

## ASSIGNMENT 1 — BASICS OF LANGUAGES

COMP 3010 — ORGANIZATION OF PROGRAMMING LANGUAGES

### 1. COMPILERS AND INTERPRETERS

**Exercise 1.** Explain the task of each of the following phases of a compiler:

- (1) Lexical analyzer (*a.k.a.* lexer)
- (2) Syntax analyzer (*a.k.a.* parser)
- (3) Semantic analyzer
- (4) Intermediate code generator
- (5) Code optimizer
- (6) Machine code generator

**Exercise 2.** Which of the following are advantages of compilers? Which are advantages of interpreters?

- (1) Can pre-examine input program for semantic (*e.g.*, type) errors
- (2) Have full knowledge of both program input and program implementation
- (3) Can specialize parts of code to optimize for specific input data
- (4) Can afford heavy weight optimizations over large sections of code
- (5) Flexible, and can easily change program behavior dynamically at run-time
- (6) Generated code can run many times

### 2. COMPUTABILITY

**Exercise 3.** (1) If  $Q$  is a program that solves the halting problem, what must  $Q(P, x)$  do?  
(2) What is the significance of the halting problem for computable functions?

**Exercise 4.** Suppose that the program  $Z$  is a *zero checker*. That is,  $Z$  takes two inputs, a string  $P$  representing a program and a number  $n$ , and checks whether or not running  $P$  with input  $n$  (written as  $P(n)$ ) would output a 0:

$$Z(P, n) = \begin{cases} \text{true} & \text{if } P(n) \text{ returns } 0 \\ \text{false} & \text{otherwise} \end{cases}$$

Because of the *Full Employment Theorem for Compiler Writers*, we know that the *zero checker* is not a computable function. Show why the program  $Z$  cannot exist by explaining how it could be used to solve the halting problem.

### 3. INDUCTION

**Exercise 5.** Consider the total function

$$f(x) = 3x + 2$$

Which of the following inductive definitions are equivalent to the above  $f$ ?

- |     |                                     |
|-----|-------------------------------------|
| (a) | $f(x+1) = f(x) + 2$                 |
| (b) | $f(x+1) = f(x) + 3$                 |
| (c) | $f(0) = 3 \qquad f(x+1) = f(x) + 2$ |
| (d) | $f(0) = 2 \qquad f(x+1) = f(x) + 3$ |

**Exercise 6.** Recall that a natural number  $x$  is *even* whenever it is double another natural number (that is to say, there is some other natural number  $n$  such that  $x = 2n$ ). A natural number  $x$  is *odd* whenever it is one more than an even number (that is to say, there is a natural number  $n$  such that  $x = 2n + 1$ ).

Consider the *partial* function

$$h(x) = \frac{x}{2} \qquad \text{if } x \text{ is even}$$

where  $h(x)$  is undefined for odd numbers  $x$ .

Which of the following inductive definitions are equivalent to the above  $h$ ?

- |     |   |
|-----|---|
| (a) | $h(x+2) = h(x) + 1$                                 |
| (b) | $h(0) = 0 \qquad h(x+2) = h(x) + 1$                 |
| (c) | $h(1) = 1 \qquad h(x+2) = h(x) + 1$                 |
| (d) | $h(0) = 0 \qquad h(1) = 1 \qquad h(x+2) = h(x) + 1$ |

**Exercise 7.**  $\mathbb{N}$  stands for the set of all natural numbers, and  $\mathbb{N}^*$  stands for the set of *all* finite lists of natural numbers of any length. For example,  $\mathbb{N}^*$  contains each of the lists  $[9, 9, 9]$ ,  $[10, 9, 8, 7, \dots, 3, 2, 1]$  and  $[5, 16, 8, 4, 2, 1]$ .  $\mathbb{N}^*$  is defined inductively as the smallest set such that:

- $\mathbb{N}^*$  contains the empty list  $[]$ , and
- given any natural number  $x_0$  from  $\mathbb{N}$  and any list  $[x_1, \dots, x_n]$  already in  $\mathbb{N}^*$ , the set  $\mathbb{N}^*$  also contains the list  $[x_0, x_1, \dots, x_n]$ .

Give an inductive definition for the function  $f$ , which takes a finite list of numbers  $xs$  and  $f(xs)$  is double the sum of the list. Your inductive definition of  $f$  should give the same result as:

$$f([x_1, x_2, \dots, x_n]) = 2 \times (x_1 + x_2 + \dots + x_n)$$

In the case of the empty list  $[]$ , the sum of  $[]$  is 0.

Hint: you might find the fact that

$$2 \times (x_1 + x_2 + \dots + x_n) = (2 \times x_1) + (2 \times x_2) + \dots + (2 \times x_n)$$

is useful for writing your inductive definition of  $f$ .

**Exercise 8.** Let  $X$  be inductively defined as the smallest set of finite number lists (taken from  $\mathbb{N}^*$ ) satisfying the following closure properties:

- Given any natural number  $n$  from  $\mathbb{N}$ , the set  $X$  contains the one-element list  $[n]$ .
- Given any list  $[x_1, x_2, x_3, \dots, x_n]$  already in  $X$ , the set  $X$  also contains the list  $[x_1, x_2, x_3, \dots, x_n, y]$  where the additional element  $y$  is the product  $x_1 \times x_2 \times x_3 \times \dots \times x_n$ .

Which of the following lists is NOT in  $X$ ?

- (a) [1, 1, 1, 1, 1]
- (b) [2, 4, 16, 256]
- (c) [3, 3, 9, 81]
- (d) [4, 4, 16, 256]

#### 4. SYNTAX AND GRAMMARS

**Exercise 9.** Consider the grammar  $G$  (note that  $\epsilon$  stands for the empty string):

$$S ::= aS \mid T$$

$$T ::= bT \mid U$$

$$U ::= cU \mid \epsilon$$

- (1) Which of the following strings is generated by the grammar  $G$ ?
  - (a) **aba**
  - (b) **bab**
  - (c) **ca**
  - (d) **ccc**
- (2) Which of the following is a derivation of **bbc** in the grammar  $G$ ?
  - (a)  $S \rightarrow T \rightarrow U \rightarrow bU \rightarrow bbU \rightarrow bbU \rightarrow bbc$
  - (b)  $S \rightarrow bT \rightarrow bbT \rightarrow bbU \rightarrow bbcU \rightarrow bbc$
  - (c)  $S \rightarrow T \rightarrow bT \rightarrow bbT \rightarrow bbU \rightarrow bbcU \rightarrow bbc$
  - (d)  $S \rightarrow T \rightarrow bT \rightarrow bTbT \rightarrow bbT \rightarrow bbU \rightarrow bbcU \rightarrow bbc$

**Exercise 10.** Which of the following grammars describes the same language as  $0^m 1^n$  where  $m \leq n$ ?

- (a)  $S ::= 0S1 \mid \epsilon$
- (b)  $S ::= 0S1 \mid S1 \mid \epsilon$
- (c)  $S ::= 0S1 \mid 0S \mid \epsilon$
- (d)  $S ::= SS \mid 0 \mid 1 \mid \epsilon$

**Exercise 11.** Consider the following grammar for *abstract* syntax of arithmetic expressions:

$$E ::= E+E \mid E-E \mid E \times E \mid E/E \mid 1 \mid 2 \mid 3 \mid 4$$

with the usual associativity and precedence for the arithmetic operators (all operators are left-associative,  $\times$  and  $/$  have a higher precedence than  $+$  and  $-$ ).

- (1) Draw an abstract syntax tree for each of the following strings:
  - (a)  $1 - 2 + 3$
  - (b)  $1 \times 2 + 3$
  - (c)  $1 \times 2 - 3/4$
- (2) Which of the following pairs of strings with different parentheses represent the same abstract syntax tree according to the above precedence and associativity? Draw that abstract syntax tree.
  - (a)  $2 \times 4 - 3$  versus  $2 \times (4 - 3)$
  - (b)  $1 + 2 + 3 + 4$  versus  $1 + (2 + (3 + 4))$
  - (c)  $2 + 3 \times 4/2$  versus  $2 + ((3 \times 4)/2)$