PRACTICE EXERCISES 1 - CS 361

Introduction

1. Explain what a procedural language is and describe the major similarities and differences between different examples of procedural languages.

2. Explain what a function is and how a functional language offers another level of abstraction to the programmer.

3. Explain some of the difficulties that you imagine would arise when trying to write algorithms for parallel programming.

4. Explain the statement "the intent of logic programming is to concentrate on what is true rather the how to find it".

5. Match the language with the disadvantage.

FORTRAN	Low-level constructs can be difficult to understand.
COBOL	Require rather verbose code.
C	Does not have while-loops.

6. Which procedural language might be most appropriate for a program to do each of the following applications and why?

   (a) Compute trajectories for a satellite launcher.

   (b) Monitor an input device feeding data from an experiment to the computer.

   (c) Process the day's transactions at an ATM.

7. The database *vendor* contains information on vendors (name, address, phone number (*phone*), the name of the product line the vendor sells (*prodname*), the amount of stock purchased from that vendor during the previous business quarter (*purchase*)).

   In the vendor database the user wants to now all of the city where there are vendors from whom the store bought more than $10,000 worth of stock the previous business quarter. Write an SQL query for this information.

8. Here is the beginning of a Prolog program about a family. Facts are:

```
male(eli) - Eli is a male.
male(bill)
male(joe)
female(mary)
```

```
female(betty)
female(sarah)
parent-of(eli,bill) - Eli is Bill's parent.
parent-of(mary,bill)
parent-of(bill,joe))
parent-of(bill,betty)
parent-of(bill,sarah)
```

(a) Draw the family tree based on these facts.

(b) What are the results of the following queries:

```
?- male(mary).
?- male(john).
?- male(bill).
```

(c) Define a rule father-of.

(d) Define a rule daughter-of.

(e) Define a rule ancestor-of.

(f) Define a rule sibling-of.

(g) Define a rule cousin-of.

(h) What is the result of the following query?

```
?- ancestor-of(X,sarah)
```

9. Here is a Prolog program about the domain of American history. The interest is in which US presidents were in office, when certain events occurred and in the chronology of those president's in office.

```
president(lincoln, gettysburg)
president(lincoln, civilwar)
president(nixon, firstmoonlanding)
president(jefferson,lewisandclark) - Jefferson was US president during the
                                     Lewis and Clark expeditions.
president(kennedy, cubanmissilecrisis)
president(fdr,worldwar2)
before(jefferson, lincoln)
before(lincoln, fdr)
before(fdr, kennedy)
before(kennedy, nixon) - Kennedy was president before Nixon.
```

(a) Give a fact in the previous program.

(b) Give a rule in the previous program.

(c) What is the result of

```
?- before(lincoln,fdr).
```

(d) What is the result of

```
?- before(fdr,nixon).
```

(e) What is the result of

```
?- president(lincoln,x).
```

(f) Write a rule to express that x was president before y.

10. We could describe a stack of blocks by:

    $top(a, b)$ means a is on top of b.

    Write a rule to say that a is on b.

11. Give an example of SIMD processing.

12. Give an example of MIMD processing.

13. Which do you think is harder job:

    (1) Compiling and translating FORTRAN code into assembly language,

    (2) Compiling and translating PROLOG code into assembly language.

    Why? Explain.

Definition of language

1. Write a BNF grammar that describes the structure of a nonterminal called $<$ *number* $>$. Assume that $<$ *number* $>$ contains an optional + sign followed by exactly 2 decimal digits, the first of which cannot be a 0. Thus 23, +91 and +40 are legal but 0,+01 and 123 are not. Show a parse true for +90 using your grammar.

2. Write a BNF grammar for identifiers that consist of an arbitrary long string of letters and digits, the first one of which must be a letter. Show a parse true for AB5C8 using your grammar.

3. What are the 4 phases of compilation.

4. Pick a Java instruction or expression and discuss what happens during each phase of its compilation.

5. Give 5 expressions recognized by the following grammars.

```
<boolean> ::= <expr><op><expr>
<expr>::=<var>|<form>
<form>::= (<var> == <var>) | (<var> < <var>) | (<var> > <var>)
<var> ::= w | x | y | z
<op> ::= AND | OR
```

6. Write a BNF grammar to describe the following simplified input statement:

```
input >> var >> var >> ... >> var;
```

The statement begins with the word "input" followed by one or more sequences of the operator >> followed by a variable. The statement ends with a semicolon. Variable names are arbitrarily long strings of digits and letters, the first of which must be a letter.

7. Assume that we write the following pairs of assignments statement:

   $Delta = 2.9 + (a + b + c * 3)/(x - 5.7);$

   $Epsilon = (a + b + c * 3) + sqrt(3.1 * y);$

   How could a compiler optimize the execution of these 2 statements?

Recursion

1. Assume the following function definition:

```
mystery(n) = if n = 0 then 0
else n + mystery(n-1)
```

n is greater or equal to 0.

   (a) Why is this definition well-formed?
   (b) Why is this definition well-defined?
   (c) Compute $mystery(5)$.
   (d) If possible provide an explicit definition of $mystery$.

2.

$$A(x,y) = \begin{cases} y+1 & \text{if } x = 0 \\ A(x-1,1) & \text{if } y = 0 \\ A(x-1, A(x, y-1)) & otherwise \end{cases}$$

Compute $A(2,2)$ and $A(3,3)$.

3. F is from Z^+ to Z and $n \geq 1$. Z^+ is the set of positive integers.

$$F(n) = \begin{cases} 1 & \text{if } n = 1 \\ F(n\ div\ 2) & \text{if } n \text{ is even} \\ 1 + F(5n - 9) & \text{if } n \text{ is odd and } n > 1 \end{cases}$$

Show that F is not well-defined.

4. G is from Z^+ to Z and $n \geq 1$.

$$G(n) = \begin{cases} 1 & \text{if } n = 1 \\ G(n \ div \ 2) & \text{if } n \text{ is even} \\ 2 * G(3n - 2) & \text{if } n \text{ is odd and } n > 1 \end{cases}$$

Show that G is not well-defined.

SML

1. Write expressions or functions in SML that have the following types:

```
int
real
int*real*bool
(int*real)*bool
int -> int
(int*int) -> (bool*real)
```

2. Write SML functions that do the following:

    (a) $fahrToCels(d)$ that takes a real in representing a temperature in degrees Fahrenheit and returns its value in degrees Celsius.

    (b) $power(a, b)$ which is a recursive function that takes in two integers and computer a to the bth power.

    (c) $divides(a, b)$ that takes 2 integers and returns true iff a divides b.

    (d) $divByThree(a)$ that returns true iff the integer a is divisible by 3.