Programming Paradigms and Compiler Design (SS 2021)

Prof. Dr. Visvanathan Ramesh, Dr. Arne Naegel

Mock Exam Questions (Compilers) (Total points: 55)

C1: The following multiple choice questions will count for 1 points each. (Total: 8 points)

i) Compiler is a program that:

(A) Accepts a program written in a high level language and produces an object program

(B) Appears to execute a source program as if it were machine language

(C) Automates the translation of assembly language into machine language

(D) Places programs into memory and prepares them for execution

ii)Which of the following is/are the phases of compiler?

(A) Code generation

(B) Syntax analyser

(C) Lexical analyser

(D) All of these

iii) The lexical analyzer takes _____ as input and produces a list of ___ of output.

(A) Machine code, mnemonic

(B) Tokens, source code

(C) Source code, tokens

(D) Both a and b

iv) Bottom-up parsing method is also called

(A) Shift reduce parsing

(B) Predictive parsing

(C) Recursive descent parsing

(D) None of these

v) CFG (Context Free Grammar) can be recognized by a

(A) Push down automata

(B) Finite state automata

(C) 2 way linear bounded automata

(D) Both a and c

vi)A grammar that produces more than one parse tree for some sentence is called as

(A) Ambiguous

(B) Regular

(C) Unambiguous

(D) All of these

vii) While doing Code optimization, substitution of values for names (whose values are constants) is done in:

(A) Local optimization
(B) Loop optimization
(C) Constant Folding
(D) Strength Reduction

viii) Dead code elimination in machine code optimization refers to:

(A) Removal of all labels
(B) Removal of values that never get used
(C) Removal of function which are not involved
(D) Removal of module after its use

C2: Parse Trees/Grammars (Total: 7 + 13 = 20 points)

a) Define the following terms used in the definition of Grammars:
(i) a non-terminal symbol (1 point)
(ii) an ambiguous grammar (1 point)
(iii) a production (1 point)
(iv) a context-free grammar (2 points)
(v) a regular grammar (2 points)

b) Consider the following grammar in Backus Naur Form for phone numbers:

```
<phoneNumber> ::= <fullNumber> | <basicNumber>

<fullNumber> ::= <areaCode> <basicNumber>

<basicNumber> ::= <spaces> <fourDigits> <spaces> <fourDigits>

<fourDigits> ::= <digit> <digit> <digit> <digit>

<areaCode> ::= "(" <digit> <digit> ")"

<digit> ::= "0" | "1" | "2" | "3" | "4" | "5" | "6" | "7" |
"8" | "9"

<spaces> ::= " " <spaces> | ""
```

Note that the terminals are the numerals from 0 through 9 and the left and right parentheses.

A) Comment about whether given phone number strings such as “(01) 78990001” , “5678 3589” , are valid strings within the grammar specified. Note spaces inside the quotes are to be treated as space and the “” is used to mainly illustrate the grouping of characters including spaces in a given string. (3 points)
B) Illustrate the parsing of the strings using bottom-up parsing or top-down parsing. (4 points)
C) Briefly illustrate how the grammar described above maps directly to implement a functional parser in Haskell Provide sufficient details. (6 points)

C3: Register Machines (Total points: 9 + 4 = 13)

i) Consider a pure stack machine, i.e. one where the only storage is a stack, and an instruction $r = F(a\_1, .., a\_n)$: pops n operands from the stack, computes the operation F using the operands, and pushes the result on to the stack.

a) Illustrate the steps of working of a pure stack machine for computing the following – (2 + 5)*6 – 7 . Assume the set of F's possible include the binary operators 'add', 'multiply', 'subtract', 'divide' in addition to standard pop and push operations. (4 points)

b) How would solution differ when the pure stack machine is changed to a n-register accumulator machine with n = 1. (3 points)

c) Elaborate the advantages and disadvantages of pure stack machine vs n-register machines. (2 points)

ii) Consider an accumulator machine. Given the current state of the stack and accumulator, what is the next line of code to generate for the code fragment (2 * 3) + 5? Note: <init> is the initial symbol used to denote the bottom of the stack. (4 points)

Current: *Acc* 5

Stack 6, <init>

1. Push acc
2. Pop
3. acc <- 6
4. acc <- acc + top_of_stack

C4: Regular Languages, DFA, NFA (6 points)

i) Draw a DFA that accepts 00 and 11 at the end of a string containing 0, 1 in it, e.g., 01010100 but not 000111010. (3 points)

ii) Draw the corresponding NFA (3 points)

C5: Other topics discussed in Class – Combinators, Interactive Programming with Haskell, Programming Language Types, Recursion: (8 points)

i) What are the 7 steps that Prof. Graham Hutton recommends when thinking about writing recursive programs? Illustrate this on the example of writing a program that multiplies all the elements in an input list. (3 points)

ii) What is a programming style? What is the difference between imperative and functional style programming? (2 points)

iii) What are side effects? Provide an example of how side effects can be handled in Haskell programs? (3 points)