Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3641 – Lenguajes de Programación I
Septiembre-Diciembre 2008

Carnet: _____

Nombre: _____

Quiz

(20 puntos)

Antes de empezar, revise bien el quiz.

Nota
20 puntos

Instrucciones:

A continuación se le presentan diez (10) preguntas de selección, numeradas de 0 a 9. Cada pregunta viene acompañada de cuatro posibles respuestas (*a,b,c,d*), entre las cuales <u>sólo una</u> es correcta. Ud. deberá marcar la opción que considere correcta o, si desea no contestar la pregunta, marcar la última opción (*e*) que indica que Ud. prefiere omitir la respuesta.

Cada una de las diez (10) preguntas tiene un valor de dos (2) puntos. Tres preguntas malas eliminan una buena. Las preguntas omitidas (marcadas en la opción (*e*)) no suman ni restan puntos.

---

0. Señale a través de cuáles de los siguientes mecanismos puede ser realizada la implementación de un lenguaje de programación:

   I. por pura traducción,

   II. por pura interpretación,

   III. por alguna combinación de traducción e interpretación.

   a) Sólo I y II.

   b) I, II y III.

   c) Sólo I y III.

   d) Sólo II y III.

   e) No sabe / No contesta.

1. Scott menciona en el libro de texto que parte del éxito del lenguaje Pascal se debió a que su diseñador Niklaus Wirth distribuía el siguiente conjunto de herramientas para promocionar su lenguaje:

   (A) un compilador de Pascal, escrito en Pascal, que generaba salida en un lenguaje sencillo intermedio llamado P-code;

   (B) el mismo compilador, ya traducido a P-code;

   (C) un interpretador de P-code, escrito en Pascal.

   La manera sencilla de instalar Pascal en una máquina cualquiera era entonces traduciendo manualmente (C) a un lenguaje previamente disponible, obteniéndose un interpretador (C'), y luego procesando cada programa en Pascal mediante:

   a) ejecución de (B) sobre (C), y luego ejecución de la salida sobre (C).

   b) ejecución de (B) sobre (C'), y luego ejecución de la salida sobre (C').

   c) ejecución de (A) sobre (C), y luego ejecución de la salida sobre (C).

   d) ejecución de (A) sobre (C'), y luego ejecución de la salida sobre (C').

   e) No sabe / No contesta.

2. Continuando con las herramientas de Pascal descritas en la pregunta anterior 1, una manera de obtener una implementación más eficiente de Pascal en la nueva máquina era modificando (A) para que generase salida en el lenguaje de máquina, obteniéndose un nuevo compilador (A'), y luego obtener un compilador independiente de P-code mediante:

   a) ejecución de (A') sobre (C) para compilar a (B), y luego ejecución de la salida sobre (C) para compilar a (A).

   b) ejecución de (A) sobre (C) para compilar a (B), y luego ejecución de la salida sobre (C) para compilar a (A').

   c) ejecución de (B) sobre (C') para compilar a (A), y luego ejecución de la salida sobre (C') para compilar a (A').

d) ejecución de (B) sobre (C') para compilar a (A'), y luego ejecución de la salida sobre (C') para compilar a (A').

e) No sabe / No contesta.

3. Entre los siguientes lenguajes:

   I. Lisp,

   II. Fortran,

   III. Java,

   IV. Haskell;

   ¿cuáles pertenecen a la familia de lenguajes funcionales (puros o impuros)?

   a) Sólo I y IV.

   b) Sólo IV.

   c) Sólo II y III.

   d) Sólo III.

   e) No sabe / No contesta.

4. Se acostumbra hacer reserva estática de espacio de memoria:

   a) para variables locales de subrutinas recursivas.

   b) para variables de tamaño conocido estáticamente con cualquier tipo de tiempo de vida.

   c) para variables de tamaño conocido estáticamente cuyo tiempo de vida sea toda la ejecución del programa.

   d) para objetos de tamaño variable con cualquier tipo de tiempo de vida.

   e) No sabe / No contesta.

5. La definición de algunos lenguajes, como C, C++ y Pascal, señala que la liberación de memoria en *heap* debe ser realizada explícitamente por el programador. En la definición de otros, como Java, C# y Modula-3, se señala que tal liberación debe ser realizada implícitamente por la implementación del lenguaje mediante un recolector de basura.

   En un lenguaje del primer grupo, en los que la liberación es explícita, esta decisión de diseño favorece:

   a) la portabilidad de los programas escritos en el lenguaje.

   b) la eficiencia de construcción y mantenimiento de los programas escritos en el lenguaje.

   c) la eficiencia de ejecución de los programas escritos en el lenguaje.

   d) la eficiencia del procesamiento de llamadas a subrutinas escritas en el lenguaje.

   e) No sabe / No contesta.

6. Continuando con la pregunta anterior 5, en un lenguaje del segundo grupo, en los que la liberación es implícita, esta decisión de diseño favorece:

   a) la portabilidad de los programas escritos en el lenguaje.

   b) la eficiencia de construcción y mantenimiento de los programas escritos en el lenguaje.

   c) la eficiencia de ejecución de los programas escritos en el lenguaje.

   d) la eficiencia del procesamiento de llamadas a subrutinas escritas en el lenguaje.

   e) No sabe / No contesta.

7. Considere las siguientes afirmaciones:

   En una clase de Java, las reglas de alcance/visibilidad asociadas a un atributo privado no-estático permiten

   I. que todo objeto de la clase manipule ese atributo tanto de sí mismo como de otros objetos de la clase;

   II. que todo objeto de la clase manipule ese atributo sólo de sí mismo pero no de otros objetos de la clase;

III. que este atributo sea manipulado en todos los métodos, estáticos o no, de la clase.

¿Cuáles de ellas son ciertas?

a) Sólo I.

b) Sólo II.

c) Sólo III.

d) Sólo I y III.

e) No sabe / No contesta.

8. En Modula-2, lenguaje con reglas de alcance estático y que permite anidamiento de subrutinas y pasaje de subrutinas como parámetro, sólo se permite que se envíen como parámetros reales aquellas subrutinas declaradas en el nivel principal. Esto se hace con el objetivo de:

   a) asegurar que en toda ejecución exista una sola instancia del contexto no-local de la subrutina enviada como parámetro.

   b) que la posibilidad de *deep binding* del contexto no-local sea más eficiente que la posibilidad de *shallow binding* del mismo.

   c) limitar de manera arbitraria la cantidad de subrutinas que pueden ser enviadas como parámetro.

   d) evitar que las subrutinas se conviertan en regiones cerradas en las que sólo pueda referenciarse su contexto local.

   e) No sabe / No contesta.

9. En lenguajes como las versiones de Fortran pre-90, en los que no se permite que las subrutinas sean recursivas, ni directa ni indirectamente, la reserva de espacio para las variables locales de una subrutina puede ser realizada:

   a) sólo estáticamente.

   b) sólo en pila.

   c) sólo en *heap*.

   d) estáticamente o en pila.

   e) No sabe / No contesta.

———————————— ○ ————————————