PONTIFICIA UNIVERSIDAD CATÓLICA DE CHILE
ESCUELA DE INGENIERÍA
DEPARTAMENTO DE CIENCIA DE LA COMPUTACIÓN

IIC2233 - Programación Avanzada
2^o semestre 2016

Tarea 3

1. Objetivos

- Aplicar conceptos y nociones de programación funcional para el correcto modelamiento de un problema.

2. Introducción

Luego de una ardua jornada de programar tus juegos preferidos, decides dejar de lado el ocio y comenzar a hacer programas que te puedan ayudar a avanzar con tus estudios universitarios, más especificamente, en el área matemática. Para ello, te propones crear tu propio intérprete de expresiones matemáticas nunca antes visto, capaz de resolver las operaciones más complicadas en tiempos muy reducidos. Entusiasmado con tu proyecto, decides nombrar tu creación llamándola Maplemathica.

3. Problema

En esta tarea, deberá implementar un intérprete de expresiones matemáticas mediante el paradigma de programación funcional.

4. El intérprete Maplemathica (66 %)

Maplemathica[1] funciona como un intérprete interactivo mediante comandos ingresados por consola, La sintaxis de los comandos serán especificados mas adelante. Además deseas poder guardar el estado actual de tu programa y cargar estados anteriores, tanto las variables creadas en el transcurso del programa como las funciones definidas. De igual forma, tu programa debe ser capaz de leer un archivo con comandos y entregar un archivo con todas las respuestas respectivas.

4.1. Sintaxis.

Tu programa debe ser capaz de recibir comandos vía consola siguiendo la siguiente sintaxis, con el fin de llegar al resultado deseado:

- **Definición de funciones y variables:**

[1] Totalmente original y definitivamente nunca antes visto.

- `f[x_] = x;`[2]
- `g[x_,y_] = Sin[x] * (Cos[y] ^ 2);`
- `x = 5;`
- `a = 2; b = 7; c = a * b;`
    - **Funciones por partes:**
        - `f[x_] = Piecewise[x, x <0; x*2, x>= 0];`

- **Operadores Booleanos**
    - **Mayor, Menor, Menor o igual, Mayor o Igual, And, Or:** >, <, >=, <= , ==, != ,&&, ||
- **Notaciones Básicas**
    - **Suma:** +
    - **Resta:** -
    - **Multiplicación:** *
    - **División:**
        - **División Normal:** /
        - **División Entera:** //
        - **Resto:** %
    - **Potencia:** ∧
    - **Π:** Pi
    - **Logaritmo Natural:** Ln[x]
    - **Exponencial:** Exp[x]
- **Funciones Trigonométricas:**
    - **Seno:** Sin[x]
    - **Coseno:** Cos[x]
    - **Tangente:** Tan[x]
    - **Secante:** Sec[x]
    - **Cosecante:** Csc[x]
    - **Arcseno:** ArcSin[x]
    - **Arccoseno:** ArcCos[x]
    - **Arctangente:** ArcTan[x]

[2] Los polinomios pueden ser de n grados

- **Factorial:** 5!
- **Valor Absoluto:** Abs[7 - 8]
- **Derivar:** Derivate[función, variable]
  - **Ejemplos:**

    Derivate[Sin[x], x]
    Derivate[Sin[x]*Cos[x],x,y]
- **Integrar:** Integrate[función, {variable, cota inferior, cota superior }]
  Ojo que se puede no especificar la cota inferior y superior.
  - **Ejemplos:**

    Integrate[Cos[x],{x}]
    Integrate[Tan[x],{x,0,Pi/3}]
    Integrate[Integrate[(x∧2)*3*(y∧3),{x,0,1/2}],{y,3,5}]
- **Sumatorias:** Sum[variable/función, {variable, cota inferior, cota superior}]
  - **Ejemplos:**

    Sum[x, {x, 0, 10}]
- **Gráficos:**
  - **Gráfico 2D:**

    Plot[variable/funcion, parametro, cota inferior, cota superior, color, linewidth]
  - **Gráfico Regiones:**

    RegionPlot[variables/funciones, parametro, cota inferior, cota superior, color, linewidth]
  - **Gráfico 3D:**

    Plot3D[variables/funciones, parametro, cota inferior, cota superior, color, linewidth]
  - **Colores disponibles:**
    - b: blue
    - g: green
    - r: red
    - c: cyan
    - m: magenta

- y: yellow
- k: black
- w: white

- **Linewidth:**
  Grosor del gráfico (2D)
  - 0, 0.1, 0.2, etc

- **Matrices:**

  - Definición de Matrices:
    - Filas
      M = [1,2,3,4,5]
    - Columnas
      C = [6 |7 |8 |9 |10]
    - $Matrices_{mxn}$
      Dos formas:
      $Matriz_{3x3}$ = [[$a_{1,1}$ |$a_{1,2}$ |$a_{1,3}$], [$a_{2,1}$ |$a_{2,2}$ |$a_{2,3}$], [$a_{3,1}$ |$a_{3,2}$ |$a_{3,3}$]]

      $Matriz_{3x3}$ = [[$a_{1,1}$, $a_{1,2}$, $a_{1,3}$], [$a_{2,1}$, $a_{2,2}$, $a_{2,3}$], [$a_{3,1}$, $a_{3,2}$, $a_{3,3}$]]

    - **Ejemplo matriz de 3x2:**
      $Matriz_{3x2}$ = [[1 |2], [4 |5], [7 |8]]
      De forma equivalente:
      $Matriz_{3x2}$ = [[1,2],[4,5],[7,8]]
  - Multiplicación de matrices: MatrixMultiply[Matriz1, Matriz2]
  - Determinante: Det[Matriz]
  - Dimension y Rango de una matriz:
    - Rango: Range[Matriz]
    - Dimension: Dim[Matriz]
  - Traspuesta de una matriz: Trans[Matriz]
  - Inversa de una matriz: Inv[Matriz]

- **Comandos Auxiliares:**

  - Borrar una variable: ClearV variable.

- Ejemplo: $ClearVa$;

- Borrar una funcion: ClearF funcion.
  - Ejemplo: $ClearFg[x]$;

- Borrar todas las variables: ClearAllV

- Borrar todas las funciones: ClearAllF

- Desplegar las variables definidas hasta el momento: Who

- Ayuda: Comando**?**[3]

  - Ejemplo: Integrate?
- Cargar y guardar archivos
  - Cargar un archivo: load nombrearchivo;
  - Guardar un archivo: save nombrearchivo;
- Simplificar los resultados
  - Simplificar el útlimo resultado: %S
  - Simplificar un resultado: FullSimplify[operación/resultado];
    - Ejemplos:
      $FullSimplify[Integrate[x \wedge 2,\{x\}]/x]$[4]

- **Resolver Ecuaciones:**

- Solve[expresiones, variables, dominio[5]]
  - **Ejemplos**
    $Solve[(x \wedge 2) * 2 * x - 3 == 0,\{x\}]$[6]
    $Solve[(x \wedge 2) * 2 * x - 3 == 0 \&\& x > 0,\{x\}]$[7]
    $Solve[Cos[x] == 0,\{x\}]$[8]
    $Solve[Cos[x] == 0,\{x\}, N]$[9]

- **Consultas booleanas:**

- Divisible[a,b]: si a es divisible por b

- MCM[a,b,c]: si a es el mínimo común múltiplo entre b y c

- MCD[a,b,c]: si a es el máximo común divisor entre b y c

- **IMPORTANTE:** El ";" indica el término de la instrucción.

---

3 La idea es que haya un pequeño resumen de la función y que muestre los parámetros que recibe
4 Output: (x∧ 2) / 3
5 Pueden ser Reals, Integers, Img, N. Corresponden a los reales, enteros, imaginarios, expresión numérica respectivamente.
6 Output: x = -3 , x = 1
7 Output: x = 1
8 Output: x = -Pi/2, x = Pi/2
9 Output: x = -1.5708, x = 1.5708

4.2. Lectura de archivos

Además de las funcionalidades explicadas, tu programa debe ser capaz de trabajar con archivos entregados por el usuario. En particular:

4.2.1. Importar/exportar estados de programa

Se debe dar la opción, a través de la interacción por consola, para que el usuario guarde el estado actual del trabajo con Maplemathica. Es decir, en el archivo debe estar la información necesaria para poder volver a cargar dicho estado. Análogamente, se debe dar la opción de importar uno de estos archivos para volver a un estado deseado. Los estados se guardan/cargan con nombres dados por el usuario.

- Un ejemplo de estado:

```
---- status1.txt ----
f[x_] = x^2 + 3;
g[x_] = Sin[x];
t[x_] = Cos[x];
h[x_] = g[f[x]];
a = 0; b = 2; c = 3; d = Pi;
p[x_,y_,z_] = (Sin[d*x])^b + y*Sin[z];
int = Integrate[f[x], {x, a, c}];
```

  La primera línea no es estrictamente necesaria. Es importante notar que la sexta línea no tiene ningún error, el carácter ";"finaliza las definiciones por separado.[10]

Un usuario podría cargar un primer estado en cierto punto de la ejecución, y posteriormente cargar un segundo estado. En este caso, las variables que tenían en común dichos estados se sobreescriben por las del estado más reciente, manteniéndose las variables del primer estado que no están contenidas en el segundo. Por ejemplo, si en el primer estado estaban definidas las variables A y B, en el segundo estado las variables B y C, quedarían definidas las variables A (valor del primer estado), B (valor del segundo estado), C (valor del segundo estado).

[10] Los archivos de estado podrían ser una línea gigante de definiciones

4.2.2. Archivo con consultas

Debe ser posible cargar un archivo de texto con consultas. Éstos archivos tendrán un formato específico, donde se especificará el estado que se quiere consultar, seguido de un número determinado de consultas. Este archivo se cargará al comenzar cada ejecución del programa y tendrá por nombre *consultas.txt*

- Formato de los archivos:

```
nombre_status1, k1
consulta_1;
...
consutak_1;
nombre_status2, k2
consulta_1;
...
consulta_k2;
...
nombre_statusn, kn
consulta_1;
...
consulta_kn;
```

- Un ejemplo referido al status del punto anterior:

```
status1.txt, 5
Integrate[g[x], {x, a, d}];
Integrate[f[t[x]], {x}];
Derivate[p[x,y,z], z];
MCM[int, 2,5];
Integrate[Integrate[Integrate[p[x,y,z], {x, a, d}], {y, a, 1}], {z, a, d}];
```

Las consultas podrían referirse a gráficos (plot, regionplot, etc.). En dichos casos, se deben guardar los gráficos correspondientes con el nombre que estime conveniente.[11]. Además, debe mostrarse en el output correspondiete el nombre con el que se guardó la imágen del gráfico.

4.2.3. Escribiendo los resultados

Luego de leer el archivo con consultas ya mencionado, se debe generar un archivo *resultados.txt* el que, como su nombre lo dice, debe contener los resultados de las consultas referentes a sus status.

[11] Puede buscar sobre el método savefig() de la librería matplotlib

- Formato:

```
---- resultados_nombre_status1 ----
resultado_consulta_1;
...
resultado_consutak_1;
---- resultados_nombre_status2 ----
resultado_consulta_1;
...
resultado_consulta_k2;
...
---- resultado_nombre_statusn ----
resultado_consulta_1;
...
resultado_consulta_kn;
```

- Ejemplo de resultados referidos a la consulta del punto anterior:

```
---- resultados_status1.txt ----
2
(7*x)/2 + (1/4)*Sin[2*x]
y*Cos[z]
False
(Pi/2)*(2 + Pi)
```

5. Apartado matemático (15 %)

5.1. Algunas series de utilidad

- $\sin(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{(2n+1)!}$, $\forall x$
- $\cos(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!}$, $\forall x$
- $\arcsin(x) = \sum_{n=0}^{\infty} \frac{(2n)! x^{2n+1}}{4^n (n)!^2 (2n+1)}$, para x $\in [-1, 1]$
- $\arccos(x) = \frac{\pi}{2} - \sum_{n=0}^{\infty} \frac{(2n)! x^{2n+1}}{4^n (n)!^2 (2n+1)}$, para x $\in [-1, 1]$
- $\arctan(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{2n+1}$, para x $\in [-1, 1]$
- $\log(x) = \sum_{n=0}^{\infty} \frac{1}{2n+1} (\frac{x^2-1}{x^2+1})^{2n+1}$,para x > 0
- $e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$, $\forall x$

5.2. Identidades trigonométricas

- $\tan(x) = \frac{\sin(x)}{\cos(x)}$
- $\cot(x) = \frac{\cos(x)}{\sin(x)}$
- $\sec(x) = \frac{1}{\cos(x)}$
- $\csc(x) = \frac{1}{\sin(x)}$

6. Programación Funcional (12 %)

Debes modelar toda la implementación con programación funcional. Se permite el uso de loops (`for` y `while`) **sólo en listas y generadores por comprensión**. A continuación, se mostrarán ejemplos del uso de loops no permitidos y permitidos[12]:

```
# Este 'for' no se puede utilizar
for empleo in empleos:
        dame_comida(empleo)

# El 'for' anterior puede ser realizado mediante la función 'map'
map(dame_comida, empleos)

# Este 'for' sí se puede utilizar
nombres_empleados = [x.nombre for x in empleados]
```

7. Manejo de Errores (7 %)

Su programa debe ser capaz de reaccionar adecuadamente frente a errores comunes dentro de las consultas. A continuación, se les especificará el tipo de errores a los que se verá enfrentado su programa, y que hacer ante él.

- **Variables. Funciones y su Dominio:**
  - Cuando se defina una funcion, su programa debe reconocer el Dominio de esta, y revisar si el/los parametro/s entregados pertenecen al Dominio, en caso contrario: **Error**[13]
  - Si se hace un llamado a una funcion/variable que no se haya definido previamente o no se haya cargado desde un archivo de estado: **Error**
  - Si se trata de sobreescribir una funcion/variable ya existente sin antes borrarla: **Error**
- **Multiplicacion de Matrices:** Sea A_{mxn} y B_{oxp} y se quiere multiplicar: A*B, si n != o: **Error**
- **Logaritmo natural:** Ln[x], si x <= 0: **Error**[14]
- **Division por Cero:** Si se intenta dividir por cero: **Error**

Se puede asumir que **NO** existirán otro tipo de errores que esten fuera del listado anterior, como por ejemplo: Errores de sintaxis en los archivos.

Junto al enunciado, habrán algunos archivos de ejemplo con consultas y sus respectivos resultados. **Es importante que su programa funcione con cualquier tipo de archivo (siempre que mantengan la estructura detallada anteriormente). No lo adapten a los archivos de ejemplo.**

8. Restricciones y alcances

- Tu programa debe ser desarrollado en Python 3.5
- Esta tarea es estrictamente individual y está regida por el Código de Honor de la Escuela: Clickear para Leer.
- Su código debe seguir la guía de estilos `PEP8`

[12] Nos interesa que no usen los ciclos `for` y `while` en lo que son las funcionalidades lógicas del programa. Pueden utilizarlos para imprimir valores o esperar input del usuario, por ejemplo.

[13] Si estas en consola, debes imprimir el error, si este error aparece durante la revision de un archivo, en el archivo de respuestas.txt debe salir especificado que la consulta falló.

[14] Este error esta de alguna manera implícito en el reconocimiento del Dominio de una función.

- Si no se encuentra especificado en el enunciado asuma que el uso de cualquier librería Python está prohibido. Pregunte por foro si se pueden usar librerías específicas.
- El ayudante puede castigar el puntaje[15] de tu tarea si le parece adecuado. Se recomienda ordenar el código y ser lo más claro y eficiente posible en la creación de algoritmos.
- Debe adjuntar un archivo `README.md` donde se comenten los alcances y el funcionamiento de su sistema (*i.e.* manual de usuario) de forma *concisa* y *clara*. **Tiene hasta un día después de la fecha límite para subir este archivo.**
- Cree un módulo para cada conjunto de clases. Divídalas por las relaciones y los tipos que poseen en común. **Se descontará hasta un punto si se entrega la tarea en un solo módulo[16].**
- El no uso de funcional será severamente castigado.
- Cualquier aspecto no especificado queda a su criterio, siempre que no pase por encima de ningún otro.
- Está **prohibido** el uso de las librerías `math`, `numpy`, `scipy`, `pandas` para el cálculo de las operaciones matemáticas. Solo está **permitido** el uso de `matplotlib` para graficar las funciones.

9. Entrega

- **Fecha/hora:** 07 de Octubre - 23:59 hrs
- **Lugar:** GIT - Carpeta: Tareas/T03

Tareas que no cumplan con las restricciones señaladas en este enunciado tendrán la calificación mínima (1.0).

[15] Hasta −5 décimas.
[16] No agarre su código de un solo y lo divida en dos módulos, module su código de forma inteligente