2ME3 - Assignment 2

Please read this document very carefully. Follow instructions exactly. If you have any questions please post them to MS Teams or ask during office hours.

This assignment is due Nov 14th, by 11:59pm

I have created an Assignment 2 channel in Teams. If you have questions about the assignment, please post them there. Thank you.
Unless specifically stated, assume you are not allowed to import external libraries.

Purpose

The primary objective of this assignment is to assess your ability to interpret a mathematical specification. The main difficulty of this assignment is not intended to come from the coding of your classes, but rather, deciphering exactly what the specification is saying.

Note:

- I have made the specifications intentionally unintuitive at times. That is, I have given functions and variables generic names, and may have stated some things in way which are less elegant than they could have been stated.
- Myself and the TAs are happy to answer questions regarding notation, but questions such as "should $f4$ return true if there are... " will most likely be answered with: "Please read the specification."

Overview

There are no other files associated with this document. You are responsible for submitting three files:

1. A2.pdf, .txt, .docx (as long as we can read it)
2. `Mystery1.java`
3. `Mystery2.java`

See below for details on what you are responsible for completing.

Your Tasks

At the end of this document are two separate MIS's. You are responsible for implementing these two specifications. Mystery1 and Mystery2 are not associated with each other in any way. Think of them as two separate questions. Submit these implementations as `Mystery1.java` and `Mystery2.java` respectively. You may import/use ArrayLists if you wish.

You are also responsible for giving a plain English explanation of the following access routines:

- $f4()$ from `Mystery1.java`
- $f2()$ from `Mystery2.java`

These explanations need not be long (no longer than a half page, and potentially much shorter than that). They should explain the nature of what these access routines are actually doing. To aid in your explanation, think of an practical application of both modules.

Submitting and Grading

This assignment will be submitted electronically via Avenue. Part of your assignment will be auto graded, part will be done manually. A rough breakdown is given below.

- `Mystery1.java`: 40%
- `Mystery2.java`: 40%
- `A2.pdf` methods: 20%

Code which does not compile will be heavily penalized. Good luck!

Academic Dishonesty Disclaimer

All of the work you submit must be done by you, and your work must not be submitted by someone else. Plagiarism is academic fraud and is taken very seriously. The department uses software that compares programs for evidence of similar code.

Please don't copy. The TAs and I want you to succeed and are here to help. Here are a couple of general guidelines to help you avoid plagiarism:

Never look at another assignment solution, whether it is on paper or on the computer screen. Never show another student your assignment solution. This applies to all drafts of a solution and to incomplete solutions. If you find code on the web that solves part or all of an assignment, do not use or submit any part of it! A large percentage of the academic offenses in involve students who have never met, and who just happened to find the same solution online. If you find a solution, someone else will too.

Mystery1 Module Interface

Uses

None

Syntax

Exported Types

Mystery1 = ?

Exported Access Routine

Routine Name	In	Out	Exceptions
new Mystery1		Mystery1	None
f1		$\mathbb{N}$	None
f2	String		None
f3	String, String		None
f4	String, String	$\mathbb{B}$	None

Semantic

Local Types

String = char[]
X = tuple(x:String, y:String)

State Variables

S_1 : String{ }
S_2 : X{ }

Assumptions

- All inputs are of the proper type.
- The notation $X(s_1, s_2)$ is shorthand for saying (s_1, s_2) is a tuple of type X, where $x = s_1$ and $y = s_2$.

Access Routine Semantics

Mystery1():

- transition: S_1, S_2 = { }, { }
- output: out := this

f1():

- output: out := $|S_1|$

f2(s):

- transition: $S_1 := S_1 \cup \{s\}$

f3(s_1,s_2):

- transition: $S_2 := S_2 \cup \{X(s_1, s_2)\}$

f4(s_1,s_2):

- output: out := $s_2 \in \cup(s : \text{String } | s \in g(s_1) : \ g(s)) \wedge s_2 \notin g(s_1)$

Local Functions

$g(s) : \text{String} \rightarrow \text{String}\{\ \}$
$g(s) = \{s' \ : \ X(s, s') \in S_2\}$

Mystery2(T) Generic Module

Uses

None

Syntax

Exported Types

Mystery2 = ?

Exported Access Routine

Routine Name	In	Out	Exceptions
new Mystery2		Mystery2	None
f1	T, T		Exception
f2		T	Exception

Semantic

Local Types

X = tuple(x_1:T, x_2:T$\cup\{k\}$, x_3:$\mathbb{N}$)

Local Constants

$k = -1$

State Variables

$S = X\{\ \}$

Assumptions

- All inputs are of the proper type.
- $k \notin T$
- The notation $X(a, b, c)$ is shorthand for saying (a, b, c) is a tuple of type X, where $x_1 = a$, $x_2 = b$ and $x_3 = c$.

Access Routine Semantics

Mystery2():

- transition: $S = \{\ \}$
- output: out := this

f1(t_1, t_2):

- transition: $S := S \cup \{X(t_1, t, n)\}$, where the following all hold:

1. $\forall(x \ : \ X|x \in S : x.x_3 < n)$
2. $\exists(x \ : \ X|x \in S : x.x_1 = t_2) \Rightarrow t = t_2$
3. $\neg\exists(x \ : \ X|x \in S : x.x_1 = t_2) \Rightarrow t = k$

- exception: $\exists(x : X|x \in S : x.x_2 = t_1) \Rightarrow$ Exception

f2():

- transition: $S := g_2(x.x_1) \cup g_3(x.x_1) - \{x\}$, where: $x \in g_1() \wedge \forall(x' \ : \ X|x' \in g_1() : \ x'.x_3 \leq x.x_3)$
- output: out := $x.x_1$ where, $x \in g_1() \wedge \forall(x' \ : \ X|x' \in g_1() : \ x'.x_3 \leq x.x_3)$
- exception: $S = \{\} \Rightarrow$ Exception

Local Functions

$g_1()$:None $\rightarrow$ X{ }
$g_1() = \{x \ : \ x \in S \wedge x.x_2 = k\}$

$g_2(t) : T \rightarrow$ X{ }
$g_2(t) = \{X(x_1, k, x_3) \ : \ X(x_1, t, x_3) \in S\}$

$g_3(t) : T \rightarrow$ X{ }
$g_3(t) = \{X(x_1, x_2, x_3) \ : \ X(x_1, x_2, x_3) \in S \wedge x_2 \neq t\}$

Considerations

It is assumed that $t_1 = t_2$, where $t_1, t_2 : T$, would be implemented using the `.equals` method in Java. That is, $t_1 = t_2$ would be equivalent to `t1.equals(t2)`.