**Parcial I Programación I**

**Nombres: Daniel Esquivel Agudelo**

**I Parte**

1. (0.3)¿Cuál es la respuesta de la siguiente operación 44mod(8)?

a. 0

b. 2

c. 4

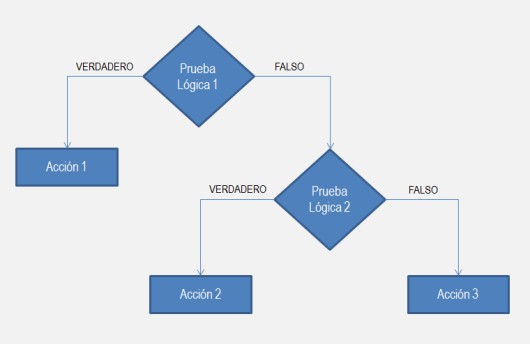
d. 8

1. (0.3) ¿Cuál es la solución de la siguiente operación teniendo en cuenta las prioridades de los signos? (3+15/5)x(24/6-3)
   1. 28.8
   2. 48
   3. 6
   4. 3.6
2. (0.3)Se desea construir un algoritmo que filtre personas que cumplan al menos una de las siguientes condiciones: sea mayor de 18 años, su estatura sea menor a 162 cm, el color de ojos debe ser diferente a azul y su cabello debe ser castaño. Para este ejercicio que operador lógico booleano utilizaría en su implementación:
   1. AND
   2. NOT
   3. OR
   4. NAND
3. (0.3) Para el ejercicio planteado en el punto 3. Qué resultado debería obtenerse si se tienen las entradas de la persona A y B.
4. Tiene 23 años, mide 160 cm, tiene ojos verdes y cabello castaño
5. Tiene 19 años, mide 164 cm, tiene ojos azules y cabello castaño.
   1. A Verdadero y B Verdadero
   2. A Falso y B Falso
   3. A Verdadero y B Falso
   4. A Falso y B Verdadero
6. (0.3) Sí por parte de negocio se les pide replantear las condiciones y se pide que en lugar de que se cumpla al menos una de las opciones, ahora sean obligatorias todas las condiciones, cuáles serían ahora los resultados de los ejemplos evaluados en el ejercicio 4.
   1. A Verdadero y B Verdadero
   2. A Falso y B Falso
   3. A Verdadero y B Falso
   4. A Falso y B Verdadero
7. (0.3) Si utilizamos el comando “git add .” qué sucederá:
   1. Todos los cambios realizados en el repositorio remoto se actualizarán en el repositorio local.
   2. Todo lo que tengamos en nuestra directorio de trabajo se moverá para el área temporal o staging área.
   3. Todos los documentos del Staging área o área temporal pasarán al repositorio local
   4. Se puede visualizar todos los hash asignados a cada versión de trabajo.
8. (0.3) Después de inicializar el repositorio, configurar usuario y agregar todos los archivos de un paquete en un espacio temporal, si utilizamos el comando “git commit -m “el exámen está fácil” ” qué sucederá:
   1. Todos los cambios realizados en el repositorio remoto se actualizarán en el repositorio local con el mensaje “el exámen está fácil” que lo distinguirá de las demás versiones.
   2. Todo lo que tengamos en nuestra directorio de trabajo se moverá para el área temporal o staging área con el mensaje “el exámen está fácil” que lo distinguirá de las demás versiones..
   3. Todos los documentos del Staging área o área temporal pasarán al repositorio local con el

mensaje “el exámen está fácil” que lo distinguirá de las demás versiones.

* 1. Se puede buscar la versión guardada con el mensaje “el exámen está fácil”

1. (0.3)La siguiente figura a qué estructura condicional pertenece:



* 1. Switch
  2. If Simple
  3. If anidada
  4. Ninguno de los anteriores

1. (0.3) Si usted define estas 2 variables var x=”2” y var y=”25” y desea encontrar el resultado de x+y, el resultado que encontrará será igual a:
   1. 252
   2. 50
   3. 27
   4. 225
2. (0.3)Según la notación del punto 9 las variables están definidas cómo qué tipos de datos:
   1. Numéricos
   2. Booleanos
   3. String

**II Parte**

Se desea desarrollar un programa que determine si un número es múltiplo de otro.

Dos números enteros positivos serán evaluados con base en si el primero es múltiplo del segundo, o viceversa, a través de una serie de condiciones. Un número es múltiplo de otro cuando al dividir el número mayor por el menor, el residuo es 0. Por ejemplo, 12 es múltiplo de 3, ya que 12 ÷ 3 = 4 sin residuo.

El programa deberá resolver este problema en tres fases:

Fase 1: Determinar si el primer número es múltiplo del segundo.

Fase 2: Si el primer número no es múltiplo del segundo, verificar si el segundo es múltiplo del primero.

Fase 3: Reportar el resultado final, indicando si alguno de los dos es múltiplo del otro o si no hay múltiplos entre ellos.

Para el problema, desarrolle lo siguiente:

1. (0.5) Indique las estructuras de control utilizadas (if, else, for, etc.) en cada fase, especificando cómo se emplean para cumplir las condiciones descritas.
2. (0.5) Escriba un algoritmo en un lenguaje de programación (puede ser en Javascript o cualquier otro) que implemente esta solución. Solo incluya el enlace al repositorio (GitHub) donde esté alojado.
3. (0.5) Identifique las posibles situaciones límite en las que el algoritmo puede fallar o arrojar errores (como divisiones por cero o posibles bucles infinitos) y extraiga fragmentos de código donde esto podría suceder.
4. (0.5) Determine cuáles de las siguientes parejas de números tienen una relación de múltiplo (indique si son o no múltiplos):
5. 7919 y 2
6. 840 y 210
7. 678.223.072.849 y 23
8. 1299827 y 104729
9. 104728 y 13