciente/código creciente.

2e) (máximo 5 puntos): Se tiene definida una instancia de Sistema: **let sistema = new Sistema();** y se asoció el siguiente evento: **document.getElementById("idBotonAgregar").addEventListener("click", agregar);**. Implementar la función agregar que, si el formulario es válido, toma los datos de pantalla y agrega la luminaria en el sistema (**asumiendo disponible el método de 2c**). Además muestra en la lista idLista todas las luminarias registradas de la misma calle (**asumiendo disponibles métodos de 2b y 2d**). Si es incorrecto el formulario, informarlo mediante un alert.