EJERCICIO 1

Mad Libs es un juego en donde llenas un montón de espacios en blanco de diferentes tipos de palabras, después se genera una historia en base a esas palabras y a veces la historia es sorprendentemente graciosa. Por ejemplo:

```
Selecciona un sustantivo: camion
Selecciona un sustantivo en plural: fresas
Selecciona un sustantivo: palo
Nombra un lugar: Granada
Selecciona un adjetivo (Describe una palabra): rojo
Selecciona un sustantivo: atun
----------------------------------------
Be kind to your camion - footed fresas
For a duck may be somebody's palo ,
Be kind to your fresas in Granada
Where the weather is always rojo .
```

Tienes que hacer el tuyo propio

EJERCICIO 2

Programar los siguientes mini-juegos de dados (random). Debéis controlar los errores asociados a valores incorrectos (negativos, un dado es de 6 caras, etc):

a) Escriba un programa que pida un número de dados, que pida un valor a conseguir y que tire después el número de dados indicado. El jugador gana si saca el valor ganador.

b) Escriba un programa que pida un número de dados, que tire el número de dados indicado y diga cuál es el valor más alto obtenido.

c) Escriba un programa que pida un número de dados y tire esa cantidad para dos jugadores. El jugador que saque el valor más alto, gana.

d) Escriba un programa que pida un número de dados y tire esa cantidad de dados. El primer jugador obtiene un punto por cada dado par. El segundo jugador obtiene un punto por cada dado impar. El jugador que saque más puntos, gana.

e) Escriba un programa que pida un número de jugadores y tire un dado para cada jugador. El jugador que saque el valor más bajo, gana.

f) Escriba un programa que pida un número de dados y tire esa cantidad de dados para dos jugadores. El jugador que saque más puntos sumando su valor más alto y su valor más bajo, gana.

g) Escriba un programa que pida un número de dados y tire esa cantidad de dados. Si no salen dos dados iguales seguidos, el jugador gana. Si salen, pierde.

EJERCICIO 3

Realizar un programa que genere contraseñas aleatorias. El programa pide cuantas contraseñas que se van generar y después la longitud de las mismas. Los caracteres admitidos en las contraseñas son:

0123456789@ABCDEFGHIJKLMNOPQRSTUVXYZ\]_abdefghijklmnpqrstuvwxyz

Consejo: Partir de un String con los caracteres admitidos para una contraseña, usar list para convertirlo en una lista y choice para seleccionar valores aleatorios de la lista

EJERCICIO 4

Escribir un programa que gestione las facturas pendientes de cobro de una empresa. Las facturas se almacenarán en un diccionario donde la clave de cada factura será el número de factura y el valor el coste de la factura. El programa debe preguntar al usuario si quiere añadir una nueva factura, pagar una existente o terminar. Si desea añadir una nueva factura se preguntará por el número de factura y su coste y se añadirá al diccionario. Si se desea pagar una factura se preguntará por el número de factura y se eliminará del diccionario. Después de cada operación el programa debe mostrar por pantalla la cantidad cobrada hasta el momento y la cantidad pendiente de cobro, además de la facturas sin pagar (numero y coste) ordenadas de mayor a menor.

EJERCICIO 5

Escribir un programa que, partiendo de una estructura de datos con los precios de las frutas de una tienda, permita realizar las siguientes acciones:

Fruta	**Precio/Kg**
Plátano	1.35
Manzana	1.80
Pera	0.85
Naranja	0.70

- Dado un número de kilos X mostrar el precio total de llevarse X kilos de todas las frutas. Hacer un resumen de cada cantidad que se va a añadiendo.
- Mostrar la fruta con menor precio.
- Crear otro diccionario a partir del anterior con las frutas ordenadas de mayor a menor precio.
- Crear otro diccionario a partir del anterior con solo las frutas que valgan más de 1€/kh.

EJERCICIO 6

Escribir un programa que permita gestionar la base de datos de clientes de una empresa. Los datos del cliente son DNI/CIF, nombre, dirección, teléfono, correo y deudas. El programa debe preguntar al usuario por una opción del siguiente menú:

(1) Añadir cliente por nombre (no puede haber dos clientes con el mismo nombre)

(2) Eliminar cliente por nombre

(3) Buscar cliente por nombre

(4) Listar todos los clientes

(5) Añadir deuda a cliente

(6) Mostrar clientes ordenados por deuda mayor a menor

(7) Backup datos de clientes (string con separadores y saltos de lineas)

(8) Recuperar datos de backup generado en el punto anterior. Previamente se borran todos los datos.

(9) Crear un diccionario cuya clave sean los nombre de los clientes y el valor su deuda si es mayor de 500€

(10) Salir del programa

Tenéis que hacerlo con al menos 2 estructuras distintas (lista de diccionarios y diccionario de diccionarios). El programa deberá dar una experiencia de usuario correcta y gestionar los datos de manera lógica impidiendo valores sin sentido.