Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
CI3641 – Lenguajes de Programación I
Septiembre–Diciembre 2024

Examen 1

(20 puntos)

A continuación encontrará 4 preguntas (y una sorpresa al final), cada una de las cuales tiene un valor de 5 puntos. Sea lo más detallado y preciso posible en sus razonamientos y procedimientos.

En algunas preguntas, se usarán las constantes X, Y y Z. Estas constantes debe obtenerlas de los últimos tres números de su carné. Por ejemplo, si su carné es 09-40325, entonces $X = 3$, $Y = 2$ y $Z = 5$.

En aquellas preguntas donde se le pida decir qué imprime un programa, incluya los pasos relevantes de la ejecución del mismo con los cuales usted pudo alcanzar su conclusión.

En aquellas preguntas donde se le pida implementar un programa, mantenga su código en un repositorio `git` remoto (preferiblemente `Github`) y coloque un enlace al mismo en lugar de su respuesta. Todo su código debe ser legible y estar debidamente documentado.

La entrega se realizará por correo electrónico a rmonascal@gmail.com hasta las 11:59pm. VET del Lunes 14 de Octubre de 2024.

1. Escoja algún lenguaje de programación de alto nivel y de propósito general cuyo nombre empiece con la misma letra que su nombre (por ejemplo, si su nombre es "Pedro", podría escoger "Perl", "PLI", "Python", etc.).

   (a) De una breve descripción del lenguaje escogido.

      i. Diga qué tipo de alcances y asociaciones posee, argumentando las ventajas y desventajas de la decisión tomada por los diseñadores del lenguaje, en el contexto de sus usuarios objetivos.

      ii. Diga qué tipo de módulos ofrece (de tenerlos) y las diferentes formas de importar y exportar nombres.

      iii. Diga si el lenguaje ofrece la posibilidad de crear aliases, sobrecarga y polimorfismo. En caso afirmativo, dé algunos ejemplos.

      iv. Diga qué herramientas ofrece a potenciales desarrolladores, como: compiladores, intérpretes, debuggers, profilers, frameworks, etc.

   (b) Implemente los siguientes programas en el lenguaje escogido:

      i. Dados tres enteros no–negativos a, b y c , tal que $c \geq 2$, calcular la potenciación modulada $a^b \mod c$. Utilice la siguiente fórmula, resultante de aplicar propiedades de módulo a la potenciación:

$$a^b \mod c = \left\{ \begin{array}{ll} 1 & \text{si } b = 0 \\ ((a \mod c) \times (a^{b-1} \mod c)) \mod c & \text{si } b > 0 \end{array} \right\}$$

      ii. Dada una matriz cuadrada A (cuya dimensión es $N \times N$), decidir si A corresponde a una **Matriz Mágica**.

      Decimos que una matriz es mágica si la suma de cada una de sus filas, columnas y las dos diagonales corresponden al mismo número. Por ejemplo, consideremos la siguiente matriz:

8	1	6
3	5	7
4	9	2

      Esta matriz es mágica ya que cualquier fila, columna o diagonal, si se suman sus elementos, el resultado es 15.

2. Considere el siguiente programa escrito en pseudo-código:

```
int a = X + 1, b = Y;

proc goma(int a) {
   b := Z * a;
}

proc pistol(int c, proc lu, proc ffy) {
   if (c < 2 * (X + 1)) {
      proc goma(int b) {
         a := b * c;
      }
      pistol(c + 2 * (X + 1), ffy, lu);
   } else if (c < 4 * (X + 1)) {
      int a = b + c;
      pistol(c + 2 * (X + 1), goma, ffy);
   } else {
      int c = Z;
      lu(a + b);
      ffy(b + c);
   }
   print(a, b)
}

pistol(a, goma, goma);
print(a, b)
```

Note que deberá reemplazar los valores para X, Y y Z como fue explicado en los párrafos de introducción del examen.

Diga qué imprime el programa en cuestión, si el lenguaje tiene:

(a) Alcance estático y asociación profunda

(b) Alcance dinámico y asociación profunda

(c) Alcance estático y asociación superficial

(d) Alcance dinámico y asociación superficial

Recuerde mostrar los pasos de su ejecución (por lo menos al nivel de cada nuevo marco de pila creado).

3. Se desea que modele e implemente, en el lenguaje de su elección, un programa que simule programas, intérpretes y traductores com los vistos al estudiar los *diagramas de T*. Este programa debe cumplir con las siguientes características:

   (a) Debe poder manejar programas, intérpretes y traductores. Estos pueden ser:

   - `PROGRAMA <nombre> <lenguaje>`
     Representa a un programa identificado por `<nombre>` escrito en `<lenguaje>`.
   - `INTERPRETE <lenguaje_base> <lenguaje>`
     Representa a un intérprete para `<lenguaje>` escrito en `<lenguaje_base>`.
   - `TRADUCTOR <lenguaje_base> <lenguaje_origen> <lenguaje_destino>`
     Representa a un traductor, desde `<lenguaje_origen>` hacia `<lenguaje_destino>`, escrito en `<lenguaje_base>`.

   (b) Todos los lenguajes deben ser cadenas alfanuméricas y no tienen por qué corresponder a algún lenguaje que exista.

   (c) Existirá una nombre especial `LOCAL` que se referirá al lenguaje que puede interpretar la máquina local.

   (d) Una vez iniciado el programa, pedirá repetidamente al usuario una acción para proceder. Tal acción puede ser:

   i. `DEFINIR <tipo> [<argumentos>]`
      Representa una definición de clase `<tipo>` con `<argumentos>` (ver arriba los tipos y argumentos que se deben soportar).
      El programa debe reportar un error e ignorar la acción si `<nombre>` ya tiene un programa asociado, en el caso de programas.

   ii. `EJECUTABLE <nombre>`
      Representa una consulta de la posibilidad de ejecutar el programa de nombre *<nombre>*.
      Su programa debe imprimir si es posible construir lo que se pide, usando únicamente las definiciones hechas hasta el momento.
      El programa debe reportar un error e ignorar la acción si `<nombre>` no tiene un programa asociado.

   iii. `SALIR`
      Debe salir del simulador.

   Al finalizar la ejecución de cada acción, el programa deberá pedir la siguiente acción al usuario.

Consideremos un ejemplo un poco más elaborado para comprender el funcionamiento del programa:

```
$> DEFINIR PROGRAMA fibonacci LOCAL
 Se definió el programa 'fibonacci', ejecutable en 'LOCAL'
$> EJECUTABLE fibonacci
 Si, es posible ejecutar el programa 'fibonacci'
$> DEFINIR PROGRAMA factorial Java
 Se definió el programa 'fibonacci', ejecutable en 'Java'
$> EJECUTABLE factorial
 No es posible ejecutar el programa 'factorial'
$> DEFINIR INTERPRETE C Java
 Se definió un intérprete para 'Java', escrito en 'C'
$> DEFINIR TRADUCTOR C Java C
 Se definió un traductor de 'Java' hacia 'C', escrito en 'C'
$> EJECUTABLE factorial
 No es posible ejecutar el programa 'factorial'
$> DEFINIR INTERPRETE LOCAL C
 Se definió un intérprete para 'C', escrito en 'LOCAL'
$> EJECUTABLE factorial
 Si, es posible ejecutar el programa 'factorial'
$> DEFINIR PROGRAMA holamundo Python3
 Se definió el programa 'holamundo', ejecutable en 'Python3'
$> DEFINIR TRADUCTOR wtf42 Python3 LOCAL
 Se definió un traductor de 'Python3' hacia 'LOCAL', escrito en 'wtf42'
$> EJECUTABLE holamundo
 No es posible ejecutar el programa 'holamundo'
$> DEFINIR TRADUCTOR C wtf42 Java
 Se definió un traductor de 'wtf42' hacia 'Java', escrito en 'C'
$> EJECUTABLE holamundo
 Si, es posible ejecutar el programa 'holamundo'
```

Investigue herramientas para pruebas unitarias y cobertura en su lenguaje escogido y agregue pruebas a su programa que permitan corroborar su correcto funcionamiento. Como regla general, su programa debería tener una cobertura (de líneas de código y de bifuración) mayor al 80%.

4. Se desea que modele e implemente, en el lenguaje de su elección, un módulo que defina el tipo de los cuaterniones y operadores aritméticas sobre estos.

   Dados en tres valores: i, j y k, tales que $i^2 = j^2 = k^2 = ijk = -1$, la forma general de cuaternión se define como: $a + bi + cj + dk$, para $a, b, c, d \in \mathbb{R}$

   La librería debe cumplir con las siguientes características:

   (a) Debe saber tratar las siguientes expresiones aritméticas:

   - **suma**: operador biádico, representado por el símbolo `+`.

     $$(a_1+b_1i+c_1j+d_1k)+(a_2+b_2i+c_2j+d_2k) = (a_1+a_2)+(b_1+b_2)i+(c_1+c_2)j+(d_1+d_2)k$$

   - **conjugada**: operador monádico prefijo, representado por el símbolo $\sim$.

     $$\sim (a + bi + cj + dk) = a - bi - cj - dk$$

   - **producto**: operador biádico, representado por el símbolo `*`.

     $$\begin{aligned}(a_1 + b_1i + c_1j + d_1k) * (a_2 + b_2i + c_2j + d_2k) = {} & a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2 \\ & + (a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2)i \\ & + (a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2)j \\ & + (a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2)k\end{aligned}$$

   - **medida o valor absoluto**: operador monádico prefijo, representado por el símbolo `&`.

     $$\&(a + bi + cj + dk) = \sqrt{a^2 + b^2 + c^2 + d^2}$$

   (b) Los cuaterniones deben poder operarse entre sí sin necesidad de que el usuario de la librería escriba llamadas adicionales a funciones. Por ejemplo, si `a`, `b` y `c` son cuaterniones, las siguientes expresiones deben ser válidas para un usuario de la librería:

   ```
   b + c
   a * b + c
   (b + b) * (c + ~a)
   &(c * b)
   ```

   (c) Los cuaterniones deben poder operarse también con números enteros o flotantes por la derecha (para suma y producto) sin necesidad de que el usuario de la librería escriba llamadas adicionales a funciones. Por ejemplo, si `a`, `b` y `c` son cuaterniones, las siguientes instrucciones deben ser válidas para un usuario de la librería:

   ```
   b + 3
   a * 3.0 + 7.0
   (b + b) * &c
   ```

   En este caso, el resultado debe ser el cuaternión con la operación propuesta. Esto es:

   $$a + b \times i + c \times j + d \times k \oplus n = a \oplus n + b \times i + c \times j + d \times k$$

   Investigue herramientas para pruebas unitarias y cobertura en su lenguaje escogido y agregue pruebas a su programa que permitan corroborar su correcto funcionamiento. Como regla general, su programa debería tener una cobertura (de líneas de código y de bifuración) mayor al 80%.

5. **RETO EXTRA: *GOLF!***

Considere los números de Narayana:

$$N_{n,k} = \frac{1}{n}\binom{n}{k}\binom{n}{k-1}$$

Considere también la función para hallar el n–ésimo número en la secuencia de fibotribonaccinacci:

$$trib(n) = \left\{ \begin{array}{ll} n & \text{si } 0 \leq n < 3 \\ trib(n-1) + trib(n-2) + trib(n-3) & \text{si } n \geq 3 \end{array} \right\}$$

Finalmente, considere el logaritmo en base 2:

$$log_2(n) = m \Leftrightarrow 2^n = m$$

Definiremos la función *maldad* como:

$$maldad(n) = trib\left(\left\lfloor \log_2(N_{n,\lfloor \log_2 n \rfloor}) \right\rfloor + 1\right)$$

Sabiendo que $\lfloor x \rfloor$ es el piso de x (el mayor valor entero que es menor o igual a x)

Desarrolle, en el lenguaje de su elección, un programa que:

- Reciba por entrada estándar o argumento del sistema un valor para n, tal que $n \geq 2$ (esto puede suponerlo, no tiene que comprobarlo).

  Debe indicar en su informe claramente si su reto recibe la entrada vía entrada estándar o argumento del sistema.
- Imprima el valor de $maldad(n)$.

Su programa debe imprimir el valor correcto y tomando menos de 1 segundo de ejecución, por lo menos hasta $n = 50$.

Reglas del reto: Intente desarrollar su programa con la menor cantidad de caracteres posibles (incluyendo espacios en blanco).

- El ganador del reto tendrá 5 puntos extras.
- El segundo lugar tendrá 3 puntos extras.
- El tercer lugar tendrá 1 punto extra.