Lenguajes de Programación, 2021-2
Examen Final

Profesora: Karla Ramírez Pulido

18 de agosto de 2021
Facultad de Ciencias UNAM

1. (2.5 pts.) Menciona, explica y ejemplifica los siguientes conceptos:
   (a) Sintaxis
   (b) Semántica
   (c) Convenciones de programación (Idioms)
   (d) Lenguaje objetivo
   (e) Lenguaje anfitrión

2. (1 pt.) Explica los paradigmas (a) *orientado a objetos*, (b) *funcional* de los lenguajes de programación, menciona dos lenguajes que pertenezcan a cada paradigma.

3. (1.5 pts.) Explica todos los tipos de análisis que se requieren para generar código ejecutable dado el código fuente.

4. (1 pt.) Explica los tres tipos de identificadores que hay en un lenguaje de programación y explica por qué la evaluación de las expresiones depende únicamente de las variables libres. Da un ejemplo usando la gramática del lenguaje FWAE visto en clase.

5. (1 pt.) Explica por qué el algoritmo de sustitución visto en clase es de $O\left(n^2\right)$.

6. (1 pt.) Transforma el siguiente código usando direcciones léxicas y evalúalo.

```
{with {a 1}
   {with {b 2}
      {with {c {+ b 3}}
         {with {d {+ a b}}
            {with {e 4}
                {+ e {+ d {+ c {+ b a}}}}}}}}}
```

7. (2 pts.) Evalúa la siguiente expresión utilizando (a) alcance estático y (b) alcance dinámico, es necesario poner el ambiente (stack) de ejecución usando representación de listas para poder evaluar.

```
{with {x -2}
    {with {x 2}
        {with {g {fun {y} {+ x y}}}
            {with {f {fun {z} {- z x}}}
                {with {x 3}
                    {with {x 2}
                       {f 10}}}}}}}
```

8. (2 pts.) Evalúa la siguiente expresión usando:

   (a) Evaluación glotona y alcance estático
   (b) Evaluación glotona y alcance dinámico
   (c) Evaluación perezosa y alcance estático
   (d) Evaluación perezosa y alcance dinámico

   y escribe el ambiente con representación de stack cuando la evaluación es perezosa y otro cuando la evaluación es glotona.

```
{with {x {+ 10 10}}
   {with {y {- x 5}}
      {with {f {fun {z} {+ x {+ y z}}}}
          {with {x {+ 5 5}}
              {with {g {fun {w} {+ x {+ y w}}}}
                 {with {x 2}
                    {with {y 1}
                       {with {z -10}
                           {with {w -5}
                              {f 15}}}}}}}}}}
```

9. (1 pt.) ¿Por qué fue necesario introducir *closures* para evaluar expresiones con alcance estático en la función `interp` vista en clase?

10. (1 pt.) ¿De qué otra manera se podría implementar en un lenguaje alcance estático si no hubiera *closures*?

11. (1.5 pts.) Dentro del Cálculo Lambda, evalúa cada una de las siguientes expresiones usando β-reducciones, Si alguna tiene Forma Normal, especifícala.

    (a) $(\lambda x.x)\,(\lambda x.xxx)$
    (b) $(\lambda x.\,(\lambda y.yxw)\,z)\,u$
    (c) $(\lambda x.\lambda y.\lambda z.x)\,(yz)$

12. (1 pt.) ¿Qué es y para qué sirve el Combinador Y? Da un ejemplo.

13. (1.5 pts.) Explica 3 características de la evaluación perezosa y a qué se le conoce como punto estricto en lenguajes perezosos.

14. (1.5 pts.) ¿Cuáles son las implicaciones que trae consigo la recursión de cola en comparación con la recursión normal en cuanto a la memoria utilizada en tiempo de ejecución?

15. (2 pts.) Escribe dos funciones en Racket que calculen el fibonacci de un números usando (a) recursión y (b) recursión de cola. Pon el número total de registros de activación usados en ambos casos y escribe el stack con los registros de activación en cada llamada tanto de la función recursiva como de la que implementes usando recursión de cola cuando se llame a ambas funciones con el argumento 3.

16. (1.5 pts.) Define y ejemplifica el concepto de continuación (*continuation*) ¿qué es? ¿cuáles son sus características ¿cómo es su comportamiento en memoria? su utilidad, etc.

17. (1.5 pts.) Evalúa los siguientes códigos y pon la función con representación $\lambda \uparrow$ que estás evaluando:

(a)
```
(+ (call/cc (lambda (k)
          (k (begin
                (set! +8_{woo} k)
                (display "inside body")
                5)))
       8)
```

(b)
```
(+ 8 (call/cc (lambda (k)
             (+ 1 (+ 2 (+ 3 (k 2)))))))
```

18. (1 pt.) Implementa la siguiente función recursiva usando CPS.

```
(define (filter-pos l)
   (cond
       [(empty? l) empty]
       [else (if (> (car l) 0)
                 (cons (car l) (filter-pos (cdr l)))
                 (filter-pos (cdr l)))]))
```

19. (1.5 pts.) ¿Por qué se introdujeron las operaciones con cajas al intérprete? Explica a profundidad.

20. (2 pts.) Evalúa la siguiente expresión usando:

    (a) Paso de parámetros por valor

    (b) Paso de parámetros por referencia

```
{with {{a 1}
       {b -1}
       {swap {fun {x y}
                  {with {{tmp x}}
                     {seqn {set x y}
                           {set y tmp}}}}}}
   {seqn {swap a b}
         {- a b}}}
```

21. (1 pt.) Evalúa la siguiente expresión usando paso de parámetros por necesidad.

```
{with {{b 3}
       {f {fun {x y}
             {seqn {set x 4}
                   y}}}}}
   {+ {f b b} b}}
```

22. (1 pt.) Evalúa la siguiente expresión usando paso de parámetros referencia-regreso.

```
{with {{b 2}
       {f {fun {x}
             {seqn {set x 4}
                   {+ x b}}}}}
    {+ {f b} b}}
```

23. (2 pts.) Define los siguientes conceptos y pon ejemplos relacionados con cada uno:

    (a) Tipo (tipos básicos, Tipos Abstractos de Datos, en particular con los tipos básicos da una categorización de éstos)

    (b) Lenguaje de propósito general.

    (c) Lenguaje envolvente.

    (d) Lenguajes con tipificado estático.

24. (2 pts.) Explica a profundidad las propiedades de (1) consistencia y (2) seguridad, da ejemplos de ambos.

25. (1.5 pts.) Menciona 3 ventajas que presentan los lenguajes tipificados implícitamente sobre los tipificados explícitamente.

26. (3 pts.) Menciona si son falsas (F) o verdaderas (V) cada uno de las afirmaciones sobre el sistema verificador de tipos.

    (a) ____ El sistema verificador de tipos no rechaza algunos programas eficientes.

    (b) ____ El sistema verificador de tipos ayuda a reducir el tiempo de depuración de un programa.

    (c) ____ El sistema verificador de tipos rechaza algunos programas válidos.

    (d) ____ En tiempo de ejecución nunca se verifican los tipos de un programa.

    (e) ____ El sistema verificador de tipos detecta errores en programas que hayan pasado el análisis sintáctico.

    (f) ____ El sistema verificador de tipos cacha todos los errores en tiempo de compilación.

27. (2 pts.) Realiza el juicio de tipo para las siguientes expresiones:

    (a)
    ```
    {rec {fib {<implementa fibonacci>}} {fib 2}}
    ```

    (b)
    ```
    {{fun {x : number} : number
       {+ {* x y} {+ 2 2}}}
     8}
    ```

28. (1.5 pts.) De la siguiente expresión, da la inferencia de tipos y menciona de qué tipo son las variables al final de la inferencia realizada, asígnale un nombre mnemotécnico a la función.

    (a)
    ```
    (define (mystery lis)
       (cond
          [(empty? lis) 0]
          [else (+ 1 (mystery (cdr lis)))]))
    ```

29. (2 pts.) Utiliza el algoritmo de unificación visto en clase con la expresión `((lambda (x) x) (* 7 1))`.

30. (1.5 pts.) Da tres ventajas de contar con polimorfismo explícito en un lenguaje de programación. Utiliza ejemplos.

31. (1.5 pts.) Explica los conceptos y las diferencias que existen entre el uso de macros y de funciones en un lenguaje de programación.

32. (2 pts.) Dentro del paradigma de los lenguajes orientados a objetos, da un programa donde pongas en práctica los conceptos de objetos, clases, herencia y polimorfismo explícito. Explica cada uno de los conceptos y su uso en tu programa (no usar pseudo-código).