|  |
| --- |
| **Calificación (reservado para el docente):** |
| **Criterios de evaluación:**   * Correcto manejo y aplicación de conceptos prácticos trabajados en clase. * Claridad en la resolución y completitud de las respuestas. * El examen deberá ser escrito totalmente en tinta en caso de que la resolución sea en papel. * El tiempo de resolución es de 1 hora y 45 minutos. * El alumno deberá subir la resolución en la plataforma de EVEA antes de que la actividad finalice. |
| **Puntuación:**   * El enunciado 1 vale 5 puntos. * El enunciado 2 vale 2 puntos. * El enunciado 3 vale 3 puntos. |

1. **Dado el siguiente fragmento de código de un programa escrito en lenguaje C:**

int sumarMayoresQueCienOY(int vec[], int N, int Y)

{

int var=0;

for(int i = 0; i < N; i++)

if(vec[i]>100 || vec[i] > Y)

var = var + vec[i];

return var;

}

1. Obtener la complejidad computacional de la función “sumarMayoresQueCienOY” a partir de un grafo dirigido.
2. Indicar cuáles caminos son válidos de recorrer y cuáles no. Ejemplificar utilizando datos representativos para los caminos válidos.
3. Indique que casos de prueba necesita para cumplir con el 100% de cobertura de **sentencia**.
4. Desarrollar una prueba unitaria que cubra el 100% de las **condiciones** de la función “sumarMayoresQueCienOY”. La función test a desarrollar debe retornar 0 si el test resultó OK y debe retornar un valor distinto de 0 si el test resultó Fallido.
5. **Dados el siguiente prototipo de función en lenguaje C:**

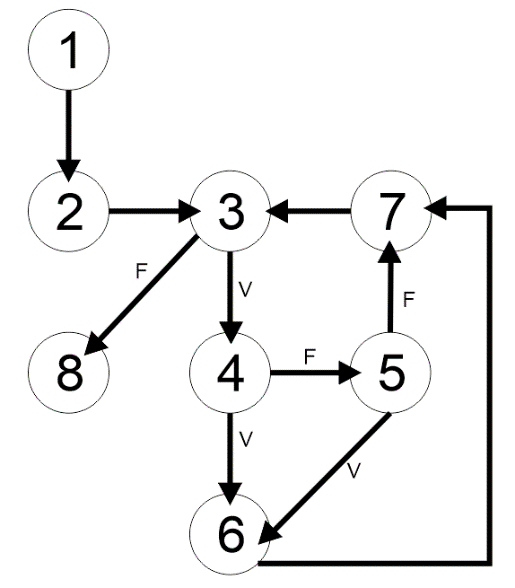
*int cantidadDeParesNegativos (int[], int N)*

La función recibe por parámetro un array de números enteros y la cantidad de elementos del array, para contabilizar cuantos números son pares negativos. Devuelve como resultado la cantidad de enteros pares y que a su vez son negativos. Si todos los números son impares o positivos, el resultado es 0.

**Se pide programar al menos 2 casos de prueba unitaria.** La función test a desarrollar debe retornar 0 si el test resultó OK y debe retornar un valor distinto de 0 si el test resultó Fallido.

1. **Diseñar los casos de prueba correspondientes al siguiente caso de uso, aplicando la técnica de Tablas de Decisión.**

|  |
| --- |
| **Alta de licencia de conducir** |
| Descripción: Este caso de uso permite dar de alta una licencia de conducir |
| Actor/es: Personal Administrativo |
| Pre-condiciones: La persona que solicita el alta de licencia debe tener aprobado los exámenes teórico y práctico. |
| Flujo Normal:   1. El actor selecciona la opción “Alta de Licencia”. 2. El sistema pide el ingreso del DNI de la persona. 3. El actor ingresa el DNI de la persona. 4. El sistema busca a la persona para verificar que no tenga licencia. 5. El sistema pide por pantalla los datos faltantes de la persona (nombre y apellido, dirección, categoría, sexo, factor sanguíneo). 6. El actor ingresa los datos y presiona aceptar. 7. El sistema verifica que la persona tenga los exámenes aprobados. 8. El sistema muestra por pantalla la confirmación y agrega los datos del nuevo conductor (fecha de otorgamiento y fecha de vencimiento). 9. El sistema guarda en la base de datos el registro del nuevo conductor. |
| Flujo alternativo 1:  4.1 El sistema informa que el conductor ya existe y no puede dar de alta la licencia.  4.2 El sistema informa los pasos a realizar en esta ocasión.  4.3 Vuelve al paso 1 del flujo normal. |
| Flujo alternativo 2:   * 1. El sistema informa que la persona no aprobó algún examen.   2. Vuelve al paso 1 del flujo normal. |
| Excepciones:  E1: 1. El sistema informa que hubo un error al guardar los datos.  2. Vuelve al paso 1 del flujo normal.  E2: 1. El usuario no completa todos los datos de la persona para el alta del registro.  2. El sistema resalta los campos vacíos y permite continuar con el flujo normal. |



1.a

Complejidad computacional es de 4.

CC= aristas – nodos + 2

CC= 10 – 8 + 2 = 4.

1.b

|  |  |
| --- | --- |
| Caminos | Valido/Invalido |
| 1,2,3,4,5,6,7,3…8 | Valido |
| 1,2,3,4,5,7,3…8 | Valido |
| 1,2,3,4,6,7,3..8 | Valido |
| 1,2,3,8 | Invalido |

1.c El caso de prueba que abarca el 100% de la función es el camino 1,2,3,4,5,6,7,3…8.

1.d

int test (){

int arr[]={11,11,22};

int N=3;

int Y=10;

int valorEsperado=44;

if(sumarMayoresQueCienOY(arr,N,Y)!=valorEsperado){

return -1;}

return 0;}

2.

int test(){

//caso 1

int arr1[]={-2,-4,-6,-8};

int resEsperado1=4;

int N=4;

if(cantidadDeParesNegativos (arr1,N)!=resEsperado1){

return -1; }

return 0;

//caso 2

int arr2[]={1,-1,3,-3};

int resEsperado=0;

int N=4;

if(cantidadDeParesNegativos (arr2,N)!=resEsperado){

return -1; }

return 0;

//caso 3

int arr3[]={2,4,6,8};

int resEsperado=0;

int N=4;

if(cantidadDeParesNegativos (arr3,N)!=resEsperado){

return -1; }

return 0;

}

|  |  |  |
| --- | --- | --- |
| **Condiciones** |  |  |
| **Ingresa DNI** | **Si** | **No** |
| **Verifica exámenes** | **Si** | **No** |
| **Completa datos** | **Si** | **No** |

|  |
| --- |
| Acciones |
| Guardar datos |
| Usuario existente |
| Usuario no paso exámenes |
| Error datos |
| Faltan datos |

