Name: ________________________________
Student ID: __________________________

Signature: ____________________________

• You have 120 minutes to write the 12 questions on this examination. A total of 120 marks is available.

• **Justify all of your answers**

• You may use **one** sheet of handwritten notes.

• Keep your answers short. If you run out of space for a question, you have written too much.

• The number in square brackets to the right of the question number indicates the number of marks allocated for that question. Use these to help you determine how much time you should spend on each question.

• Use the back of the page for rough work.

• **Good luck**

<table>
<thead>
<tr>
<th>Question</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>15</td>
</tr>
<tr>
<td>5</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>19</td>
</tr>
<tr>
<td>10</td>
<td>16</td>
</tr>
<tr>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td><strong>120</strong></td>
</tr>
</tbody>
</table>

**UNIVERSITY REGULATIONS:**

• No candidate shall be permitted to enter the examination room after the expiration of one half hour, or to leave during the first half hour of the examination.

• **CAUTION:** Candidates guilty of any of the following, or similar, dishonest practices shall be immediately dismissed from the examination and shall be liable to disciplinary action.

  1. Making use of any books, papers or memoranda, electronic equipment, or other memory aid devices, other than those authorized by the examiners.

  2. Speaking or communicating with other candidates.

  3. Purposely exposing written papers to the view of other candidates. The plea of accident or forgetfulness shall not be received.
Question 1 [9 points]
• [2/9] How does simplicity and orthogonality relate to language design?

• [2/9] Recall that some languages are compiled, some are interpreted, and some are hybrid. Explain briefly how a hybrid language is implemented and give an example of a hybrid language.


(Question 1 continued on next page)
Miscellaneous

• [3/9] Under what conditions are the following two code snippets equivalent?

\[ a = (b > 0) \&\& (a++ > 0) \quad a = (b > 0) \&\& (a > 0) \]
\[ a++; \]

(End of question 1)
Weakest Precondition

Question 2 [7 points]

• [3/7] What is a weakest precondition?

• [4/7] Compute the weakest precondition $P$ for the following code:

\[
\{P\}
\]
\[
\text{if } (x > 0)
\]
\[
y = y - 1;
\]
\[
\text{else}
\]
\[
y = y + 1;
\]
\[
\{Q\} = \{y > 0\}
\]

(End of question 2)
Question 3 [13 points]

• [2/13] What is the advantage of using BNF to specify a grammar over simply using words.

• [3/13] What is the general appearance of a BNF rule? That is, show a general rule and explain what the different components can be.

• [4/13] Consider the following context free grammar:

\[
\begin{align*}
A & \rightarrow b B \\
& \quad \mid c B B b \\
& \quad \mid a \\
B & \rightarrow A y
\end{align*}
\]

and let $xA\alpha$ be a sentential form. Show the set of sentential forms (mixed strings) derivable in maximum two steps with left-most derivations starting with $xA\alpha$.

(Question 3 continued on next page)
Would you get the same set of sentential forms if you did right-most derivations instead? If yes, explain why, if no, show the set.

(End of question 3)
LL(1) Grammars

**Question 4** [15 points]

- [3/15] Consider the following context free grammar:

  \[ A \rightarrow A \ a \ | \ b \]

  Explain why this grammar cannot be implemented by a recursive descent parser?

- [3/15] Rewrite the grammar so that it can be implemented by a recursive descent parser (You may not change the language generated by the grammar).

- [3/15] Is the language regular? If no, explain why not. If yes, write the corresponding regular expression.
• [3/15] Why is

\[ A \rightarrow a \text{ } B \mid a \text{ } A \text{ } b \]

not LL(1)?

• [3/15] Rewrite it so it becomes LL(1) (You may assume that FIRST(A) ∩ FIRST(B) = ∅).

(End of question 4)
**Parsing**

**Question 5** [8 points]
Briefly explain the actions of a shift/reduce parser. That is, what happens when a shift, a reduce, an accept, or an error is performed. You may refer to the following general configuration:

\[ (s_0x_1s_1x_2 \ldots s_{n-1}x_ns_n, a_ia_{i+1} \ldots a_n\$) \]

and the rule

\[ A \to \alpha, \quad |\alpha| = k \]

<table>
<thead>
<tr>
<th>Shift:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reduce:</td>
</tr>
<tr>
<td>Error:</td>
</tr>
<tr>
<td>Accept:</td>
</tr>
</tbody>
</table>

(End of question 5)
Binding

**Question 6** [5 points]
- [3/5] List 3 of at least 5 binding times.
- [2/5] Using C/C++ give an example of a heap dynamic variable.

(End of question 6)
The Espresso Virtual Machine

**Question 7 [7 points]**

- [3/7] Explain how the EVM deals with method overloading. That is, what does the EVM have to do in order to call the right method when methods are invoked using the invokevirtual/invokenonvirtual/... instructions?

- [4/7] Give an example of a check that cannot be performed by the Espresso compiler, but must be done by the virtual machine (i.e., EVM).

(End of question 7)
Scoping

**Question 8** [6 points]
Consider the following program:

```c
int a = 2;
int b = 9;

void f(int a) {
    a = a + 3;
    g(a);
}

void g(int c) {
    b = c + a;
    a = a + 1;
}
```

- [3/6] What are the values of the global variables \(a\) and \(b\) after the call \(f(a)\) with static scoping?

- [3/6] What are the values of the global variables \(a\) and \(b\) after the call \(f(a)\) with dynamic scoping (You can assume that they start out at 2 and 9 again, respectively)?

(End of question 8)
Syntax

**Question 9 [19 points]**
- [3/19] What is the dangling else problem?
- [4/19] Give 2 different ways to design a language to avoid the dangling else problem.
- [3/19] Give 3 different iterative language constructs and show their general form.

(Question 9 continued on next page)
Syntax

• [4/19] Rewrite the following while-loop to a for-loop (Hint: It is not as easy as it looks!):

```c
a = 10;
while (a>0) {
    a = a -1;
    ...
}
```

(Question 9 continued on next page)
Syntax

• [5/19] Write a BNF that can generate the following string with a parse tree that looks like the one below. That is, + and * take precedence over /::

\[ A + B/C * D \]

(End of question 9)
Data Types

**Question 10** [16 points]
• [2/16] What is the difference between primitive/atomic types and abstract data types?

• [2/16] What is an ordinal type?

• [5/16] Does C allow you to create ragged 2-dimensional arrays of integers? If no, explain why not. If yes, write C-code for creating this array:

(Question 10 continued on next page)
Data Types

• [3/16] Explain how a union type can be implemented using a record type.

• [4/16] Sketch a possible method for a (interpreted) language to avoid dangling pointers (This is a hard question, leave it for now if you don’t know!)

(End of question 10)
Side Effects

**Question 11 [8 points]**

- [3/8] Java (and Espresso) allows lines like

  \[
  a = b = c = d = <expr>;
  \]

  How does the code generated for EVM deal with this?

- [3/8] Is the following always equivalent to the code line from the previous subquestion? Explain why/why not.

  \[
  a = <expr>;
  b = <expr>;
  c = <expr>;
  d = <expr>;
  \]

  (End of question 11)
Subprograms and Parameters

**Question 12 [7 points]**

- [4/7] Consider a new language called Latte. Latte is exactly like Espresso expect you can specify different return types for overloaded functions. In Espresso (and Java) it is required that all overloaded functions must return the same return type. For example, the following program will compile in Latte:

```java
public class A {
    public int f() {
        return 7;
    }
}

public class B extends A {
    public String f() {
        return "Hello";
    }
}

public class Main {
    public static void main() {
        int k;
        A a = new B();
        k = a.f();
    }
}
```

Explain why this compiles without errors, but why it cannot possibly run without errors.

(Question 12 continued on next page)
Subprograms and Parameters

• [3/7] Consider a language with call-by-result (i.e., IN/OUT) call semantics. What is the problem with the following code:

```plaintext
proc f(int a, int b) {
    a = a - 1;
    b = b + 1;
}
int c;
f(c,c);
```

(End of question 12)
(End of the exam)