Exam Structure

• The exam consists of two Sections A and B
  – Section A: Rolf’s half: 30 marks
    – 6 questions
  – Section B: Robert’s half: 30 marks
    – 4 questions
• As a rule of thumb:
  – Each mark requires about 3 minutes of work.
Section A Questions

• Typical topics:
  – Key concepts required to develop speech applications
  – Dialog design, prompts, and grammars
  – VoiceXML and VoiceXML platforms
  – VoiceXML and JavaScript
  – Mixed Initiative in VoiceXML
  – VoiceXML versus SALT
Q1: Barge-in

- Q1: Name advantages and disadvantages of the barge-in function and explain what consequences barge-in has for the prompt design. [2 marks]
Q1: Solution

- Advantages: Barge-in
  - enables callers to interrupt before the end of the prompt
  - provides a more user-directed dialog
  - can dramatically speed up a verbal conversation.
- Disadvantages: The caller
  - will not hear the rest of the prompt
  - may therefore miss important information.
- Consequences:
  - design prompts so that important information appears early in the prompt
  - put most important information at phrase boundary (turn taking rule).
Q2: Prompts

- Q2: Explain what problem the following prompt might have and suggest a better solution. [2 marks]

Please choose a car. The available choices are a Hyundai Accent from Avis, a Mitsubishi Lancer from Thrifty or a Nissan Pulsar from Hertz, or say 'back' to change the car size.
Q2: Some Hints

• This prompt is too complex.
• Difficult to keep an expression such as *Mitsubishi Lancer from Thrifty* in mind.
• Instruction
  
  *say 'back' to change the car size.*

  follows the prompt.
• Select first the car size:
  
  – small, medium, large
Q3: Grammars

• Q3: Write a modular XML grammar that allows the caller to confirm or deny a request in at least 3 different ways. [3 marks]
<?xml version="1.0" encoding="ISO-8859-1"?>
<grammar root="main" version="1.0">
  <rule id="main" scope="public">
    <one-of>
      <item><ruleref uri="#yes" tag="yes"/></item>
      <item><ruleref uri="#no" tag="no"/></item>
    </one-of>
  </rule>
  <rule id="yes">
    <one-of>
      <item>yes</item>
      <item>yeah</item>
      <item>yep</item>
      <item>sure</item>
    </one-of>
  </rule>
  <rule id="no">
    <one-of>
      <item>no</item>
      <item>not</item>
      <item>nope</item>
    </one-of>
  </rule>
</grammar>
Q4: VoiceXML

- Q4: Write a VoiceXML application (with XML grammars) that collects preference data from a caller after a usability test is completed. This document should collect data for the following questions:

  [10 marks]

  - Do you want to use this application again?
  - Would you recommend this application to a friend?
  - On a scale of 1 to 5 with 5 being the highest score, how would you rate the usability of this application.
  - On a scale of 1 to 5 with 5 being the highest score, how would you rate the usefulness of this application.
Q4: A Partial Solution

```xml
<?xml version = "1.0"?>
<vxml version = "2.0">

<form id = "questionnaire">
  <!-- Add explanation about the purpose of the questionnaire -->

  <field name = "question1">
    <prompt>
      Do you want to use this application again?
    </prompt>
    <grammar src = "yes_no.grxml"
      type = "application/srgs+xml"/>
  </field>

</form>
</vxml>
```
... 

<field name = "question3">
  <prompt>
    On a scale of 1 to 5 with 5 being the highest score, how would you rate the usability of this application?
  </prompt>
  <grammar src = "scale.grxml"
    type = "application/srgs+xml"/>
</field>

...
Q4: A Partial Solution

...  

<filled>

<prompt>
    Thank you. We appreciate your help.
</prompt>

</filled>

</vxml>
Q5: Spoken Output

- Prompts are one type of spoken output of a SLDS. Name four other types of spoken output and explain the function of each type [4 marks].
Q5: Solution

<table>
<thead>
<tr>
<th>Type</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prompt</td>
<td>Indicates it is time for user input, and thus serves as a turn-taking cue</td>
</tr>
<tr>
<td>Feedback</td>
<td>Presents the app state that results from user input, allowing the user to compare original intent with result</td>
</tr>
<tr>
<td>Instructions</td>
<td>Provide information about operating the user interface or understanding the task</td>
</tr>
<tr>
<td>Help</td>
<td>Help instructions often adopt a separate mode or state aimed at coaching the user</td>
</tr>
<tr>
<td>App Data</td>
<td>Information presented to the user as part of the task: eg weather, stock information, flight times</td>
</tr>
</tbody>
</table>
Q6: SALT

• Describe in full English what functionality the following SALT code provides and explain in a step wise manner how the execution model (RunAsk function) works.
Q6: SALT

<body onload="RunAsk()">
<form id="travelForm">
<input name="txtBoxOriginCity" type="text" />
<input name="txtBoxDestCity" type="text" />
</form>

<!-- Speech Application Language Tags -->
<salt:prompt id="askOriginCity"> Where would you like to leave from? </salt:prompt>
<salt:prompt id="askDestCity"> Where would you like to go to? </salt:prompt>

<salt:listen id="recoOriginCity" onreco="procOriginCity()">
    <salt:grammar src="city.xml" />
</salt:listen>

<salt:listen id="recoDestCity" onreco="procDestCity()">
    <salt:grammar src="city.xml" />
</salt:listen>
Q6: SALT

<!--- scripted dialog flow -->
<script>
  function RunAsk() {
    if (travelForm.txtBoxOriginCity.value==""") {
      askOriginCity.Start();
      recoOriginCity.Start();
    } else if (travelForm.txtBoxDestCity.value==""") {
      askDestCity.Start();
      recoDestCity.Start();
    }
  }

  function procOriginCity() {
    travelForm.txtBoxOriginCity.value = recoOriginCity.text;
    RunAsk();
  }

  function procDestCity() {
    travelForm.txtBoxDestCity.value = recoDestCity.text;
    travelForm.submit();
  }
</script>
Good luck in the exam!