DESARROLLO FORMAL DE SOFTWARE

SEGUNDA ENTREGA DEL PROYECTO

CONSTRUCCIÓN, CONTRATOS Y VERIFICACIÓN INICIAL DE FUNCIONALIDADES

Continuidad del proyecto. Se conserva el mismo proyecto y las tres funcionalidades seleccionadas en la primera entrega. Ahora deben mostrar cómo se implementan en código las condiciones que especificaron y cómo comprobaron el comportamiento.

Información	Indicaciones
Modalidad	Trabajo grupal, máximo 3 integrantes.
Producto	Formato diligenciado en un único PDF, código fuente o enlace accesible y capturas de ejecución.
Alcance	Revisar las tres funcionalidades; no se exige un sistema comercial completo, pero sí código funcional y evidencias suficientes.
Lenguaje	Java, C++, Python u otro lenguaje usado por el grupo, siempre que el código sea revisable y ejecutable.
Entrega	Cargar en AVAS según fecha e indicaciones de la docente.

1. Propósito

Pasar del análisis y especificación de la primera entrega a la construcción y verificación inicial del código. Deben demostrar la relación entre PRE, POST, instrucciones, estructuras de control, resultados esperados y resultados observados.

2. ¿Qué deben hacer?

1. Recuperar las tres funcionalidades de la primera entrega. Mantengan el mismo alcance o expliquen cualquier cambio justificado.
2. Revisar qué está implementado en el código y qué falta. No basta con copiar el análisis anterior: señalen el código que responde a cada condición.
3. Para cada funcionalidad, escribir la PRE (qué debe cumplirse antes de ejecutar) y la POST (qué debe cumplirse después). Indicar dónde se validan las condiciones y dónde se produce el resultado esperado.
4. Como requisito del proyecto, al menos una funcionalidad debe contener una estructura condicional (if/else o equivalente) y otra funcionalidad debe contener un ciclo (for/while o equivalente). Identifiquen exactamente cuáles son.
5. Verificar cada funcionalidad con, como mínimo, una ejecución válida y una ejecución no válida o de borde, según corresponda. Comparen el resultado esperado con el observado y agreguen capturas legibles.
6. Revisar los conceptos del segundo corte: ejecución secuencial, condicionales, ciclos, invariantes y terminación, correctitud parcial y total, funciones/métodos, algoritmos, recursividad y verificación orientada a objetos cuando corresponda.
7. Si encuentran una diferencia, documentar el problema, la corrección y la nueva ejecución. Si no encuentran diferencias, registrarlo y presentar las evidencias que respaldan esa conclusión.

3. Conceptos que deben revisar

Concepto	Evidencia que deben señalar
Secuencia	Instrucciones ejecutadas en orden y cómo cambian los datos.
Condicional	Condición evaluada, ramas posibles y comportamiento de cada rama relevante.
Ciclo	Inicialización, condición, actualización y condición de salida; explicar por qué no queda ejecutándose indefinidamente.
Invariante y terminación	Propiedad que se mantiene durante las iteraciones y explicación de cómo el ciclo progresa hasta terminar.
Correctitud parcial	Por qué, si el algoritmo termina, el resultado satisface la POST.
Correctitud total	Además de la corrección del resultado, argumento de que el algoritmo termina.
Funciones/métodos y algoritmos	Parámetros, resultado o efecto, contrato y pasos que resuelven el problema.
Recursividad	Si se usa: caso base, caso recursivo y terminación. Si no se usa: explicar por qué no aplica; no agregarla artificialmente.
Orientación a objetos	Si se usan clases/objetos: estado, contratos de métodos e invariantes de clase. Si no aplica, justificar.

4. Evidencias mínimas para cada funcionalidad

- Código fuente o enlace accesible al código completo, además de los fragmentos pertinentes identificados por archivo/línea.
- PRE, POST y explicación de cómo se reflejan en el código.
- Una o más condiciones de verificación y explicación de cómo se comprueban.
- Una ejecución válida y otra no válida o de borde, con entradas, resultados esperados, resultados observados y capturas.
- Análisis de las estructuras y conceptos aplicables. Cuando un concepto no aparezca, indicar “No aplica” y justificarlo.
- Problemas encontrados, correcciones y nueva evidencia, o declaración explícita de que no se identificaron diferencias.

5. Orden sugerido del PDF

1. Identificación del proyecto e integrantes.
2. Resumen de continuidad con la primera entrega y listado de las tres funcionalidades.
3. Desarrollo de las funcionalidades 1, 2 y 3 usando el formato.
4. Matriz integradora de conceptos del segundo corte.
5. Conclusiones: qué se verificó, qué problemas se encontraron y qué queda pendiente.
6. Referencias o enlace al código y capturas.

6. Aclaraciones importantes

- No basta con que el programa se ejecute: deben explicar por qué el resultado satisface las condiciones especificadas.
- No peguen el código completo sin señalar los fragmentos relevantes.
- No se exige que cada funcionalidad contenga todos los conceptos; sí se exige al menos una funcionalidad con condicional y otra con ciclo.
- Recursividad y orientación a objetos se analizan cuando correspondan al código; no deben añadirse artificialmente.
- Las capturas no sustituyen el código.
- Las tres funcionalidades deben conservar coherencia con la primera entrega. Si cambiaron algo, justifiquen por qué.

7. Rúbrica sugerida (100 puntos)

Criterio	Descripción	Puntos
Continuidad y especificación	Mantiene las tres funcionalidades y su relación con la primera entrega.	10
Implementación y organización	Código funcional, legible, revisable, con entradas, validaciones y resultados coherentes.	15
Contratos PRE/POST	Contratos correctos y relación concreta con el código.	15
Estructuras y verificación	Secuencia, condicional y ciclo; análisis de invariante, terminación y correctitud cuando corresponda.	20
Ejecuciones y evidencias	Casos válidos y no válidos/de borde para las tres funcionalidades; capturas y comparación.	15
Funciones, algoritmos, recursividad y objetos	Aplica conceptos cuando correspondan y justifica los que no aplican.	10
Correcciones y calidad	Registra discrepancias/correcciones y vincula resultados con calidad.	10
Organización y presentación	Documento coherente, PDF legible y código accesible.	5

8. Lista de verificación

- ☐ Se mantienen las tres funcionalidades o se justifican cambios.
- ☐ Cada funcionalidad tiene PRE y POST.
- ☐ Se identifican fragmentos de código que implementan y comprueban las condiciones.
- ☐ Hay condicional en al menos una funcionalidad y ciclo en otra.
- ☐ Se analizan invariante, terminación y correctitud parcial/total cuando corresponda.
- ☐ Se revisan funciones/métodos, algoritmos, recursividad y objetos cuando aplique.
- ☐ Hay ejecuciones y capturas para las tres funcionalidades.
- ☐ Se comparan resultados esperados y observados y se documentan correcciones.
- ☐ El código está accesible y el PDF es legible.

8. Entrega: 10 de noviembre de 2026 hasta las 7:00pm

El archivo debe estar realizado en el formato establecido.

Nombrar el documento como segunda entrega y un nombre de cada integrante. Evite el uso de caracteres especiales al nombrar los documentos, con el fin de prevenir errores de visualización. [#, %, ñ, &, *, :, ?, ¡, !, /, \, ", ', , |, =, @,(],

Se recomienda cargar un único archivo PDF en el AVA institucional con los nombres de todos los integrantes.

La entrega puede realizarse antes de la fecha máxima.

Nota: El no cumplimiento de este punto puede rebajar la nota.