# Pruebas de usuario TP 3

## Conocimientos mínimos esperables

1. Robustez del programa (si rompe en ejecución con alguna de las pruebas implica desaprobar)
2. Manejo de errores (si hay errores en la muestra de datos o pasajeros en blancos implica desaprobar o muy baja nota)
3. Si no cumple la consigna implica desaprobar o muy baja nota (no cumplir la consigna es que falte alguna de las opciones de menú, que no se utilicen las funciones de la Linked List pedidas como mínimo)
4. Pedir id o declarar variables globales implica desaprobar. Que el id no sea autonumerico implica nota muy baja.
5. La falla de guardado de archivos implica desaprobar, lo mismo si no puede abrirse alguno.
6. Se tiene que respetar el sentido de cada biblioteca, su funcionalidad específica. Agregarle cosas que no correspondan a la misma implica baja nota. Se pueden agregar más bibliotecas que ayuden a que cada una de las mismas sean independientes entre sí.
7. El programa debe funcionar si o si con ids únicos. No debe haber repeticiones de los mismos.
8. No se deben mezclar funcionalidades entre las bibliotecas, tiene que haber granularidad consistente y no pueden repetirse funcionalidades.

## Pruebas de funcionales 1:

1. Se chequea que compile
2. Se trata de ingresar a las opciones 4, 5, 6 y 7 sin haber cargado ningún pasajero.
3. Se ingresa 11 como opción de menú. Debería pedir reingreso de dato o volver a mostrar el menú.
4. Se deberá de dar de alta un pasajero y chequear que este se imprima en la lista.
5. Se eliminará el pasajero y habrá que chequear que el id del pasajero eliminado no se reutilice.
6. Se modificará un pasajero (se deben pedir qué campos modificar) y se chequeará que se impriman los cambios.
7. Se prueba levantar el archivo de forma texto y guardar este pasajero.
8. Imprimir y chequear que el id sea consecutivo y no pise algún otro pasajero.

*Objetivo: ver que se tome en cuenta el id de forma externa y que no se pise en ningún momento*

## Pruebas funcionales 2:

1. Se chequea que compile.
2. Se trata de ingresar a las opciones de imprimir sin ningún pasajero dado de alta.
3. Se levanta el archivo csv, dos veces.
4. Se intentará levantar el archivo binario, esto debe ser controlado ya que puede generar que se carguen las listas en memoria dos veces.
5. Se deberá dar de alta un pasajero, realizando las validaciones pertinentes.
6. Se guardará.
7. Se stopea el programa.
8. Se levanta el archivo y se chequea que el pasajero se haya creado.

9. Si existe un archivo binario se elimina y se trata de salvar como binario, para ver si se crea el mismo sin pisar el anterior.

*Objetivo: Que la funcionalidad de los archivos, crear y guardar sea correcta*

# Alta de pasajero:

## Pruebas para campos de tipo string:

Primer ingreso:
**NOMBRE**: 1234
Se espera mensaje de error y que se reintente el ingreso del nombre

Segundo ingreso:
**NOMBRE:** (enter o vacío)
Se espera mensaje de error y que se reintente el ingreso del nombre

Tercer ingreso:
**NOMBRE:** Mariela
todo ok

## Pruebas para campos numéricos:

Primer ingreso:
**SALARY**: asd
Se espera mensaje de error y que se reintente el ingreso del salario

Segundo ingreso:
**SALARY:** (enter o vacío)
Se espera mensaje de error y que se reintente el ingreso del salario

Tercer ingreso:
**SALARY:** -3
Se espera mensaje de error y que se reintente el ingreso del salario

Cuarto ingreso:
**SALARY:** 1213
todo ok