CS 3360 Design and Implementation of Programming Languages
Summer 2017, Test 1

Name:

1a. What was the first widely used programming language?  
1b. When was it developed?  
1c. What were its main limitations?

1. a) FORTRAN was the first widely used programming language.
   b) Developed in the 1950's (Fortran 0 in 1954, never implemented, Fortran 1 in 1957) by IBM.
   c) There was no data typing (certain letters variables represented floating point numbers, while the rest were integers), had a 6 character variable name limit, implementation was initially limited to IBM computers (IBM 704), and there was originally no ability for recursion.
2. What is the difference between imperative, functional, logic (declarative), and object-oriented languages?

**Imperative** languages use assignment statements and iteration, (e.g. C, Pascal, FORTRAN, COBOL...).

**Functional** languages use functions with given parameters, (e.g. LISP, Scheme, SML, Haskell).

**Logic** languages are rule based, specified in no particular order (e.g. Prolog, OBJ3), and

**Object-Oriented** languages encapsulate data objects with processing inheritance and dynamic type binding, an evolution of imperative languages (e.g. Smalltalk, Eiffel, C++, Java).
3a. Assuming that the notion of <variable> is already defined, describe boolean expressions in Java in BNF form. The only constants are true and false. (Feel free to ignore non-lazy operations.)

3b. Why do we need to describe languages in BNF form? Why cannot we just describe them in English?

\[
\begin{align*}
&\text{3a) } \langle \text{VAR} \rangle ::= \ldots \\
&\langle \text{aux} \rangle ::= \text{true} | \text{false} | \langle \text{var} \rangle | !\langle \text{var} \rangle \\
&\langle \text{and} \rangle ::= \langle \text{aux} \rangle | \langle \text{and} \rangle \& \langle \text{aux} \rangle \\
&\langle \text{B.Expr} \rangle ::= \langle \text{and} \rangle | \langle \text{and} \rangle \text{||} \langle \text{B.Expr} \rangle
\end{align*}
\]

b) by using BNF form, we can accurately describe the syntax of our languages while avoiding ambiguity. It is very difficult to do so in English without the language becoming excessively verbose. We can also compile
4a. Translate the following EBNF description into BNF:
\[
\begin{align*}
\langle \text{variable} \rangle & := \langle \text{letter} \rangle \{\langle \text{letter} \rangle | \langle \text{digit} \rangle | _\} \\
\langle \text{letter} \rangle & := a | \ldots | z | A | \ldots | Z \\
\langle \text{digit} \rangle & := 0 | \ldots | 9
\end{align*}
\]

4b. Why do we need EBNF?

4. a) BNF: \[
\begin{align*}
\langle \text{letter} \rangle & := a | \ldots | z | A | \ldots | Z \\
\langle \text{digit} \rangle & := 0 | \ldots | 9 \\
\langle \text{word} \rangle & := \langle \text{letter} \rangle \langle \text{word} \rangle | \langle \text{digit} \rangle \langle \text{word} \rangle | _\langle \text{word} \rangle \\
\langle \text{var} \rangle & := \langle \text{letter} \rangle \langle \text{word} \rangle
\end{align*}
\]

b) EBNF eliminates redundancy when describing a language, at the cost of increasing complexity (small cost)
5a. Show, on an example, that the following description of propositional expressions containing \( \| \) is ambiguous:

\[
\text{expression} ::= \text{variable} \mid \text{expression} \| \text{expression}.
\]

5b. Show how to make this description unambiguous.

---

5: a) \( \text{exp} := \text{var} \mid \text{exp} \| \text{exp} \)

Consider all \( b \| c \), it can be expressed:

\[
\begin{align*}
\text{allb} \| c & \quad \text{all} b \| c \\
\text{exp} & \quad \text{exp} \\
\text{var} & \quad \text{var} \\
\text{exp} & \quad \text{exp}
\end{align*}
\]

because the tree can branch in more than one unique way, the description is ambiguous.

b) to make the description unambiguous:

\[
\text{expression} ::= \text{variable} \mid \text{variable} \| \text{expression}
\]

Can only be a variable.
6a. What is the strongest postcondition for the following program: \( \{ a > 0.0 \} \ a = 2 \ast a + 3; \)

6b. What is the weakest precondition for the following program: \( a = 2 \ast a + 3; \ \{ a > 0.0 \} \)
7. Describe a state diagram for the computer to detect Java fixed-point real numbers.
8. Use bottom-up approach to parse the following expression: \((a \lor b) \&\& (c \&\& d \lor e) \lor f \&\& (g \lor h)\). Remember that "and" has higher priority than "or".

\[
\begin{align*}
& 8: \quad \text{Bottom-Up} \\
& (a \lor b) \&\& (c \&\& d \lor e) \lor f \&\& (g \lor h) \\
& \uparrow \\
& r_1 \leftarrow a \lor b \\
& r_1 \&\& (c \&\& d \lor e) \lor f \&\& (g \lor h) \\
& \uparrow \\
& r_2 \leftarrow c \&\& d \\
& r_1 \&\& (r_2 \lor e) \lor f \&\& (g \lor h) \\
& \uparrow \\
& r_3 \leftarrow r_2 \lor e \\
& r_1 \&\& r_3 \lor f \&\& (g \lor h) \\
& \uparrow \\
& r_4 \leftarrow r_1 \&\& r_3 \\
& r_4 \lor f \&\& (g \lor h) \\
& \uparrow \\
& r_5 \leftarrow g \lor h \\
& r_4 \lor f \&\& r_5 \\
& \uparrow \\
& r_6 \leftarrow f \&\& r_5 \\
& r_4 \lor r_6 \\
& \uparrow \\
& r \leftarrow r_4 \lor r_6
\end{align*}
\]
9. Use pre-and post-conditions to prove that the following Java program for computing $a + b$ is correct:

```java
int sum = a;
for (int i = 1; i <= b; i++)
    {sum++;

| Sum | i | Consider $\text{Sum}=a+(i-1)$ | and $i'=i+1$
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
<td></td>
<td>$\frac{\text{Sum}}{2}=a+(i-1)$</td>
</tr>
<tr>
<td>a+1</td>
<td>2</td>
<td>$\frac{\text{Sum}}{2}=a+(i-1)$</td>
<td>$\text{Sum} = \text{Sum}+1$</td>
</tr>
<tr>
<td>a+2</td>
<td>3</td>
<td></td>
<td>$\frac{\text{Sum}'}{2}=a+(i'-1)$</td>
</tr>
<tr>
<td>a+3</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a+(i-1)</td>
<td>i_n</td>
<td>LHS</td>
<td>RHS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\text{Sum}' = \text{Sum}+1$</td>
<td>$\text{Sum}'=a+(i'-1)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\Rightarrow \text{Sum}'=(a+(i-1))+1$</td>
<td>$\Rightarrow \text{Sum}' = a + ((i+1)-1)$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$= a + i$</td>
<td>$= q + i$</td>
</tr>
</tbody>
</table>
```

The condition $\text{Sum}=a+(i-1)$ is met before the first iteration (i.e. $\text{sum}=a$, $i=1$, $q=a+(i-1)=a+(0)=a$). The condition is also met before every following iteration, thus the condition will be met after every iteration.
10. Describe, step-by-step, memory allocation and values of all the variables for the program segment from Problem 9. Assume that \( a = 3 \), \( b = 2 \), that the variable \( b \) was the last variable allocated before this program segment, and that the hex address of the variable \( b \) is 2017.

```c
int sum = a;
for (int i = 1; i < b; i++)
    sum += i;

int i = 1;
while (i < b)
    sum += i;
```

<table>
<thead>
<tr>
<th>Step</th>
<th>a</th>
<th>b</th>
<th>i</th>
<th>sum</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
</tbody>
</table>

**Address:**
- \( a \): 0x8
- \( b \): 0x12
- \( i \): 0x1
- \( sum \): 0x12

**Notes:**
- \( int \): 4 bytes
- \( addr \): 8 bytes