ESTUDIOS
GENERALES
CIENCIAS

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

FUNDAMENTOS DE PROGRAMACIÓN

EXAMEN FINAL
SEMESTRE ACADÉMICO 2021-2

Horarios: Todos
Duración: 180 minutos
Revisado por los profesores del curso.

INDICACIONES DEL EXAMEN

- El examen final se realizará el día viernes 10 de diciembre desde las 15:00 horas hasta las 18:00 horas.
- La rúbrica para la corrección del examen se encontrará publicada desde la semana anterior al examen en la plataforma PAIDEIA, en el curso 2021-2 FUNDAMENTOS DE PROGRAMACIÓN (1INF01), sección ”Examen Final”.
- Los enunciados de las preguntas se irán habilitando, en forma progresiva, en la plataforma PAIDEIA en el curso 2021-2 FUNDAMENTOS DE PROGRAMACIÓN (1INF01) en la sección ”Examen Final”
    - El día viernes 10 de diciembre se ocultarán todas las secciones del curso en la plataforma PAIDEIA desde las 12:00 horas hasta las 19:00 horas, quedando sólo habilitada la sección del examen respectivo. No se podrá tener acceso a material alguno durante las horas indicadas. Esto con el fin de que la sección ”Examen Final” sea ubicada con facilidad.
    - Ninguna información relacionada al examen final aparecerá en las secciones de PAIDEIA de los cursos por horario. Estas secciones no se usarán durante el desarrollo del examen.
- El examen debe rendirlo en un computador que tenga el software PSeInt para desarrollar pseudocódigos, así como Dev-C++ o Visual Studio Code, para desarrollar programas en lenguaje C. Solo se puede usar Visual Studio Code si cuenta con sistema operativo macOS. Es su responsabilidad verificar que su computador y los softwares requeridos funcionen correctamente. Se recomienda realizar las pruebas necesarias para verificarlo.
- Debe verificar, el día anterior al examen, que su computador tenga instalada la última versión de Zoom.
- Debe verificar que el ingreso a Zoom se realice con la cuenta Zoom asociada al usuario PUCP. Si se conecta como invitado a Zoom no ingresará a la sesión del examen.
- En la sección ”Examen Final” en la plataforma PAIDEIA en el curso 2021-2 FUNDAMENTOS DE PROGRAMACIÓN (1INF01), se encontrarán los enlaces para poder iniciar la sesión en Zoom. Esta se encontrará habilitada desde 15 minutos antes de la hora de inicio del examen. Las sesiones de Zoom están separadas por intervalo de códigos de alumnos, NO POR HORARIO. Debe ingresar a la sesión de Zoom del rango de códigos al que pertenezca su código de alumno.
- El día del examen a la hora programada, dentro de la sección ”Examen Final”, se encontrarán 3 tareas configuradas para cada pregunta (Pregunta 1, Pregunta 2 y Pregunta 3). En cada una de estas tareas, se encontrará el enunciado correspondiente. Deberá asegurarse que su computador pueda leer archivos en formato PDF. Este formato es el mismo que se ha usado para las presentaciones del curso, los enunciados de problemas y los enunciados de laboratorio durante todo el semestre.
- Cada alumno deberá conectarse a la plataforma Zoom 15 minutos antes de la hora programada del examen. Al ingresar debe colocar su código y nombre en el chat. El nombre del usuario de Zoom debe ser su nombre y apellidos.
- Cuando el Jefe de Laboratorio le indique, deberá iniciar la grabación con el Zoom, aceptar la invitación que se le envíe en forma inmediata para acceder a su sala personal. En la sala personal, debe compartir todo su escritorio, tener el micrófono y la cámara activos. La cámara debe enfocar en todo momento el rostro completo de frente del alumno, el cual debe mostrarse en forma nítida. El alumno no puede usar audífonos, a excepción de que se le haya permitido utilizarlo en los laboratorios.
- El alumno debe tener el mismo comportamiento que tendría en una evaluación presencial.
- El Jefe de Laboratorio le solicitará verificar su identidad, para lo cual el alumno debe mostrar a la cámara su DNI, TI u otro documento de identidad con fotografía.
- Los jefes de laboratorio, estarán cuidando el examen y estarán visitando las salas. A los alumnos que no compartan todo su escritorio, no tengan su micrófono y cámara activos NO SE LES CORREGIRÁ EL EXAMEN, y se les pondrá la nota cero(0). Ante cualquier problema técnico que pueda existir para compartir el escritorio, deberá comunicarse con el jefe de laboratorio.
- Si el alumno necesita usar una hoja durante el desarrollo del enunciado, puede únicamente usar una hoja en blanco; pero antes de ello deben llamar al Jefe de Laboratorio para mostrar dicha hoja a la cámara y él autorizará el uso de esta. A los alumnos que usen una hoja en blanco se les supervisará más seguido durante el examen.
- No está permitido utilizar otros dispositivos ajenos al computador donde se resuelve la evaluación, por ejemplo, tablet, celular, otra laptop u otro computador. Este incumplimiento se analizará como un caso de plagio.
- Si se detectan posibles casos de plagio, el jefe de laboratorio notificará a los profesores y la junta de profesores revisará cada caso. En caso que la junta de profesores compruebe la falta, se procederá a anular el examen, a comunicar a EEGGCC y se proseguirá como indica el ”Reglamento Unificado de Procedimientos Disciplinarios de la Universidad”.

INDICACIONES DEL EXAMEN (2):

- **Durante el desarrollo del examen, mientras desarrolla cada pregunta debe explicar oralmente con sus palabras lo que está realizando. En caso de no hacerlo, se le asignará la nota cero(0) en la pregunta respectiva.**

- Cada pregunta del examen final, deberá resolverse en 50 minutos. Pasados los 50 minutos por pregunta, el alumno tendrá 10 minutos para subir sus archivos a la plataforma PAIDEIA:

  - La pregunta 1 aparecerá en la plataforma PAIDEIA desde la hora del inicio del examen, es decir desde las 15:00 horas. Existirán diversas variantes de enunciados para la pregunta, la identificarán fácilmente por que se le agregará una letra al número de la pregunta, por ejemplo "Pregunta 1A", "Pregunta 1B", y así sucesivamente, dependiendo de la pregunta que les toque (a cada alumno le tocará una pregunta 1 de manera aleatoria). El alumno tendrá hasta las 15:50 hrs para responder esta pregunta. Los siguientes 10 minutos, el alumno deberá dedicarlos a subir el archivo con su solución a la plataforma PAIDEIA (desde las 15:50 hrs. hasta las 15:59 hrs). El archivo debe ser grabado con el nombre **E2_código_pregunta1.psc** (donde la palabra código debe ser reemplazada por el código del alumno, por ejemplo, **E2_20206666_pregunta1.psc**, sin espacios). La plataforma PAIDEIA NO ACEPTARÁ archivos luego de dicho lapso de tiempo (a las 16:00 hrs no se permitirá la subida de archivos de esta pregunta). La solución de la pregunta 1 debe ser subida en la tarea correspondiente a la pregunta 1, sólo así la pregunta será considerada en la calificación. **Antes de subir la pregunta 1 a PAIDEIA debe asegurarse que se ha grabado correctamente en PseInt, es responsabilidad del alumno verificar que el archivo contenga lo que desarrolló. No se aceptará ningún pedido de corrección debido a que la pregunta no tiene contenido o que por este motivo no puede colocarse en la tarea de PAIDEIA.**
  - La pregunta 2 aparecerá en la plataforma PAIDEIA desde el inicio de la segunda hora del examen, desde las 16:00 hrs. Existirán diversas variantes de enunciados para la pregunta, la identficarán fácilmente por que se le agregará una letra al número de la pregunta, por ejemplo "Pregunta 2A", "Pregunta 2B", y así sucesivamente, dependiendo de la pregunta que les toque (a cada alumno le tocará una pregunta 2 de manera aleatoria). El alumno tendrá hasta las 16:50 hrs para responder esta pregunta. Los siguientes 10 minutos, el alumno deberá dedicarlos a subir el archivo con su solución a la plataforma PAIDEIA (desde las 16:50 hrs. hasta las 16:59 hrs). El archivo debe ser grabado con el nombre **E2_código_pregunta2.c** (donde la palabra código debe ser reemplazada por el código del alumno, por ejemplo, **E2_20206666_pregunta2.c**, sin espacios). La plataforma PAIDEIA NO ACEPTARÁ archivos luego de dicho lapso de tiempo (a las 17:00 hrs no se permitirá la subida de archivos de esta pregunta). La solución de la pregunta 2 debe ser subida en la tarea correspondiente a la pregunta 2, sólo así la pregunta será considerada en la calificación.
  - La pregunta 3 aparecerá en la plataforma PAIDEIA desde el inicio de la tercera hora del examen, desde las 17:00 hrs. Existirán diversas variantes de enunciados para la pregunta, la identificarán fácilmente por que se le agregará una letra al número de la pregunta, por ejemplo "Pregunta 3A", "Pregunta 3B", y así sucesivamente, dependiendo de la pregunta que les toque (a cada alumno le tocará una pregunta 3 de manera aleatoria). El alumno tendrá hasta las 17:50 hrs para responder esta pregunta. Los siguientes 10 minutos, el alumno deberá dedicarlos a subir el archivo con su solución a la plataforma PAIDEIA (desde las 17:50 hrs. hasta las 17:59 hrs). El archivo debe ser grabado con el nombre **E2_código_pregunta3.c** (donde la palabra código debe ser reemplazada por el código del alumno, por ejemplo, **E2_20206666_pregunta3.c**, sin espacios). La plataforma PAIDEIA NO ACEPTARÁ archivos luego de dicho lapso de tiempo (a las 18:00 hrs no se permitirá la subida de archivos de esta pregunta). La solución de la pregunta 3 debe ser subida en la tarea correspondiente a la pregunta 3, sólo así la pregunta será considerada en la calificación.

- Todas las preguntas que se han propuesto han sido revisadas por todos los profesores del curso. Se está garantizando que: todas las preguntas puedan ser solucionadas en menos de 50 minutos y que todas las preguntas abarquen los temas vistos en clase.

- Durante la evaluación solo se puede tener abierto el Zoom, plataforma PAIDEIA, un visor de PDF, PseInt o el IDE que se utilice para desarrollar los pseudocódigos o programas y Discord. En Discord solo puede tener abiertos los canales definidos para realizar consultas sobre las preguntas del examen. No está permitido el uso de ningún material, archivo u otro aplicativo (correo, WhatsApp, Facebook, Twitter, Calculadora, Spotify u otro diferente), tampoco puede abrir otra pestaña en el navegador. Debe apagar su teléfono móvil y tenerlo cerca para mostrarlo al Jefe de Laboratorio si se lo solicita. El incumplir estas indicaciones, será motivo de anulación del examen.

- Está terminantemente prohibido tener comunicación con cualquier persona que no sea el Jefe de Laboratorio o el profesor, sin importar el medio utilizado. De igual manera, se prohibe también la comunicacion entre alumnos, sin importar el medio utilizado. Si se demuestra que dos o más alumnos se han comunicado durante el examen, será motivo para la anulación total de los exámenes pertenecientes a los alumnos involucrados.

- No está permitido compartir total o parcialmente soluciones del examen. Si se identifican dos o más desarrollos iguales, se procederá a anular la totalidad del examen de todos los alumnos involucrados. La Coordinación del Curso aplicará las herramientas disponibles para encontrar desarrollos iguales entre las soluciones presentadas por los alumnos.

- El profesor del curso puede en cualquier momento ingresar a la sala personal del alumno y hacerle alguna consulta sobre el desarrollo que usted está realizando. El objetivo de esta consulta es verificar que el alumno es quién está resolviendo el problema, por lo tanto, debe estar en la capacidad de explicar la implementación que realiza. En caso sus respuestas no evidencien que el alumno está desarrollando el examen, el profesor lo reportará a la Coordinación del Curso quién analizará el caso y determinará si se corregirá el examen o se anulará la totalidad de este.

- Se considera presente y se califica el examen a los alumnos que se conectan a Zoom, que graban el desarrollo de todo el examen y suben la grabación al Classroom. También se considera presente a los alumnos que ingresan a cualquiera de los enunciados de las preguntas de PAIDEIA pero que no se conectan a Zoom, que no realizan la grabación de todo el examen o no suben la grabación a Classroom, en estos casos se les asigna la nota cero(0),porque no es posible comprobar que fueron ellos quiénes desarrollaron el examen.

- No se considera problema alguno no subir el archivo dentro del tiempo establecido, por haberlo usado para continuar con la solución del problema. Tampoco se considera error, no saber usar la plataforma PAIDEIA o las herramientas PSEINT, DEVC++ o VISUAL STUDIO CODE.

INDICACIONES DEL EXAMEN (3):

- En caso de que usted tuviera algún problema de conexión para ingresar al examen final deberá rendir el examen especial. Si ingresó al examen final y durante el mismo presenta un problema de conexión deberá justificar con las evidencias respectivas el problema de conexión, para lo cual debe proceder como se indica en el siguiente ítem.
- En caso de que usted tuviera algún problema específico, diferente a los mencionados anteriormente puede comunicarse con la Coordinación del Curso, con la profesora Silvia Vargas a través del correo silvia.vargas@pucp.edu.pe. Deberá guardar evidencia del problema que está teniendo. Se consideran evidencias: fotos, videos, pantallas de error, etc. En las evidencias debe quedar claramente definida la hora y la fecha de lo ocurrido. Para analizar el caso es necesario que el alumno presente las evidencias, si no se envían evidencias el caso no se analizará. El correo debe tener el siguiente formato:
  - Asunto: 1INF01 - Problema especial para el Examen Final.
  - Cuerpo del correo:
    Horario:
    Código:
    Apellidos:
    Nombres:
    Descripción del problema:
  - Adjuntar evidencias: fotos, videos o pantallas de error; donde se visualice la fecha y hora del problema ocurrido.
- **En caso el correo no siga el formato establecido, no se analizará la solicitud.**
- **Los correos recibidos se responderán durante la semana siguiente a la semana del examen.**
- **La Coordinación del Curso enviará un correo a los alumnos con el enlace correspondiente a Classroom para el examen, en el cual debe subir la grabación del examen correspondiente. El plazo para subir la grabación es hasta las 23:59 horas del domingo 12 de diciembre. Si no sube la grabación, no se corregirá el examen y se le asignará la nota cero(0). De tener inconvenientes con la grabación debe escribir un correo a la Coordinación del Curso indicando los problemas que existen con la grabación respectiva y adjuntando las evidencias que lo sustenten. En caso no adjunte la evidencia, no se analizará el caso y se le asignará la nota cero(0). No se responderá ningún mensaje enviado por Classroom.**
- Durante el proceso de corrección, el profesor puede determinar que requiere consultar al alumno sobre la solución que realizó. En este caso, el profesor escribirá un correo al alumno para agendar una reunión por Zoom, a esta reunión el alumno debe asistir puntualmente, compartir todo su escritorio, tener el micrófono y la cámara activos, y responder las preguntas que realice el profesor. En caso las respuestas no evidencien que el alumno es el que desarrolló el examen, el profesor lo comunicará a la Coordinación del Curso quién analizará el caso y determinará si se corregirá la pregunta o se anulará todo el examen.

SOBRE DUDAS Y ACLARACIONES RESPECTO A LOS ENUNCIADOS:

- **Si, durante la evaluación, tiene alguna duda sobre el enunciado, debe realizarla únicamente a través del canal de Discord específico para la pregunta, al cual debe estar conectado desde su computador. El servidor de Discord será creado por la Coordinación del Curso previo al examen y se le comunicará oportunamente, para que pueda unirse con anticipación. Debe probar el ingreso al servidor de Discord antes del examen.**
- En este servidor de Discord, existirá un canal por cada pregunta del examen final. De esta manera, si por ejemplo tuviera dudas acerca del enunciado de la pregunta 1A, podrá acceder al canal pregunta-1a, y escribir su consulta allí. Un profesor del curso responderá sus consultas por chat. No podrá conversar con los otros alumnos en este canal, solo podrá conversar con los profesores. El incumplir esta indicación, será motivo de anulación del examen.
- No está permitido que durante el examen, esté conectado a otros canales en Discord que no sean los exclusivos para realizar consultas sobre las preguntas del examen. El incumplir esta indicación, será motivo de anulación del examen.

INDICACIONES GENERALES:

- Debe utilizar variables y constantes descriptivas, comentarios que expliquen el funcionamiento de la solución y mensajes descriptivos.
- El orden y la eficiencia de su implementación serán considerados en la calificación.

Pregunta 1 – Pseudocódigo con iterativas anidadas (6 puntos)

Pregunta 1A (6 puntos) *[propuesta por Jennifer Zárate]*

En el siglo XII, a la fracción se llamaba FRACTIO que significa quebrar o romper. Es por ello, que antiguamente a las fracciones también se le conocían como quebrados. Su origen se da desde la época de los griegos y se empiezan a utilizar por la necesidad de repartir. Por ejemplo, se usaban comúnmente para los materiales de las construcciones, distribución del pan, medidas de la tierra, entre otros.

Pueden realizarse diferentes operaciones entre fracciones, así como con los números enteros, por ejemplo: resta, multiplicación, potenciación, radicación, etc.

Las operaciones de suma y resta entre fracciones heterogéneas se desarrollan utilizando el cálculo del Mínimo Común Múltiplo (mcm) de los denominadores, de manera que se convierten a fracciones con el mismo denominador y equivalentes en su numerador. Para realizar el cálculo se utilizará lo siguiente:

$$\frac{A}{B} \pm \frac{C}{D} = \frac{mcm(B,D)/B * A \pm mcm(B,D)/D * C}{mcm(B,D)}$$

Recordar que el Mínimo Común Múltiplo es la cifra más pequeña que es múltiplo de dos o más números y tiene relación con el Máximo Común Divisor (MCD) a través de la siguiente fórmula:

$$mcm(B,D) = \frac{B * D}{MCD(B,D)}$$

Por otro lado, el Máximo Común Divisor es el mayor número por el cual se pueden dividir dos o más números, sin dejar residuo y tiene las siguientes propiedades:

- MCD(B,D) = MCD(D,B)
- MCD(D,B) = MCD(B,D-B), si D $\geq$ B
- MCD(0,D) = D

Tomando en cuenta lo descrito anteriormente, se le pide elaborar un algoritmo en pseudocódigo que permita realizar operaciones como suma y resta entre diferentes pares de fracciones. El programa terminará cuando se ingrese 0 como numerador o denominador de alguna de las fracciones. También deberá validar la operación ingresada: + para sumar y – para restar, en caso se ingrese otro valor, deberá enviar el mensaje "Solo se trabajará con sumas y restas". Para cada par de fracciones, se deberá calcular y mostrar el resultado de la operación indicando si se trata de una fracción propia o impropia, en el caso de ser impropia deberá mostrar el número mixto formado. Considerar que el resultado de la resta sea positivo.

Solo para el caso del resultado obtenido en la operación de la última fracción válida, se deberá mostrar la fracción irreductible, es decir, ya simplificada.

Recordar que:

- Fracción propia: Si el numerador es menor que el denominador
- Fracción impropia: Si el numerador es mayor que el denominador

Casos de prueba:

```
Ingrese el numerador y denominador de la primera fracción:
> 12
> 13
Ingrese el numerador y denominador de la segunda fracción:
> 5
> 20
```

```
Ingrese la operación a realizar: (+) suma o (-) resta:
> -
Fracción propia. El resultado es: 12/13 - 5/20 = 175/260
Ingrese el numerador y denominador de la primera fracción:
> 3
> 5
Ingrese el numerador y denominador de la segunda fracción:
> 12
> 5
Ingrese la operación a realizar: (+) suma o (-) resta:
> +
Fracción impropia. El resultado es: 3/5 + 12/5 = 3
Ingrese el numerador y denominador de la primera fracción:
> 0
> 4
Ingrese el numerador y denominador de la segunda fracción:
> 6
> 7
La fracción irreductible del último resultado válido es:  3/1
```

```
Ingrese el numerador y denominador de la primera fracción:
> 16
> 17
Ingrese el numerador y denominador de la segunda fracción:
> 8
> 0
```

```
Ingrese el numerador y denominador de la primera fracción:
> 45
> 53
Ingrese el numerador y denominador de la segunda fracción:
> 3
> 16
Ingrese la operación a realizar: (+) suma o (-) resta:
> /
Solo se trabajará con sumas y restas.
Ingrese el numerador y denominador de la primera fracción:
> 11
> 16
Ingrese el numerador y denominador de la segunda fracción:
> 23
> 27
Ingrese la operación a realizar: (+) suma o (-) resta:
> +
Fracción impropia. El resultado es: 11/16 + 23/27 = 1 233/432
Ingrese el numerador y denominador de la primera fracción:
> 3
> 5
Ingrese el numerador y denominador de la segunda fracción:
> 8
> 0
La fracción irreductible del último resultado válido es:  665/432
```

```
Ingrese el numerador y denominador de la primera fracción:
> 12
> 15
Ingrese el numerador y denominador de la segunda fracción:
> 1
> 3
Ingrese la operación a realizar: (+) suma o (-) resta:
> -
Fracción propia. El resultado es: 12/15 - 1/3 = 7/15
Ingrese el numerador y denominador de la primera fracción:
```

```
> 21
> 8
Ingrese el numerador y denominador de la segunda fracción:
> 9
> 8
Ingrese la operación a realizar: (+) suma o (-) resta:
> +
Fracción impropia. El resultado es: 21/8 + 9/8 = 3 6/8
Ingrese el numerador y denominador de la primera fracción:
> 0
> 3
Ingrese el numerador y denominador de la segunda fracción:
> 5
> 6
La fracción irreductible del último resultado válido es:  15/4
```

```
Ingrese el numerador y denominador de la primera fracción:
> 200
> 420
Ingrese el numerador y denominador de la segunda fracción:
> 10
> 420
Ingrese la operación a realizar: (+) suma o (-) resta:
> +
Fracción propia. El resultado es: 200/420 + 10/420 = 210/420
Ingrese el numerador y denominador de la primera fracción:
> 4
> 5
Ingrese el numerador y denominador de la segunda fracción:
> 6
> 0
La fracción irreductible del último resultado válido es:  1/2
```

```
Ingrese el numerador y denominador de la primera fracción:
> 21
> 52
Ingrese el numerador y denominador de la segunda fracción:
> 1
> 4
Ingrese la operación a realizar: (+) suma o (-) resta:
> -
Fracción propia. El resultado es: 21/52 - 1/4 = 8/52
Ingrese el numerador y denominador de la primera fracción:
> 5
> 63
Ingrese el numerador y denominador de la segunda fracción:
> 8
> 0
La fracción irreductible del último resultado válido es:  2/13
```

Programa 1: Propuesta de solución

```
1  Algoritmo Pregunta1Final
2      Repetir
3          Escribir 'Ingrese el numerador y denominador de la primera fracción: '
4          Leer num1,den1
5          Escribir 'Ingrese el numerador y denominador de la segunda fracción: '
6          Leer num2,den2
7          originalDen1 <- den1
8          originalDen2 <- den2
9          Si (num1>0 y den1>0 y num2>0 y den2>0) Entonces
10             Escribir 'Ingrese la operación a realizar: (+) suma o (-) resta: '
11             Leer operacion
```

```
12          Si (operacion='+' o operacion='-') Entonces  //MCM de denominadores
13              finIteracion <- Falso
14              Mientras (no finIteracion) Hacer
15                  Si (den2<den1) Entonces
16                      auxiliar <- den1
17                      den1 <- den2
18                      den2 <- auxiliar
19                  FinSi
20                  Si (den1<1) Entonces
21                      finIteracion <- Verdadero
22                  SiNo
23                      den2 <- den2-den1
24                  FinSi
25              FinMientras
26              mcm <- originalDen1*originalDen2/den2
27              Si (operacion='+') Entonces
28                  resultadoNumerador <- (mcm/originalDen1)*num1 + (mcm/originalDen2)*num2
29              Sino
30                  Si (operacion='-') Entonces
31                      resultadoNumerador <- abs((mcm/originalDen1)*num1 - (mcm/originalDen2)*num2)
32                  Fin Si
33              FinSi
34              si (resultadoNumerador>mcm) Entonces
35                  parteEntera <- trunc(resultadoNumerador/mcm)
36                  residuo <- resultadoNumerador mod mcm
37                  Si (residuo=0) Entonces
38                      Escribir 'Fracción impropia. El resultado es: ',num1,'/',originalDen1,' ',operacion,' ',num2,'/',originalDen2,' = ',
                            parteEntera
39                  SiNo
40                      Escribir 'Fracción impropia. El resultado es: ',num1,'/',originalDen1,' ',operacion,' ',num2,'/',originalDen2,' = ',
                            parteEntera,' ',residuo,'/',mcm
41                  FinSi
42              SiNo
43                  Escribir 'Fracción propia. El resultado es: ',num1,'/',originalDen1,' ',operacion,' ',num2,'/',originalDen2,' = ',
                        resultadoNumerador,'/',mcm
44              FinSi
45          SiNo
46              Escribir 'Solo se trabajará con sumas y restas.'
47          Fin Si
48      FinSi
49  Mientras Que (num1>0 y originalDen1>0 y num2>0 y originalDen2>0) //se simplificará la fracción del resultado
50  i <- 2
51  num <- resultadoNumerador
52  den <- mcm
53  Si (operacion='+' o operacion='-') Entonces
54      menorNum <- 2
55  FinSi
56  Mientras (i<=menorNum) Hacer
57      Si ((num mod i)=0 y (den mod i)=0) Entonces
58          num <- trunc(num/i)
59          den <- trunc(den/i)
60      Sino
61          i <- i+1
62      FinSi
63      Si (num<den) Entonces
64          menorNum <- num
65      SiNo
66          menorNum <- den
67      FinSi
68  FinMientras
69  Escribir 'La fracción irreductible del último resultado es: ',num,'/',den
70  FinAlgoritmo
```

Pregunta 1B (6 puntos) *[propuesta por Sergio Ponce]*

Un número dócil es aquel que es igual a la suma de dos números a y b; además, se debe tener en cuenta lo

siguiente:

- a debe ser mayor que b
- La suma de los dígitos de a debe ser igual a la suma de los dígitos de b

Por ejemplo, los siguientes números son dóciles:

- 11 = 10 + 1
- 13 = 11 + 2
- 15 = 12 + 3
- 17 = 13 + 4
- 21 = 15 + 6

También existen los números dóciles que son primos (divisibles únicamente por 1 y por el mismo número).

Por ejemplo, los siguientes números son dóciles y también son primos:

- 11 = 10 + 1
- 13 = 11 + 2
- 17 = 13 + 4
- 19 = 14 + 5
- 23 = 16 + 7

Se le pide elaborar un algoritmo expresado en pseudocódigo que solicite ingresar un número y que muestre la combinación de valores a + b que permita demostrar que el número es dócil y también es primo; Así mismo, el algoritmo debe solicitar ingresar la cantidad de combinaciones a + b que el usuario cree que se formarán, el algoritmo determinará si su suposición fue correcta. Finalmente debe imprimir la cantidad de dígitos pares e impares del número ingresado inicialmente.

En esta pregunta se deben mostrar mensajes específicos ante las siguientes situaciones:

- Al ingresar el número y la cantidad de combinaciones debe validar que sean mayores que 0; además, el número debe ser menor o igual que 1000 y la cantidad de combinaciones debe ser menor que el número. Si no se cumple lo solicitado se debe mostrar el siguiente mensaje “Los datos ingresados no son correctos” y el algoritmo debe terminar.
- Si el número no es dócil y primo, se debe mostrar el siguiente mensaje “El número num NO es dócil y primo”, donde num es el número ingresado inicialmente.
- Si el número es dócil y primo, se debe mostrar el siguiente mensaje “El número num es dócil y primo y se encontraron n combinaciones”, donde num es el número ingresado inicialmente y n es la cantidad de combinaciones encontradas.

A continuación se presentan algunos ejemplos de ejecución del algoritmo. Utilice los mensajes que se muestran en los casos de prueba para el desarrollo de su algoritmo.

```
Ingrese un número:
> -4
Ingrese la cantidad de combinaciones que piensa encontrar:
> 10
Los datos ingresados no son correctos
```

```
Ingrese un número:
> 567
Ingrese la cantidad de combinaciones que piensa encontrar:
> 700
Los datos ingresados no son correctos
```

```
Ingrese un número:
> 22
Ingrese la cantidad de combinaciones que piensa encontrar:
> 1
El número 22 NO es dócil y primo
```

```
Ingrese un número:
> 997
Ingrese la cantidad de combinaciones que piensa encontrar:
> 12
539 + 458 = 997
548 + 449 = 997
629 + 368 = 997
638 + 359 = 997
719 + 278 = 997
728 + 269 = 997
809 + 188 = 997
818 + 179 = 997
908 + 89 = 997
El número 997 es dócil y primo y se encontraron 9 combinaciones
No adivinaste la cantidad de combinaciones
La cantidad de dígitos pares del número ingresado es:0
La cantidad de dígitos impares del número ingresado es: 3
```

```
> 101
Ingrese la cantidad de combinaciones que piensa encontrar:
> 2
55 + 46 = 101
64 + 37 = 101
73 + 28 = 101
82 + 19 = 101
100 + 1 = 101
El número 101 es dócil y primo y se encontraron 5 combinaciones
No adivinaste la cantidad de combinaciones
La cantidad de dígitos pares del número ingresado es: 1
La cantidad de dígitos impares del número ingresado es: 2
```

```
Ingrese un número:
> 101
Ingrese la cantidad de combinaciones que piensa encontrar:
> 5
55 + 46 = 101
64 + 37 = 101
73 + 28 = 101
82 + 19 = 101
100 + 1 = 101
El número 101 es dócil y primo y se encontraron 5 combinaciones
```

```
Adivinaste la cantidad de combinaciones
La cantidad de dígitos pares del número ingresado es:1
La cantidad de dígitos impares del número ingresado es: 2
```

Programa 2: Propuesta de solución

```
Algoritmo numerosDociles
    Escribir "Ingrese un número: "
    Leer num
    Escribir "Ingrese la cantidad de combinaciones que piensa encontrar:"
    Leer combinaciones
    Si num>0 y num<=1000 y combinaciones<num y combinaciones>0 Entonces
        a<-1
        b<-0
        cantCombinaciones<-0
        contPares<-0
        contImpares<-0
        Mientras a<=num Hacer
            b<-0
            Mientras b<a Hacer
                Si a+b=num Entonces
                    auxB<-b
                    sumaDigB<-0
                    digB<-0
                    Mientras auxB>0 Hacer
                        digB<-auxB Mod 10
                        auxB<-trunc(auxB/10)
                        sumaDigB<-sumaDigB+digB
                    FinMientras
                    auxA<-a
                    sumaDigA<-0
                    digA<-0
                    Mientras auxA>0 Hacer
                        digA<-auxA Mod 10
                        auxA<-trunc(auxA/10)
                        sumaDigA<-sumaDigA+digA
                    FinMientras
                    Si sumaDigA=sumaDigB Entonces
                        cantDivisores<-0
                        Para j<-1 Hasta num Con Paso 1 Hacer
                            Si num mod j = 0 Entonces
                                cantDivisores<-cantDivisores+1
                            FinSi
                        Fin Para
                        Si cantDivisores=2 Entonces
                            //Es dócil y es primo
                            Escribir a," + ",b," = ", num
                            cantCombinaciones<-cantCombinaciones+1
                        FinSi
                    FinSi
                FinSi
                b<-b+1
            FinMientras
            a<-a+1
        Fin Mientras
        Si cantCombinaciones>0 Entonces
            Escribir "El número ", num," es dócil y primo y se encontraron ", cantCombinaciones," combinaciones"
            Si combinaciones=cantCombinaciones Entonces
                Escribir "Adivinaste la cantidad de combinaciones"
            SiNo
                Escribir "No adivinaste la cantidad de combinaciones"
            FinSi
            Mientras num>0 Hacer
                digito<-num mod 10
                num<-trunc(num/10)
                Si digito mod 2=0 Entonces
```

```
61                  contPares<-contPares+1
62              SiNo
63                  contImpares<-contImpares+1
64              FinSi
65          FinMientras
66          Escribir "La cantidad de dígitos pares del número ingresado es:", contPares
67          Escribir "La cantidad de dígitos impares del número ingresado es: ", contImpares
68      SiNo
69          Escribir "El número ", num," NO es dócil y primo"
70      FinSi
71  SiNo
72      Escribir "Los datos ingresados no son correctos"
73  FinSi
74  FinAlgoritmo
```

El archivo debe ser grabado con el nombre **E2_código_pregunta1.psc** (donde la palabra código debe ser reemplazada por el código del alumno, por ejemplo, **E2_20206666_pregunta1.psc**, sin espacios).

Pregunta 2 – Programa en C con iterativas anidadas y programación modular (7 puntos)

Pregunta 2A (7 puntos) *[propuesta por Jorge Berrocal]*

Para un sistema de dos masas (A y B) en reposo unidas a través de una cuerda y colocadas en un plano inclinado como se muestra en la figura. Se desea realizar diferentes pruebas, a fin de calcular su aceleración y si el sistema avanza o no (aceleración >0) en la dirección de un punto indicado como Meta.

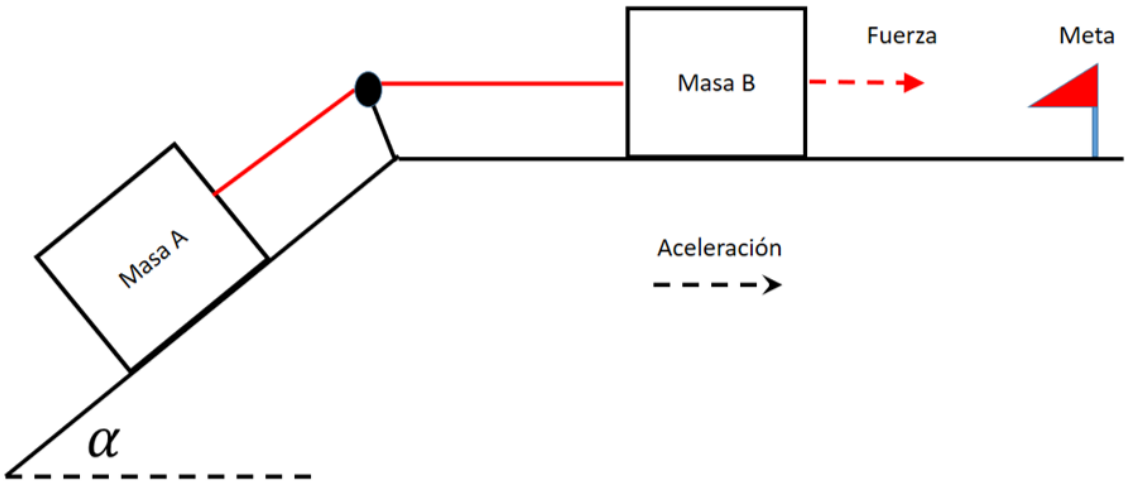

Figura 1: Ejemplo de un plano inclinado.

En cada prueba a realizar se deberá indicar los valores de cada una de las masas, así como del ángulo de plano inclinado. Con esos valores ya determinados, se deberá proceder a calcular la aceleración del sistema considerando diferentes variaciones del valor de la Fuerza. Sabiendo que el primer valor de la Fuerza deberá ser de 10 N (Newton: Kg x m / s2) mientras que los siguientes valores de la Fuerza a evaluar deberán ir aumentando de 10 N en 10 N cada uno. El número de variaciones de la Fuerza en todas las pruebas deberá ser el mismo y deberá ser ingresado.

Al finalizar cada prueba, después de haber evaluado las diferentes fuerzas deberá determinar si:

- El sistema no avanzó inicialmente, pero a partir de una fuerza de "X" Newton empezó a avanzar,
- El sistema no avanzó con ninguna de las fuerzas probadas o
- El sistema avanzó desde la primera fuerza probada.

Por último deberá consultar si se desea continuar con una nueva prueba o no.

Se pide elaborar un programa en Lenguaje C que en su módulo principal (main) solicite el ingreso de la cantidad de variaciones de la fuerza con las que se evaluarán todas las pruebas. En el main deberá validarse que el número de variaciones de la Fuerza sea mayor que cero (0). Si no lo fuera, deberá emitir un mensaje de error y finalizar el programa, mientras que si lo fuera se procederá a invocar un módulo que reciba dicho valor como parámetro.

El módulo invocado desde el main deberá realizar cada una de las pruebas del sistema, solicitando en cada una el ingreso de las masas de los dos bloques en kilogramos, así como el ángulo del plano inclinado en grados, verificando que las masas ingresadas sean mayores que cero (0) y el ángulo mayor a cero y menor a noventa. Si alguno de estos datos no fuera el correcto deberá emitir un mensaje de error y volver a solicitar el ingreso de los datos de la prueba, mientras que si los datos fueran correctos deberá calcular la aceleración del sistema considerando una fuerza inicial igual a 10 Newton (N), incrementando la fuerza en 10 N hasta completar todas las

variaciones indicadas de la fuerza. Para cada fuerza evaluada deberá imprimir la Fuerza aplicada en Newtons, la aceleración obtenida del sistema y si el sistema avanza o no (aceleración >0) en dirección de la meta.

Al finalizar cada prueba, el módulo deberá imprimir si el sistema no avanzó inicialmente, pero a partir de una fuerza de "X" Newton empezó a avanzar, si el sistema no avanzó con ninguna de las fuerzas probadas o si el sistema avanzó desde la primera fuerza probada.

Por último deberá consultar si desea realizar una siguiente prueba, para lo cual deberá solicitar el ingreso de un caracter. En caso se ingrese una S o s se volverá a realizar un nueva prueba. Caso contrario finalizará el programa.

Deberá considerar que la solución del programa cuente con al menos cuatro módulos incluido el módulo principal, de los cuales por lo menos dos deben devolver un valor.

Para resolver el problema podrá utilizar las siguientes consideraciones:

$$\boldsymbol{Aceleración} = \frac{\boldsymbol{Fuerza} - \boldsymbol{Masa\_A} * \boldsymbol{GRAVEDAD} * \sin(\boldsymbol{Ángulo\ en\ Radianes})}{(\boldsymbol{Masa\_A} + \boldsymbol{Masa\_B})}$$

$$\boldsymbol{Ángulo\ en\ Radianes} = \frac{\boldsymbol{PI} * \boldsymbol{Ángulo\ en\ Grados}}{180}$$

$$\boldsymbol{PI} = 3.14159 \quad y \quad \boldsymbol{GRAVEDAD} = 9.80\ \frac{m}{s^2}$$

Casos de prueba:

```
Ingrese el numero de variaciones de la fuerza para las pruebas a realizar: -5
El numero de variaciones de la fuerza debe ser mayor que cero.
```

```
Ingrese el numero de variaciones de la fuerza para las pruebas a realizar: 10

Ingrese la masa del cuerpo A en kg de la prueba 1: 5
Ingrese la masa del cuerpo B en kg de la prueba 1: -5
Ingrese el ángulo de la plataforma en grados de la prueba 1: 30
Las masas deben ser mayor a cero y el ángulo mayor a cero y menor a noventa.
Ingrese nuevamente la prueba 1.
Ingrese la masa del cuerpo A en kg de la prueba 1: 5
Ingrese la masa del cuerpo B en kg de la prueba 1: 5
Ingrese el ángulo de la plataforma en grados de la prueba 1: 0
Las masas deben ser mayor a cero y el ángulo mayor a cero y menor a noventa.
Ingrese nuevamente la prueba 1.
Ingrese la masa del cuerpo A en kg de la prueba 1: 5
Ingrese la masa del cuerpo B en kg de la prueba 1: 5
Ingrese el ángulo de la plataforma en grados de la prueba 1: 30

Los resultados de un sistema con masas de 5 kg y 5 kg y un ángulo de 30 grados son:
Con una Fuerza de  10.00 N se obtiene una aceleración de  -1.45 m/s2 el sistema no avanza.
Con una fuerza de  20.00 N se obtiene una aceleración de  -0.45 m/s2 el sistema no avanza.
Con una fuerza de  30.00 N se obtiene una aceleración de   0.55 m/s2 el sistema avanza.
Con una fuerza de  40.00 N se obtiene una aceleración de   1.55 m/s2 el sistema avanza.
Con una fuerza de  50.00 N se obtiene una aceleración de   2.55 m/s2 el sistema avanza.
Con una fuerza de  60.00 N se obtiene una aceleración de   3.55 m/s2 el sistema avanza.
Con una fuerza de  70.00 N se obtiene una aceleración de   4.55 m/s2 el sistema avanza.
Con una fuerza de  80.00 N se obtiene una aceleración de   5.55 m/s2 el sistema avanza.
Con una fuerza de  90.00 N se obtiene una aceleración de   6.55 m/s2 el sistema avanza.
Con una fuerza de 100.00 N se obtiene una aceleración de   7.55 m/s2 el sistema avanza.
El sistema no avanzó inicialmente, pero a partir de una fuerza de  30.00 N empezó a avanzar.
Desea realizar una nueva prueba: (S / N): n
```

```
Ingrese el numero de variaciones de la fuerza para las pruebas a realizar: 8
```

```
Ingrese la masa del cuerpo A en kg de la prueba 1: 1000
Ingrese la masa del cuerpo B en kg de la prueba 1: 500
Ingrese el ángulo de la plataforma en grados de la prueba 1: 25

Los resultados de un sistema con masas de 1000 kg y 500 kg y un ángulo de 25 grados son:
Con una fuerza de  10.00 N se obtiene una aceleración de  -2.75 m/s2, el sistema no avanza.
Con una fuerza de  20.00 N se obtiene una aceleración de  -2.75 m/s2, el sistema no avanza.
Con una fuerza de  30.00 N se obtiene una aceleración de  -2.74 m/s2, el sistema no avanza.
Con una fuerza de  40.00 N se obtiene una aceleración de  -2.73 m/s2, el sistema no avanza.
Con una fuerza de  50.00 N se obtiene una aceleración de  -2.73 m/s2, el sistema no avanza.
Con una fuerza de  60.00 N se obtiene una aceleración de  -2.72 m/s2, el sistema no avanza.
Con una fuerza de  70.00 N se obtiene una aceleración de  -2.71 m/s2, el sistema no avanza.
Con una fuerza de  80.00 N se obtiene una aceleración de  -2.71 m/s2, el sistema no avanza.
El sistema no avanzó con ninguna de las fuerzas probadas.
Desea realizar una nueva prueba: (S / N): S

Ingrese la masa del cuerpo A en kg de la prueba 2: 0.50
Ingrese la masa del cuerpo B en kg de la prueba 2: 0.25
Ingrese el ángulo de la plataforma en grados de la prueba 2: 80

Los resultados de un sistema con masas de 0.50 kg y 0.25 kg y un ángulo de 80.00 grados son:
Con una fuerza de  10.00 N se obtiene una aceleración de   6.90 m/s2, el sistema avanza.
Con una fuerza de  20.00 N se obtiene una aceleración de  20.23 m/s2, el sistema avanza.
Con una fuerza de  30.00 N se obtiene una aceleración de  33.57 m/s2, el sistema avanza.
Con una fuerza de  40.00 N se obtiene una aceleración de  46.90 m/s2, el sistema avanza.
Con una fuerza de  50.00 N se obtiene una aceleración de  60.23 m/s2, el sistema avanza.
Con una fuerza de  60.00 N se obtiene una aceleración de  73.57 m/s2, el sistema avanza.
Con una fuerza de  70.00 N se obtiene una aceleración de  86.90 m/s2, el sistema avanza.
Con una fuerza de  80.00 N se obtiene una aceleración de 100.23 m/s2, el sistema avanza.
El sistema avanzó desde la primera fuerza probada.
Desea realizar una nueva prueba: (S / N): N
```

Programa 3: Propuesta de solución

```c
#include <stdio.h>
#include <math.h>

#define PI 3.14159
#define GRAVEDAD 9.8 /*En metro / segundo al cuadrado.*/
#define TAMANO_VARIACION_FUERZA 10 /*En Kg x metro / segundo al cuadrado.*/

/*Declarar funciones del programa.*/
void ProcesarPlanoInclinado(int);
double ConvertirAnguloRadianes(double);
double CalcularAceleracion(double, double, double, double);

int main ()
{
	int Num_Variaciones_Fuerza;

	printf("Ingrese el numero de variaciones de la fuerza para las pruebas a realizar: ");
	scanf("%d", &Num_Variaciones_Fuerza);

	if ( Num_Variaciones_Fuerza>0 )
	{
		ProcesarPlanoInclinado(Num_Variaciones_Fuerza);
	}
	else
		printf("El numero de variaciones de la fuerza debe ser mayor que cero.");

	return 0;
}

void ProcesarPlanoInclinado(int Num_Variaciones_Fuerza)
{
```

```
32      int Cont_Pruebas = 1, Cont_Fuerzas, Datos_Ok, Avanzo_al_Inicio, Avanzo_Durante, Avanzo, Continua = 1;
33      double Masa_CuerpoA, Masa_CuerpoB, Angulo_Grad, Angulo_Rad, Fuerza, Fuerza_Avance, Aceleracion;
34      char MasPruebas;
35
36      /*Mientras se deseen realizar más pruebas.*/
37      do
38      {
39              printf("\nIngrese la masa del cuerpo A en kg de la prueba  %d: ", Cont_Pruebas);
40              scanf(" %lf", &Masa_CuerpoA);
41              printf("Ingrese la masa del cuerpo B en kg de la prueba  %d: ", Cont_Pruebas);
42              scanf(" %lf", &Masa_CuerpoB);
43              printf("Ingrese el ángulo de la plataforma en grados de la prueba  %d: ", Cont_Pruebas);
44              scanf(" %lf", &Angulo_Grad);
45
46              //Verifica que los datos ingresados sean los correctos.
47              Datos_Ok = ( Masa_CuerpoA > 0 ) && ( Masa_CuerpoB > 0 ) && ( Angulo_Grad > 0 ) && ( Angulo_Grad < 90 );
48              if  ( Datos_Ok )
49              {
50                      /*Inicializa la fuerza y el contador de fuerzas evaluadas.*/
51                      Fuerza = TAMANO_VARIACION_FUERZA;
52                      Cont_Fuerzas = 1;
53
54                      /*Convierte el ángulo a radianes.*/
55                      Angulo_Rad = ConvertirAnguloRadianes(Angulo_Grad);
56
57                      /*Obtiene la aceleración con la primera fuerza a evaluar para determinar si la primera fuerza es suficiente
                            para avanzar.*/
58                      Aceleracion = CalcularAceleracion(Fuerza, Masa_CuerpoA, Masa_CuerpoB, Angulo_Rad);
59
60                      /*En función de la aceleración para la primera fuerza determinar si avanza o no al inicio.*/
61                      Avanzo_al_Inicio = ( Aceleracion > 0 );
62                      Avanzo = Avanzo_al_Inicio;
63
64                      printf("\nLos resultados de un sistema con masas de  %.2lf kg y  %.2lf kg y un ángulo de  %.2lf grados son:\n",
65                                      Masa_CuerpoA, Masa_CuerpoB, Angulo_Grad);
66
67                      while ( Cont_Fuerzas <= Num_Variaciones_Fuerza )
68                      {
69                              /*Convierte el ángulo dado en grados en Radianes.*/
70                              Angulo_Rad = ConvertirAnguloRadianes(Angulo_Grad);
71
72                              /*Con la fuerza, las masas y el ángulo de la plataforma calcula la aceleración del sistema.*/
73                              Aceleracion = CalcularAceleracion(Fuerza, Masa_CuerpoA, Masa_CuerpoB, Angulo_Rad);
74
75                              /*Verifica si el sistema avanza con la fuerza dada.*/
76                              Avanzo_Durante = Aceleracion > 0;
77
78                              //Si no había avanzado y durante las variaciones de la fuerza avanza, actualiza Avanzo a Verdadero
                                    (1).
79                              if ( !Avanzo && Avanzo_Durante>0 )
80                              {
81                                      Avanzo = 1;
82                                      Fuerza_Avance = Fuerza;
83                              }
84
85                              /*Imprime el resultado de la evaluación del sistema para la fuerza dada de la prueba*/
86                              printf("Con una fuerza de  %6.2lf N se obtiene una aceleración de  %6.2lf m/s2, ", Fuerza,
                                    Aceleracion);
87                              if ( Avanzo )
88                                      printf("el sistema avanza.\n");
89                              else
90                                      printf("el sistema no avanza.\n");
91
92                              /*Incrementa la fuerza y el contador de fuerzas evaluadas.*/
93                              Fuerza += TAMANO_VARIACION_FUERZA;
94                              Cont_Fuerzas ++;
95                      }
```

```
96
97                          /*Imprime el resultado final de la prueba realizada.*/
98                          if ( !Avanzo )
99                                  printf("El sistema no avanzó con ninguna de las fuerzas probadas.\n");
100                         else if ( !Avanzo_al_Inicio )
101                                 printf("El sistema no avanzó inicialmente, pero a partir de una fuerza de  %6.2lf N empezo a avanzar.\
                                        n", Fuerza_Avance);
102                         else
103                                 printf("El sistema avanzó desde la primera fuerza probada.\n");
104
105                         /*Consulta si desea realizar más pruebas (MasPruebas).*/
106                         printf("Desea realizar una nueva prueba: (S / N): ");
107                         scanf("  %c", &MasPruebas);
108
109                         /*Si MasPruebas es 'S' o 's' incrementa en uno el contador de pruebas, caso contrario marca centinela (
                                Continua) en Falso (0).*/
110                         if ( MasPruebas=='S' || MasPruebas=='s' )
111                                 Cont_Pruebas ++;
112                         else
113                                 Continua = 0;
114                 }
115                 else
116                 {
117                         /*Imprime mensaje de error.*/
118                         printf("Las masas deben ser mayor a cero y el ángulo mayor a cero y menor a noventa.\n");
119                         printf("Ingrese nuevamente la prueba  %d.", Cont_Pruebas);
120                 }
121         } while ( Continua );
122 }
123
124 /*Calcula y retorna para un ángulo en grados dado su equivalente en radianes.*/
125 double ConvertirAnguloRadianes(double Angulo_Grad)
126 {
127         /*Calcula y retorna el ángulo en Radianes.*/
128         double Angulo_Rad = Angulo_Grad * PI / 180;
129         return Angulo_Rad;
130 }
131
132 /*Calcula y retorna la aceleración del sistema para una fuerza y masas dadas y con el ángulo de la plataforma en radianes.*/
133 double CalcularAceleracion(double Fuerza, double Masa_CuerpoA, double Masa_CuerpoB, double Angulo_Rad)
134 {
135         double Aceleracion = ( Fuerza – Masa_CuerpoA * GRAVEDAD * sin(Angulo_Rad) ) / ( Masa_CuerpoA + Masa_CuerpoB );
136         return Aceleracion;
137 }
```

Pregunta 2B (7 puntos) *[propuesta por Silvia Vargas]*

El enunciado más antiguo relacionado al teorema de los restos chinos se encuentra en el Manual de Matemática de Sun Zi [**?**], el cual menciona: Tenemos un número de cosas, pero no sabemos exactamente la cantidad. Si las contamos de a tres, quedan dos sobrando. Si las contamos de a cinco, quedan tres sobrando. Si las contamos de a siete, quedan dos sobrando. ¿Cuántas cosas pueden ser?

La solución planteada indica que el número 233 soluciona el problema, ya que: si se divide en grupos de 3 el resto es 2, si se divide entre 5 el resto es 3 y si se divide entre 7 el resto es 2. En operaciones matemáticas:

233 = 3 x 77 + 2

233 = 5 x 46 + 3

233 = 7 x 33 + 2

La cantidad de grupos que hacen posible estas igualdades son 77, 46 y 33. Es decir con 233 se forman 77 grupos de 3 y sobran 2 elementos; se forman 46 grupos de 5 y sobran 3 elementos y se forman 33 grupos de 7 y sobran 2 elementos.

Note que otra forma de escribir estas operaciones es:

233 - 2 = 3 x 77

233 - 3 = 5 x 46

233 - 2 = 7 x 33

Usando variables:

conjunto_de_elementos - resto_1 = tamaño_grupo_1 x cantidad_grupos_1

conjunto_de_elementos - resto_2 = tamaño_grupo_2 x cantidad_grupos_2

conjunto_de_elementos - resto_3 = tamaño_grupo_3 x cantidad_grupos_3

Además, los números 3, 5 y 7 (los tamaños de los grupos) son primos relativos 2 a 2; que significa que si se evalúan en parejas el único divisor común es el 1.

Pero 233 no es el único número que soluciona el problema, también lo hace el número 23:

23 - 2 = 3 x 7

23 - 3 = 5 x 4

23 - 2 = 7 x 3

Y así como 23 y 233. hay otros números que cumplen la igualdad anteriormente descrita.

Se pide desarrollar un programa en lenguaje C que dados 3 números que representan el tamaño de los grupos y sus respectivos restos, encuentre la cantidad de los grupos y el conjunto de elementos totales que se desean repartir. También se debe solicitar el máximo valor que puede tener el tamaño de un grupo.

El programa a desarrollar deberá estar definido con los siguientes módulos:

- Un módulo main que solicite el ingreso de los tamaños de los grupos a formar, si los valores que se ingresan son 0 0 0 se debe mostrar el mensaje ”Fin de la evalución” y terminar la ejecución del programa. Si todos los tamaños no tienen el valor de 0, debe solicitar el tamaño máximo del grupo y los restos respectivos para cada grupo. Con estos datos debe invocar a un módulo que realice la validación de los datos. Si los datos no son válidos se debe mostrar el mensaje ”Datos inválidos”. Caso contrario, debe invocar a un módulo que : muestre y retorne la cantidad de conjuntos de elementos totales que se pueden repartir, para cada conjunto de elementos calcule y muestre el tamaño del conjunto y calcule y muestre la cantidad de los grupos respectivos. En este módulo se debe mostrar la cantidad de conjuntos de elementos totales que cumplen el teorema de los restos chinos.
- Un módulo que reciba como parámetros todos los datos de entrada y devuelva si los datos son válidos o no. Debe validar que los tamaños de los grupos sean positivos y primos relativos 2 a 2. También que el tamaño máximo del grupo sea positivo y menor o igual a 500. Finalmente debe validar que cada resto sea menor al tamaño del grupo respectivo. Los grupos y los restos se ingresan en el mismo orden. Es decir el primer resto ingresado corresponde al tamaño del primer grupo y así sucesivamente.
- Un módulo que reciba dos números y determine si los números son primos relativos.
- Un módulo que reciba del tamaño de los grupos, los restos y el tamaño máximo del grupo, encuentre y muestre los conjuntos de elementos que cumplen la identidad, para cada conjunto encuentre y muestre la cantidad de los grupos. También debe contar la cantidad de conjuntos que cumplen las identidades y debe devolver este valor.

A continuación se presentan algunos ejemplos de ejecución del programa: utilice los mensajes que se muestran dentro del desarrollo delprograma.

```
Ingrese los tamaños de los grupos: 3 5 7
Ingrese el tamaño máximo del grupo: 100
Ingrese los restos para cada grupo: 2 3 2
Número encontrado 1: 23
23 = 2 + 3(7)
23 = 3 + 5(4)
23 = 2 + 7(3)
Número encontrado 2: 128
128 = 2 + 3(42)
128 = 3 + 5(25)
128 = 2 + 7(18)
Número encontrado 3: 233
233 = 2 + 3(77)
233 = 3 + 5(46)
233 = 2 + 7(33)
Se encontraron 3 conjuntos de números que cumplen el teorema de los restos chino
Ingrese los tamaños de los grupos: 2 7 9
Ingrese el tamaño máximo del grupo: 200
Ingrese los restos para cada grupo: 1 5 1
Número encontrado 1: 19
19 = 1 + 2(9)
19 = 5 + 7(2)
19 = 1 + 9(2)
Número encontrado 2: 145
145 = 1 + 2(72)
145 = 5 + 7(20)
145 = 1 + 9(16)
Número encontrado 3: 271
271 = 1 + 2(135)
271 = 5 + 7(38)
271 = 1 + 9(30)
Número encontrado 4: 397
397 = 1 + 2(198)
397 = 5 + 7(56)
397 = 1 + 9(44)
Se encontraron 4 conjuntos de números que cumplen el teorema de los restos chino
Ingrese los tamaños de los grupos: 0 0 0
Fin de la evaluación
```

```
Ingrese los tamaños de los grupos: 4 7 9
Ingrese el tamaño máximo del grupo: 150
Ingrese los restos para cada grupo: 2 5 6
Número encontrado 1: 222
222 = 2 + 4(55)
222 = 5 + 7(31)
222 = 6 + 9(24)
Número encontrado 2: 474
474 = 2 + 4(118)
474 = 5 + 7(67)
474 = 6 + 9(52)
Se encontraron 2 conjuntos de números que cumplen el teorema de los restos chino
Ingrese los tamaños de los grupos: 11 15 7
Ingrese el tamaño máximo del grupo: 200
Ingrese los restos para cada grupo: 4 10 5
Número encontrado 1: 565
565 = 4 + 11(51)
565 = 10 + 15(37)
565 = 5 + 7(80)
Se encontraron 1 conjuntos de números que cumplen el teorema de los restos chino
Ingrese los tamaños de los grupos: 0 0 0
Fin de la evaluación
```

```
Ingrese los tamaños de los grupos: 8 4 5
Ingrese el tamaño máximo del grupo: 130
```

```
Ingrese los restos para cada grupo: 2 1 1
Datos inválidos
Ingrese los tamaños de los grupos: 3 5 7
Ingrese el tamaño máximo del grupo: -3
Ingrese los restos para cada grupo: 2 1 1
Datos inválidos
Ingrese los tamaños de los grupos: 3 5 7
Ingrese el tamaño máximo del grupo: 50
Ingrese los restos para cada grupo: 5 3 1
Datos inválidos
Ingrese los tamaños de los grupos: 0 0 0
Fin de la evaluación
```

Programa 4: Propuesta de solución

```c
#include <stdio.h>
int validarNumeros(int ,int ,int ,int);
int verificarPrimosRelativos(int ,int );
int mostrarCalcularNumerosChinos(int ,int ,int ,int ,int ,int ,int);

int main(){
	int tamGrupo1,tamGrupo2,tamGrupo3,resto1,resto2,resto3,maximo,cantidad;
	do{
		printf("Ingrese los tamaños de los grupos: ");
		scanf(" %d %d %d",&tamGrupo1,&tamGrupo2,&tamGrupo3);
		if (tamGrupo1==0 && tamGrupo2==0 && tamGrupo3==0){
			printf("Fin de la evaluación\n");
			break;
		}
		printf("Ingrese el tamaño máximo del grupo: ");
		scanf(" %d",&maximo);
		printf("Ingrese los restos para cada grupo: ");
		scanf(" %d %d %d",&resto1,&resto2,&resto3);
		if (validarDatos(tamGrupo1,tamGrupo2,tamGrupo3,maximo,resto1,resto2,resto3)){
			cantidad=mostrarCalcularNumerosChinos(tamGrupo1,tamGrupo2,tamGrupo3,resto1,resto2,resto3,maximo);
			if (cantidad>0)
				printf("Se encontraron %d conjuntos de números que cumplen el teorema de los restos chino\n",
					cantidad);
			else
				if (cantidad==1)
					printf("Se encontró 1 conjunto de números que cumplen el teorema de los restos chino\n");
				else
					printf("No se encontró ningún conjunto de números que cumplen el teorema de los restos
						chino\n");
		}
		else
			printf("Datos inválidos\n");
	}while (1);
	return 0;
}

int validarDatos(int tamGrupo1,int tamGrupo2,int tamGrupo3,int maximo,int resto1,int resto2,int resto3){
	int valPositivos,valPrimosRel,valMax,valRestos;
	valPositivos=tamGrupo1>0 && tamGrupo2>0 && tamGrupo3>0;
	valPrimosRel=verificarPrimosRelativos(tamGrupo1,tamGrupo2) && verificarPrimosRelativos(tamGrupo2,tamGrupo3)
		&& verificarPrimosRelativos(tamGrupo3,tamGrupo1);
	valMax=maximo>0 && maximo<=500;
	valRestos=resto1<tamGrupo1 && resto2<tamGrupo2 && resto3<tamGrupo3;
	return valPositivos && valPrimosRel && valMax && valRestos;
}

int verificarPrimosRelativos(int num1,int num2){
	int menor,i;
	if (num1>num2)
		menor=num2;
	else
```

```
                menor=num1;
        for (i=2;i<=menor;i++){
                if (num1 %i==0 && num2 %i==0)
                        return 0;
        }
        return 1;
}

/* función con iterativas anidadas*/
int mostrarCalcularNumerosChinos(int tamGrupo1,int tamGrupo2,int tamGrupo3,int resto1,int resto2,int resto3,int maximo){
        int i,j,k,cantidad=0,cal1,cal2,cal3;
        for (i=1;i<=maximo;i++){
                for (j=1;j<=maximo;j++){
                        for (k=1;k<maximo;k++){
                                cal1=tamGrupo1*i+resto1;
                                cal2=tamGrupo2*j+resto2;
                                cal3=tamGrupo3*k+resto3;
                                if (cal1==cal2 && cal2==cal3){
                                        cantidad++;
                                        printf("Número encontrado  %d:  %d\n",cantidad,cal1);
                                        printf(" %d =  %d +  %d( %d)\n",cal1,resto1,tamGrupo1,i);
                                        printf(" %d =  %d +  %d( %d)\n",cal1,resto2,tamGrupo2,j);
                                        printf(" %d =  %d +  %d( %d)\n",cal1,resto3,tamGrupo3,k);
                                }
                        }
                }
        }
        return cantidad;
}

/*función con solo una iteración, otra forma de resolver lo solicitado*/

/*
int mostrarCalcularNumerosChinos(int tamGrupo1, int tamGrupo2, int tamGrupo3, int resto1, int resto2, int resto3, int maximo){
        int i=1, resto_cal1, resto_cal2, resto_cal3, coc_cal1, coc_cal2, coc_cal3, cantidad = 0;

        while (1){
                resto_cal1 = i  % tamGrupo1;
                resto_cal2 = i  % tamGrupo2;
                resto_cal3 = i  % tamGrupo3;
                coc_cal1 = i / tamGrupo1;
                coc_cal2 = i / tamGrupo2;
                coc_cal3 = i / tamGrupo3;
                if (coc_cal1>maximo || coc_cal2>maximo || coc_cal3>maximo)
                        break;
                if ( resto_cal1 == resto1 && resto_cal2 == resto2 && resto_cal3 == resto3 ){
                        cantidad++;
                        printf("Número encontrado  %d:  %d\n",cantidad, i);
                        printf(" %d =  %d +  %d(  %d)\n", i, resto1, tamGrupo1, coc_cal1);
                        printf(" %d =  %d +  %d(  %d)\n", i, resto2, tamGrupo2, coc_cal2);
                        printf(" %d =  %d +  %d(  %d)\n", i, resto3, tamGrupo3, coc_cal3);
                }
                i++;
        }
        return cantidad;
}
*/
```

El archivo debe ser grabado con el nombre **E2_código_pregunta2.c** (donde la palabra código debe ser reemplazada por el código del alumno, por ejemplo, **E2_20206666_pregunta2.c**, sin espacios).

Pregunta 3 – Programa en C con iterativas anidadas y programación modular (6 puntos)

Pregunta 3A (7 puntos) *[propuesta por Layla Hirsh]*

Los números de Motzkin tienen muchas interpretaciones combinatorias. En particular, M_n es el número total de formas en las que es posible dibujar acordes que no se crucen entre n puntos en un círculo.

Por ejemplo, para $n = 7$, es posible dibujar dichos acordes de 127 formas. En la imagen a un lado, muestran solo los 16 que se distinguen por rotación o reflexión.

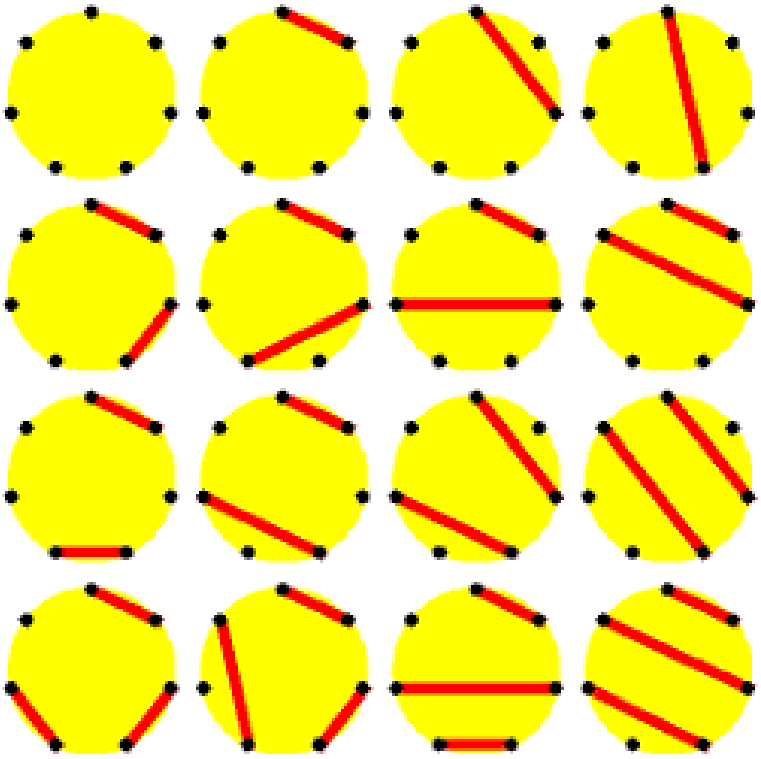

Figura 2: Números Motzkin

Los números de Motzkin se pueden calcular con la recurrencia: $M_n = \frac{3(n-1)M_{n-2}+(2n+1)M_{n-1}}{n+2}$

donde $M_0 = M_1 = 1$

Por otro lado, se puede calcular un número Motzkin con la siguiente sumatoria:

$$M_n = \sum_{k=0}^{n/2} \frac{1}{k+1} \binom{2k}{k} \binom{n}{2k}$$

donde:

$$\binom{a}{b} = \frac{a!}{b!(a-b)!}$$

Se le pide que haciendo uso de la sumatoria dada, calcule el número Motzkin solicitado por el usuario. Este número debe ser positivo mayor a 1. El usuario deberá poder hacer consultas hasta ingresar el valor 0. Considere los mensajes de error y de los términos mostrados en el ejemplo de prueba.

Para esto debe usar programación modular, al menos cuatro módulos incluído el main, donde debe hacer uso de iterativas anidadas, al menos uno de los módulos debe hacer uso de parámetros por referencia actualizando parámetros y al menos un módulo debe ser una función que calcule y devuelva un valor.

Uno de los módulos debe calcular y devolver el número de Motzkin, en este módulo debe usar una iterativa e imprimir el valor del término en las diversas iteraciones. Para este módulo debe desarrollar la tabla de datos para el número 8, si este módulo invoca a otros no se requiere hacer la tabla de datos de los módulos invocados.

```
Ingrese el número n para calcular Mn: 7
En la iteración 1 el término vale: 1
En la iteración 2 el término vale: 22
En la iteración 3 el término vale: 92
En la iteración 4 el término vale: 127
Para el valor 7 es posible dibujar dichos acordes de 127 formas
Ingrese el número n para calcular Mn: 9
En la iteración 1 el término vale: 1
En la iteración 2 el término vale: 37
En la iteración 3 el término vale: 289
En la iteración 4 el término vale: 709
En la iteración 5 el término vale: 835
Para el valor 9 es posible dibujar dichos acordes de 835 formas
Ingrese el número n para calcular Mn: 4
En la iteración 1 el término vale: 1
En la iteración 2 el término vale: 7
En la iteración 3 el término vale: 9
Para el valor 4 es posible dibujar dichos acordes de 9 formas
Ingrese el número n para calcular Mn: 0
```

```
Ingrese el número n para calcular Mn: -5
El valor ingresado no es mayor a 1
Ingrese el número n para calcular Mn: 5
En la iteración 1 el término vale: 1
En la iteración 2 el término vale: 11
En la iteración 3 el término vale: 21
Para el valor 5 es posible dibujar dichos acordes de 21 formas
Ingrese el número n para calcular Mn: 2
En la iteración 1 el término vale: 1
En la iteración 2 el término vale: 2
Para el valor 2 es posible dibujar dichos acordes de 2 formas
Ingrese el número n para calcular Mn: 0
```

Programa 5: Propuesta de solución

```c
#include<stdio.h>

int calcularNumeroMotzkinImprimirTerminos(int n);
int calcCombinatorio(int a,int b);
int calcfactorial(int n);
void calculayDevCombinatorios(int n,int k,int *combinatorio1,int *combinatorio2);

int main(){
    int suma,n;
    do{
        printf("\nIngrese el número n: ");
        scanf(" %d",&n);
        if(n>1){
            suma=calcularNumeroMotzkinImprimirTerminos(n);
            printf("Para el valor  %d es posible dibujar dichos acordes de  %d formas",n,suma);
        }
        else{
            if(n==0){
                break;
            }
            else{
                printf("El valor ingresado no es mayor a 1");
            }
```

```
            }
        }while(1);
        return 0;
}

int calcularNumeroMotzkinImprimirTerminos(int n){
            int centinela=0,term,suma=0,k=0,combinatorio1,combinatorio2;
            while(!centinela){
            calculayDevCombinatorios(n,k,&combinatorio1,&combinatorio2);
            term=((double)1/(k+1))*combinatorio1*combinatorio2;
                        suma=suma+term;
                        k++;
            printf("En la iteración  %d el termino vale:  %d\n",k,suma);
            if (k>(double)n/2)
                        centinela=1;
        }
        return suma;
}

void calculayDevCombinatorios(int n,int k,int *combinatorio1,int *combinatorio2){
        (*combinatorio1)=calcCombinatorio(2*k,k);
        (*combinatorio2)=calcCombinatorio(n,2*k);
}

int calcCombinatorio(int a,int b){
        int result;
        int numerador,denominador_b,denominador_resta;
        numerador=calcfactorial(a);
        denominador_b=calcfactorial(b);
        denominador_resta=calcfactorial(a-b);
        result=(double)numerador/(denominador_b*denominador_resta);
        return result;
}

int calcfactorial(int n){
        int i,prod=1;
        for(i=1;i<=n;i++){
            prod=prod*i;
        }
        return prod;
}

/*Tabla de datos calcularNumeroMotzkinImprimirTerminos(8)
iteración n centinela term suma k combinatorio1 combinatorio2
0  8  0  N.D  0  0  N.D   N.D
1  8  0  1  1  1  1   1
2  8  0  28  29  2  2   28
3  8  0  140  169  3  6   70
4  8  0  140  309  4  20   28
5  8  1  14  323  5  70   1
*/
```

Pregunta 3B (7 puntos) *[propuesta por David Allasi]*

Los **números de Munchausen** son aquellos números que son iguales a la suma de sus cifras elevadas a ellas mismas. Por ejemplo, el número 3435 es un número de Munchausen, ya que $3^3 + 4^4 + 3^3 + 5^5 = 3435$.

El nombre fue sugerido por el matemático e ingeniero de software holandés Daan van Berkel, en su artículo On a curious property of 3435 (2009), en referencia al barón de Munchausen. En los números de Munchausen cada cifra se eleva a sí misma, de la misma forma que el barón de Munchausen se eleva a sí mismo, tirando de su coleta, con lo cual consigue volar y evita caer en una ciénaga.

También se pueden definir los **números de Munchausen opuestos**, es decir, aquellos números que son iguales

a la suma de sus cifras elevadas a ellas mismas, pero no cada una con la suya, sino en el sentido opuesto. Por ejemplo, si consideramos el número 48625, sus cifras son 4, 8, 6, 2 y 5, y vamos a tomar sus potencias elevadas a las cifras, pero en el orden opuesto, 5, 2, 6, 8, 4 quedando $4^5 + 8^2 + 6^6 + 2^8 + 5^4 = 48625$, por lo que este número es de Munchausen opuesto.

Se le pide elaborar un programa en lenguaje C que permita evaluar si un número es de Munchausen o Munchausen opuesto. Para ello debe seleccionar en un menú de opciones cuál de las dos alternativas desea evaluar, opcion 1 para Munchausen u opcion 2 para Munchausen opuesto, en caso desee terminar las evaluaciones debe elegir la opción 3 de salir. Si la opción ingresada es 1 o 2 debe solicitar que ingrese un número y validar si el número cumple o no con la opción seleccionada. Al finalizar la evaluación debe imprimir el resultado de dicha evaluación.

En esta pregunta se deben mostrar mensajes específicos ante las siguientes situaciones:

- La opción ingresada debe ser 1, 2 o 3. Si la opción ingresada no es correcta debe mostrar el siguiente mensaje `La opción ingresada no es correcta` y debe volver a mostrar el menú de opciones para que vuelva a ingresar su opción.
- Debe validar que el número ingresado sea mayor que 0, si no cumple debe mostrar el siguiente mensaje `El número debe ser mayor que 0` y debe volver a mostrar el menú de opciones para que vuelva a ingresar su opción.

Para la implementación del programa debe desarrollar como mínimo 3 módulos adicionales al main. Además, uno de los módulos, que no sea el main, debe invocar a otro módulo.

Si requiere calcular la cantidad de dígitos de un número debe usar estructuras iterativas.

Se le pide también desarrollar la tabla de datos para el módulo donde realice la evaluación de si un número es Munchausen opuesto. Considere para la evaluación el número 48625. Si este módulo invoca a otros no se requiere hacer la tabla de datos de los módulos invocados.

A continuación se presenta varios ejemplos de ejecución:

```
Menú de opciones
1. Número de Munchausen
2. Número de Munchausen opuesto.
3. Salir
Ingrese su opción: 4
La opción ingresada no es correcta.
Menú de opciones
1. Número de Munchausen
2. Número de Munchausen opuesto.
3. Salir
Ingrese su opción: 1
Ingrese el número: -10
El número debe ser mayor que 0
Menú de opciones
1. Número de Munchausen
2. Número de Munchausen opuesto.
3. Salir
Ingrese su opción: 2
Ingrese el número: -10
El número debe ser mayor que 0
Menú de opciones
1. Número de Munchausen
2. Número de Munchausen opuesto.
3. Salir
Ingrese su opción: 1
Ingrese el número: 172
El número 172 no es Munchausen
Menú de opciones
1. Número de Munchausen
```

```
2. Número de Munchausen opuesto.
3. Salir
Ingrese su opción: 2
Ingrese el número: 48625
El número 48625 es Munchausen opuesto.
Menú de opciones
1. Número de Munchausen
2. Número de Munchausen opuesto.
3. Salir
Ingrese su opción: 3
```

```
Menú de opciones
1. Número de Munchausen
2. Número de Munchausen opuesto.
3. Salir
Ingrese su opción: 1
Ingrese el número: 3435
El número 3435 es Munchausen.
Menú de opciones
1. Número de Munchausen
2. Número de Munchausen opuesto.
3. Salir
Ingrese su opción: 2
Ingrese el número: 3435
El número 3435 no es Munchausen opuesto.
Menú de opciones
1. Número de Munchausen
2. Número de Munchausen opuesto.
3. Salir
Ingrese su opción: 3
```

Programa 6: Propuesta de solución

```c
#include <stdio.h>
#include <math.h>

int evaluarMunchausen(int numero);
int evaluarMunchausenOpuesto(int numero);
int calcularCantidadDigitos(int numero);

int main(){
	int opcion, numero, esMunchausen, esMunchausenOpuesto;
	while (1){
		printf("Menu de opciones: \n");
		printf("1. Número de Munchausen\n");
		printf("2. Número de Munchausen opuesto\n");
		printf("3. Salir\n");
		printf("Ingrese su opción: ");
		scanf(" %d",&opcion);
		if (opcion==1 || opcion==2){
			printf("Ingrese el número: ");
			scanf(" %d",&numero);
			if (numero>0){
				if (opcion==1){
					esMunchausen = evaluarMunchausen(numero);
					if (esMunchausen){
						printf("El número  %d es Munchausen\n",numero);
					}
					else {
						printf("El número  %d no es Munchausen\n",numero);
					}
				}
				else {
					if (opcion==2){
						esMunchausenOpuesto = evaluarMunchausenOpuesto(numero);
						if (esMunchausenOpuesto){
							printf("El número  %d es Munchausen opuesto\n",numero);
						}
						else {
```

```
                                                printf("El número  %d no es Munchausen opuesto\n",numero);
                                        }
                                }
                        }
                }
                else {
                        printf("El número debe ser mayor que 0\n");
                }
        }
        else {
                if (opcion==3){
                        break;
                }
                else {
                        printf("La opción ingresada no es correcta\n");
                }
        }
    }
    return 0;
}

int evaluarMunchausen(int numero){
    int suma = 0, copiaNumero=numero, digito;
    while (numero>0){
        digito = numero %10;
        numero = numero/10;
        suma = suma + (int)pow(digito,digito);
    }
    return (suma==copiaNumero);
}

int evaluarMunchausenOpuesto(int numero){
    int cantDigitos, suma = 0, num1=numero, num2=numero, digito1, digito2;
    cantDigitos = calcularCantidadDigitos(numero);
    while (num1>0){
        digito1 = num1 %10;
        digito2 = num2/((int)pow(10,cantDigitos-1));
        num1 = num1/10;
        num2 = num2 %((int)pow(10,cantDigitos-1));
        suma = suma + (int)pow(digito2,digito1);
        cantDigitos--;
    }
    return (suma==numero);
}

int calcularCantidadDigitos(int numero){
    int cantDigitos = 0;
    while (numero>0){
        numero = numero/10;
        cantDigitos++;
    }
    return cantDigitos;
}

/* Tabla de datos
    Nro.Iteracion cantDigitos  suma num1 numero num2 digito1  digito2
        0   5  0  48625 48625 48625 ND   ND
        1   4  1024 4862 48625 8625 5   4
        2   3  1088 486  48625 625  2   8
        3   2  8881 48  48625 25  6   6
        4   1  48000 4  48625 5  8   2
        5   0  48625 0  48625 0  4   5
*/
```

El archivo debe ser grabado con el nombre **E2_código_pregunta3.c** (donde la palabra código debe ser reemplazada por el código del alumno, por ejemplo, **E2_20206666_pregunta3.c**, sin espacios).