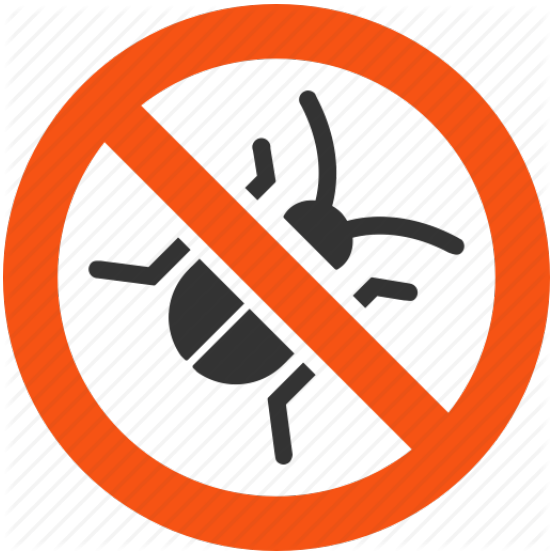
laSalle
Universitat Ramon Llull

Curso 2018 - 2019
Proyectos de Programación 1

Ejercicio de Debugging



The point of writing problem report (bug report) is to get bugs fixed.

Cem Kaner

laSalle
Universitat Ramon Llull

Curso 2018 - 2019
Proyectos de Programación 1

ÍNDICE

1. INTRODUCCIÓN

Dado que la resolución de errores o "bugs"de un programa es una tarea que los desarrolladores deben afrontar a menudo, el objetivo de este ejercicio es que el alumno aprenda y experimente las distintas fases del proceso de detección y corrección de errores que tienen lugar en el ciclo de desarrollo y mantenimiento de un programa.

En el ejercicio actual, el alumno se deberá enfrentar a un programa que experimenta un conjunto de errores sintácticos, de funcionamiento y/o de diseño. El alumno deberá utilizar las herramientas aprendidas en clase, así como los recursos de los que quiera disponer, para detectar y posteriormente corregir, todos (o el mayor número posible) los errores del programa.

Para reportar cada uno de los errores, se deberá hacer uso de una plantilla especificada por la asignatura. Dicha plantilla seguirá algunas de las *guidelines* recomendadas y utilizadas en el mundo real.

2. EL PROYECTO

Los alumnos deberán enfrentarse a un proyecto escrito en lenguaje C y estructurado en módulos.

El código fuente del proyecto se podrá descargar desde el repositorio de GitHub `https://github.com/PPROG1-1819/Trip-Organizer`.

Cada alumno deberá clonar el repositorio fuente a un repositorio propio de visibilidad pública y trabajar sobre este repositorio para detectar y corregir los posibles errores.

Finalmente, el alumno deberá generar un documento en el que, por cada uno de los errores detectados, se identifique, explique y resuelva el problema.

2.1. ELEMENTOS DEL *report*

Como ya se ha explicado, cada uno de los errores detectados en el programa se deberá reportar siguiendo un formato específico. En la figura 1 podemos ver un ejemplo de la plantilla que vamos a utilizar en nuestro informe.

ID		Status		Severity	
Description					
Steps to reproduce					
Expected result					
Actual result					
Solution					

Figura 1: Plantilla para reportar errores

1. **ID**: Identificador numérico único e incremental.
2. **Status**: Estado actual del bug. Este puede ser *Resuelto* (*Solved*) o *Sin resolver* (*Unresolved*).
3. **Severity**: Gravedad del bug

- *Blocker*: Debido a este bug, nos es imposible seguir testeando el programa.
- *Critical*: Este bug provoca que el programa deje de funcionar.
- *Major*: Este bug provoca que una funcionalidad entera deje de funcionar.
- *Minor*: Este bug provoca que una funcionalidad deje de funcionar parcialmente.
- *Trivial*: Este bug se puede mejorar con algunas mejoras de la UI.

4. **Description**: Descripción detallada del problema.
5. **Steps to reproduce**: Pasos a seguir para poder reproducir el error.
6. **Expected result**: Resultado esperado tras ejecutar la funcionalidad que contiene el error.
7. **Actual result**: Resultado recibido tras ejecutar la funcionalidad que contiene el error.
8. **Solution**: Explicación detallada de como se ha conseguido resolver el error (si está resuelto).

A continuación, en la figura 2 y en la figura 3, podéis ver un par de reportes de ejemplo.

ID	#23	Status	Solved	Severity	Critical
Description	Program crashed with Segmentation Fault (Core Dumped) after printing all the students with the grades.				
Steps to reproduce	Just run the program.				
Expected result	A list with the name of all students and their grades.				
Actual result	The list, plus a core dumped.				
Solution	In line 55 from students.c, the loop which accessed the array of students accessed position *MAX_STUDENTS* which did not exist. We replaced the condition from `i<=MAX_STUDENTS` to `i<MAX_STUDENTS` to solve the problem.				

Figura 2: Ejemplo 1 de reporte de bug simplificado

ID	#24	Status	Unresolved	Severity	Major
Description	The program is not sorting the students by grade (from 10 to 0), only when the students' grades are already inversely sorted (from 0 to 10).				
Steps to reproduce	1. Insert a set of 3 students 2. Insert the grades for each student being 0, 5 and 10 3. Run option 3 on the menu (sort by grade) 4. Get the list of students from 0 to 10 (inverse order)				
Expected result	A list of students sorted by grade from 10 to 0				
Actual result	A list of students sorted by grade from 00 to 10				
Solution	---				

Figura 3: Ejemplo 2 de reporte de bug simplificado

3. NORMATIVA DE ENTREGA

En este apartado se concretará la normativa de entrega del ejercicio.

3.1. GRUPOS

El trabajo se deberá realizar de forma individual.

3.2. FECHAS DE ENTREGA

La fecha máxima de entrega del ejercicio será el día **7 de enero del 2019**.

3.3. FORMADO DE ENTREGA

El ejercicio se tendrá que entregar en el pozo correspondiente. Habrá que colgar un único fichero correspondiente al report del ejercicio, en **PDF**.

3.4. FORMATO DEL REPORT

El documento que se entregue en formato PDF deberá contener los siguientes apartados:

1. Portada con el nombre del alumno
2. Índice
3. Reports de cada uno de los bugs siguiendo el template dado por la asignatura

4. Conclusiones del ejercicio
5. Link al repositorio de GitHub público utilizado para resolver los bugs