CS103 Machines, Languages and Computation

Workbook for Assignments 6-10, Semester 1, Weeks 7-11

Version 1, November 2nd 2015

Name:

Matriculation number:

This workbook contains the five assignments for the second half of CS103 Machines, Languages and Computation in Semester 1.

You should attempt all five assignments before the second Class Test, which will be on Friday 4th December 2015 at 10am.

Keep this workbook – when you've filled it in with your answers, you can use it as a revision aid for the class test.

The Class Test

The Class Test will contain five questions, one on each assignment. You are asked to attempt all five questions. Each question is marked out of 20.

The questions will be directly related to the assignments in this workbook.

Assessment Structure

Your final mark for this class will be 7.5% for each of the four class tests and 70% for the 2 hour degree exam in May/June 2015.

If you pass all four class tests (with 40% or more in each) and achieve an average of more than 60%, then you will be exempted from the degree exam. Your final mark for the class will be your average mark in the four class tests.

Assignment 6: Languages and Recursion

Consider the following grammar:

Sentence → NounPhrase Verb2 NounPhrase
NounPhrase → Name
NounPhrase → Article NounGroup
NounGroup → Noun
NounGroup → Adjective Noun

Name → “Brian” | “Beryl”
Verb2 → “ate” | “fed” | “chased” | “kissed”
Article → “a” | “the”
Adjective → “big” | “purple” | “ugly” | “fat” | “eccentric”
Noun → “frog” | “dog” | “clock” | “monster” | “princess”

(a) Generate some random sentences using the grammar.

(b) Give the derivation of the sentence “Brian ate a big frog”.

(c) Can you modify the grammar so it can generate noun phrases containing *any* number of adjectives, such as “Brian ate a big fat ugly purple frog” but *without* increasing the number of rules?

(d) Extend the definition of the portion of the grammar that generates names so that it can generate lists of names as a NounPhrase as in the following examples:

“*Brian* ate a frog.”
“*Brian and Beryl* ate a frog.”
“*Brian and Beryl and Fred* ate a frog.”
“*Brian and Beryl and Fred and Bill* ate a frog.”
(etc)

Assignment 7: Recursive Grammars and Regular Expressions

Consider the following grammar:

$S \rightarrow AB$
$A \rightarrow aA$
$A \rightarrow a$
$B \rightarrow b$

(a) Generate some strings using this grammar.

(b) What regular expression describes the language this grammar generates?

(c) Write a grammar which can generate strings of the form a^+b^+, e.g. ab, aaab, abbbb, abbb, but not aaa, bbb ($^+$ means “repeated at least once”).

(d) Write a grammar which can generate strings of the form $(a^+b^+)^+$, e.g. ab, aabb, aaaabbaabb, ababbbb, but not aaaa, aabbaa, bbabbaaabb, etc.

(e) Can you write a grammar to can generate strings of the form a^nb^n, e.g. ab, aabb, aaabbb, aaaabbbb, but not abbb, aaab, aaaabb, etc?

Assignment 8: Knowledge Representation and Inference

(a) Consider the following database of facts and rules:

cat(tom)
mouse(jerry)
dog(spike)
dog(tyke)
father(spike,tyke)
hits(tom,tyke)

cat(X), mouse(Y) → fights(X,Y)
dog(X), cat(Y) → hates(X,Y)
fights(X,Y) → fights(Y,X)
hits(X,Z), father(Y,Z) → hits(Y,X)

Give the derivation of these facts, if possible:

fights(tom,jerry)

fights(jerry,tom)

hates(spike,tom)

hates(tom,spike)

hits(spike,tom)

(b) In some of the Tom and Jerry cartoons, Jerry and Spike are friends because Tom fights Jerry and Spike hates Tom. Can you encode this as a rule and use it to derive the fact “friends(spike,jerry)”?

(c) Translate the following sentences into a database of facts and rules:

Eric is a lion.
Fred is a tiger.
Gerald is a gorilla.
Every lion is friends with Gerald.
Every tiger is friends with Eric.
If A is friends with B, then B is friends with A.
If A is friends with B and A is friends with C, then B is friends with C.

(d) Use your database to derive the fact that Fred is friends with Gerald.

Assignment 9: The Lambda Calculus

(a) Write the following functions as lambda expressions:

(i) f(x) = x+5

(ii) f(v) = and(v,v)

(iii) f(x,y) = x+y

(iv) f(x,y,s) = (x+y)*(length s)

(b) Perform beta reduction on the following:

(i) (λx.even x) 2

(ii) (λs.(length s)+3) “abc”

(iii) (λa.λb.(length a)+(length b)) “fred” “eric”

(c) Give the types of the lambda expressions in Q2.

(d) (hard) Consider the function application as given in Lecture 16 on the slide entitled “Functions as Arguments”:

(λf.f 3) (λx.x+2)

What is the type of the expression (λf.f 3) ?

Assignment 10: Recursion and Function Definitions

(a) Which of the following Python functions are recursive?

(i)
```
def myNot(input):
    if input == False:
        return True
    else:
        return False
```

(ii)
```
def myNand(v1, v2):
    return myNot(myAnd(v1, v2))
```

(iii)
```
def member(t, l):
    if l == []:
        return False
    if t == l[0]:
        return True
    else:
        return member(t, l[1:])
```

(iv)
```
def factorial(n):
    result = 1
    while True:
        if n == 0:
            return result
        else:
            result = result * n
            n = n - 1
```

(v)
```
def myLength(l):
    if l == []:
        return 0
    else:
        return 1 + myLength(l[1:])
```

(b) Consider the Python function shown below:

```
def mystery(x):
    if x == []:
        return 0
    else:
        return x[0] + mystery(x[1:])
```

(i) What does `mystery([1,2,3])` return?

(ii) In general, what does this function do?

(iii) Why do we say that this function is recursive?

(iv) Draw a calling diagram for `mystery([1,3,5])`.