TC2025 / Programación avanzada

Examen 1

Tecnológico de Monterrey

Primer examen práctico de Programación Avanzada

ITESM - Campus Estado de México

Profesor: Miguel Angel Medina Pérez

Fecha: 17 de septiembre del 2019

Clave y grupo: TC2025-201913.1

Nombre: Jesús Omar Cuenca Espino Matrícula: A01378844

Compromiso de integridad académica

Apegándome al Reglamento de Integridad Académica del Instituto Tecnológico y de Estudios Superiores de Monterrey, me comprometo a que mi actuación en este examen esté regida por la Integridad Académica. En congruencia con el compromiso adquirido con dicho reglamento, realizaré este examen de forma honesta y personal, para reflejar, a través de él, mi conocimiento y aceptar, posteriormente, la evaluación obtenida.

Marque el recuadro para aceptar: ⬓

Instrucciones generales

Usted debe anexar este documento en la publicación del examen en Google Classroom titulada Examen 1 con los datos que se le solicitaron hasta este punto. Complete este formulario en formato pdf usando Acrobat Reader. También deberá anexar a dicha publicación los archivos individuales con sus códigos fuente. Si usted no entrega este documento con los datos solicitados, no entrega los códigos fuente, entrega o modifica alguno de estos documentos una vez transcurrida la hora final del examen, su solución será invalidada. Este examen debe ser elaborado de manera individual. Cualquier indicio de copia, trampa o fraude será sancionado de acuerdo al reglamento de Integridad Académica.

Ejercicios

1. **Codificando pares de bits (50%).**

Implemente una función de cantidad de parámetros variable (*variadic*) que recibe parámetros enteros sin signo de 16 bits y retorna un entero sin signo de 32 bits. La salida de la función contiene los dos bits menos significativos del primer parámetro variable, luego el par de bits de las posiciones 2 y 3 del siguiente parámetro variable y así sucesivamente. De esta manera, si la función recibiera los siguientes parámetros variables:

```
1. … 1100 0110

2. … 1010 0001
```

3. … 1101 0110

4. … 0111 1011

⋮

9. … 0110 1111

10. … 1110 1001

el resultado sería:

0… 1011 … 0101 0010

Nótese que la función solo procesará hasta 16 parámetros enteros sin signo de 16 bits. Todo parámetro adicional a los 16 enteros sin signo de 16 bits opcionales deberá ser ignorado.

Lista de cotejo

(50/12 % cada elemento)

Si su solución contiene arreglos, la calificación es de 0 puntos.

i. ☐ El estudiante crea un archivo de cabecera en C con el prototipo de la función y un archivo de código en C con la implementación de la función. La firma de la función en ambos archivos coincide y no tiene errores de sintaxis.
ii. ☐ El código fuente relativo a esta función compila sin errores con `GCC`.
iii. ☐ La firma de la función respeta las restricciones del problema: es una función de cantidad de parámetros variable que retorna un entero sin signo de 32 bits.
iv. ☐ El nombre de la función, los nombres de variables y parámetros de la función tienen sentido, excepto las variables de iteración de ciclos donde por convenio se aceptan nombres como `i`, `j` y `k`.
v. ☐ La función itera correctamente hasta por 16 de los parámetros opcionales enteros sin signo de 16 bits.
vi. ☐ Las pares de bits a concatenar son extraídos correctamente de los primeros 8 parámetros opcionales de la función.
vii. ☐ Las pares de bits a concatenar son extraídos correctamente de los primeros 16 parámetros opcionales de la función.
viii. ☐ Se concatenan correctamente los pares de bits obtenidos de los primeros 8 parámetros opcionales.
ix. ☐ Se concatenan correctamente los pares de bits obtenidos de los primeros 16 parámetros opcionales.
x. ☐ En caso de recibir más de 16 parámetros opcionales, la función solo itera por los primeros 16.
xi. ☐ La función retorna un entero sin signo de 32 bits que contiene los pares de bits concatenados.

xii. ☐ Es correcta la declaración del puntero para iterar por la lista de parámetros opcionales. Es correcta la inicialización del puntero para iterar por la lista de parámetros opcionales. Es correcta la lectura de cada parámetro opcional usando el puntero declarado para iterar por la lista de parámetros. Es correcta la liberación/terminación del puntero usado para iterar por la lista de parámetros opcionales.

2. Prueba de 0 parámetros opcionales (15%).

Implemente una función que prueba que la función del ejercicio 1 funciona correctamente al recibir 0 parámetros opcionales.
Tenga en cuenta que en nuestra clase en Google Classroom aparece un ejemplo de función de prueba.

Lista de cotejo

(15/7 % cada elemento)

Si su solución contiene arreglos, la calificación es de 0 puntos.

i. ☐ El estudiante incluye el prototipo de la función de prueba en el archivo de cabecera en C creado en el ejercicio 1. El estudiante incluye la implementación de la función de prueba en el archivo de código en C creado en el ejercicio anterior. La firma de la función de prueba en ambos archivos coincide y no tiene errores de sintaxis.
ii. ☐ El código fuente relativo a la función de prueba compila sin errores con `GCC`.
iii. ☐ La firma de la función de prueba respeta las restricciones del problema: es una función que no recibe parámetros y retorna un valor entero. La implementación de la función no imprime ningún mensaje en consola.
iv. ☐ El nombre de la función de prueba, los nombres de variables y parámetros de la función tienen sentido, excepto las variables de iteración de ciclos donde por convenio se aceptan nombres como `i`, `j` y `k`.
v. ☐ La función de prueba invoca correctamente a la función del ejercicio 1 pasando 0 parámetros opcionales.
vi. ☐ La función de prueba verifica que al invocar la función del ejercicio 1 se retorna la salida correcta de acuerdo con los parámetros descritos en el ejercicio.
vii. ☐ La función de prueba retorna 1 si la prueba pasa y cero en caso contrario.

3. Prueba de 3 parámetros opcionales (15%).

Implemente una función que prueba que la función del ejercicio anterior funciona correctamente al recibir 3 parámetros opcionales. Cada uno de los parámetros opcionales debe contener algún bit activo de los que se concatenarán.
Tenga en cuenta que en nuestra clase en Google Classroom aparece un ejemplo de función de prueba.

Lista de cotejo

(15/7 % cada elemento)

Si su solución contiene arreglos, la calificación es de 0 puntos.

i. ☐ El estudiante incluye el prototipo de la función de prueba en el archivo de cabecera en C creado en el ejercicio 1. El estudiante incluye la implementación de la función de prueba en el archivo de código en C creado en el ejercicio anterior. La firma de la función de prueba en ambos archivos coincide y no tiene errores de sintaxis.
ii. ☐ El código fuente relativo a la función de prueba compila sin errores con `GCC`.
iii. ☐ La firma de la función de prueba respeta las restricciones del problema: es una función que no recibe parámetros y retorna un valor entero. La implementación de la función no imprime ningún mensaje en consola.
iv. ☐ El nombre de la función de prueba, los nombres de variables y parámetros de la función tienen sentido, excepto las variables de iteración de ciclos donde por convenio se aceptan nombres como `i`, `j` y `k`.
v. ☐ La función de prueba invoca correctamente a la función del ejercicio 1 pasando 3 parámetros opcionales. Cada uno de los parámetros opcionales debe contener algún bit activo de los que se concatenarán.
vi. ☐ La función de prueba verifica que al invocar la función del ejercicio 1 se retorna la salida correcta de acuerdo con los parámetros descritos en el ejercicio.
vii. ☐ La función de prueba retorna 1 si la prueba pasa y cero en caso contrario.

4. Prueba de todos los bits activos (20%).

Implemente una función que prueba que la función del ejercicio 1 funciona correctamente al recibir 18 parámetros opcionales de los cuales los primeros 16 tienen todos los bits activos y los siguientes dos parámetros no tienen ningún bit activo.
Tenga en cuenta que en nuestra clase en Google Classroom aparece un ejemplo de función de prueba.

Lista de cotejo

(20/7 % cada elemento)

Si su solución contiene arreglos, la calificación es de 0 puntos.

i. ☐ El estudiante incluye el prototipo de la función de prueba en el archivo de cabecera en C creado en el ejercicio 1. El estudiante incluye la implementación de la función de prueba en el archivo de código en C creado en el ejercicio anterior. La firma de la función de prueba en ambos archivos coincide y no tiene errores de sintaxis.
ii. ☐ El código fuente relativo a la función de prueba compila sin errores con `GCC`.
iii. ☐ La firma de la función de prueba respeta las restricciones del problema: es una función que no recibe parámetros y retorna un valor entero. La implementación de la función no imprime ningún mensaje en consola.
iv. ☐ El nombre de la función de prueba, los nombres de variables y parámetros de la función tienen sentido, excepto las variables de iteración de ciclos donde por convenio se aceptan nombres como `i`, `j` y `k`.
v. ☐ La función de prueba invoca correctamente a la función del ejercicio 1 pasando 16 parámetros opcionales con todos sus bits activos y 2 parámetros opcionales más con ningún bit activo.

vi. ☐ La función de prueba verifica que al invocar la función del ejercicio 1 se retorna la salida correcta de acuerdo con los parámetros descritos en el ejercicio.

vii. ☐ La función de prueba retorna 1 si la prueba pasa y cero en caso contrario.

Tecnológico de Monterrey